**React Js Info**

1. We have to download the following extension
   1. React Developer Tool(for chrome and from chrome)
   2. Thunder Client
   3. Prettier- code formator
   4. ES7 React/Redux/GraphQL/React-Native snippets
   5. Bracket Pair Colorization Toggler
   6. Auto rename tag
2. npm is an package manager and it help us to install packages and comes by default while downloading the node.js

🡪Open console in the folder of project OR open windows power shell in the folder and write the following statement :

npx create-react-app my-app

\*Here the npx used is right package

🡪And then wait till the package is installing and then done and make the workspace of vs code inside that folder

1. In react js the states means the data given in the website
2. In react js the prompts means the structure given in the website
3. To run the website go to the open the app folder in vs code🡪src🡪index.js🡪console🡪write npm start🡪React app will start

🡪If error comes and the app does not load than check the direction of the terminal folder it should have the app name at the last and if it is not then change the location of the folder by the help of the cd command

1. Always prefer the Microsoft edge for the development instead of the chrome as some times in the chrome the statements on the console is not printed
2. We can also check the performance of the web app by selecting tools from the browser and for it we have to go to browser and there we have many selection like sources, console, network(No throteling🡪check different networks), performance, memory, security, **lighthouse(make an report before publishing)**
3. To create the production-ready app, we can use the code npm run build
4. The App.js is a javascript file with the function in it and, in which the html is written and further in which we can also write the js language in the {} brackets, this file is called as the JSX

Eg. name=”neel”;

🡪In jsx

<h1>Hello <name></h1>

1. The className=”container” is used to make all the components in the perfect order and align all of them in the center
2. In JSX instead of the class we have to write the className as the class is the reserved keyword in the js
3. In jsx we can not write the for, we have to write the htmlFor instead
4. In JSX, we can return only the one tag and the other tags should be covered into the other tags like :

<>

<h1>My name is neel shah</h1>

<div className=”neel”>

<p>Hello friends…</p>

</div>

</>

1. To use the bootstrap in the react project we have to add the following files to it:
   1. Bootstrap javascript cdn link to the lower part of the body of the public🡪index.html
   2. And bootstrap css cdn link to the upper part of the body in public🡪index.html
2. We have to change the href=”#” to the href=”/” in the JSX which is in the src🡪app.js
3. It is advised that in the react app make an folder named component and then add the components file in it in .js extension and make the name of component start with the capital word
4. Now we can make the different **components** in the react, so if we want the navbar we will get it by only using the navbar tag from an component folder in which the code will be already written

🡪Here the Navbar tag which will be generated is know as the alias but in this documentation the word alias is not used instead the tag word is used for it

🡪By the help of it we can make the length of code small and reuse the part of the code

🡪Steps for using the component

1. Make an components named folder in the src folder
2. Make the file as <Component-name>.js but **take care that the component name would have the first letter capital**
3. Now in the <component-name>.js, we have to write the **rcf and enter to get the react component based function**, due to which our file will now work as the component
4. And now in between the div tags, we can write our code of the navbar
5. And then go to the app.js in the src folder and then make the use of the tag which is similar to the file name in the components folder you want to use. i.e. Here to call the code we use <Navbar/>
6. And now the component is visible to the our website or webpage
7. Now we can use the **Props in the components**, due to which we can make the some change in the component file while using it, and not to edit it

🡪This an effective way to make small changes in the component prototype or sample component

🡪The props can be implemented in the project by the two ways the first is the function and the other is the class

🡪The following instrunctions till the page 8 and point 42 is related to the react function based component and then after it the things are for the react class based component(rcc)

🡪We can directly use the props by making first the component tag and in it we have to give the value which we want to set

Eg. <Navbar title=”Util Text”/>

🡪Here the Navbar is component’s name

🡪title is the thing which’s value we want to set

🡪And Util text is the value which we want to set

🡪Steps for making the prop(s) in the component

1. Make the change in the function type in the <component-name>.js file like:

Export default function Navbar() 🡪 Export default function Navbar(props)

1. And where you want to set the value of the porp there make the following change of the red color :

<a className=”my” href=”/”>{props.title}</a>

🡪Here the title is the value that we are giving from where it is called in the form of the prop

🡪And do not forget to put the curly braces, as it shows that it is the js

🡪Here the file is of js in which we write the html and by the help of the curly braces, we use the js again

1. We can also pass the more than one props in the component by the same process

🡪We have to just change the thing which we have to give in the component file to the {props.<name-of-thing-to-five>}

🡪And also change the tag for calling the component to :

<Navbar title=”Util Text” name-of-thing-to-five=”value”/>

\*Here we have given the two props in which the one is added to the above example

🡪Tada, this works

1. Now we can also define the value of the different props, So by mistake no one can give the wrong value

🡪The steps for it are :

1. In the component file first we have to import the prop types, We can do it by writing impt and then enter and the import statement will be there

🡪Note that have to import this on the top of the file

1. Now at the end of the file, we can declare the type of the proptype, by the help of the :

<component-file-name>.propTypes={title.PropTypes.string.

aboutText.PropTypes.string}

🡪Here the .propTypes, should be written with the p small

🡪And the other two should be written with the P capital

\*Here the title and the aboutText are the prop name

1. We can also set the default prop value in the component file

🡪The following are the steps for it :

1. Write the code in the last of the component file :

Navbar.defaultProps={

title: “Set title Here”,

aboutText: “About text here”

};

\*Here the title and the aboutText are the prop name

1. In react, we can use the state to change the value of anything without loading the page

🡪The steps for it are:

1. Change the import statement for react from to: (Hooks)

import React from ‘react’ 🡪 import React, {useState} from ‘react’

1. Make an variable having state inside the main function of the file:

Eg. const [text, setText] = useState(“Enter the text here”); (Ghokhvanu)

🡪By the help of this function, we can change the value of variable from one state to another, having the name of the first state

🡪We cannot directly update the value of the variable

🡪We can do it by the help of the function of the state :

Eg. as per the above example of state :

setText(“cwekmkmekl”);

1. Example program of the event :

import React, {useState} from 'react'

export default function TextForm(props) {

  const [text, setText]=useState("");

  const handleUpClick =()=>{

    console.log("Upper case was clicked");

    let newText=text.toUpperCase();

    setText(newText);

  }

  const handleOnChange=(event)=>{

    console.log("On Change");

    setText(event.target.value);

  }

  return (

    <>

    <h1>{props.heading}</h1>

    <div className="mb-3">

    <textarea className="form-control" id="exampleFormControlTextarea1" rows="8" value={text} onChange={handleOnChange}></textarea>

    <button className="btn btn-primary my-4" onClick={handleUpClick}>Convert to upper case</button>

    </div>

    </>

  )

}

🡪The important concepts of the events are highlighted

1. For making the change in the JSX component in the java script, we use the curly braces but in it we can not use the template literal i.e. ${} for putting the value of variable

🡪We can also use it by the help of the backtiks :

Eg. <nav className={`navbar navbar-expand-lg navbar-${props.dark} bg-dark`}>

🡪Backticks are present below the esc button

1. We have to use the htmlFor instead of the html in jsx
2. We can write in console in jsx by : {console.log(porps.mode)}
3. We can use the logical terniary operator like :

${props.mode==='light'?'dark':'light'}

🡪In this if the mode is equal to light than it will change to dark , otherwise to light

1. We can extend the background in the js to the whole page by writing the following code in the App.js in the src :

document.body.style.backgroundColor=”gray”;

1. In js all the variables are considered as the objects

Eg. Here the dual brackets are used as the inner one are objects

<div className="container" style={{color: props.mode==="dark"?"white":"black"}}>

1. We can handle the error of the null by the help of the following code for the alert :

props.alert && <..code from the bootstrap..>

<strong>{props.alert.type}</strong>: {props.alert.msg}

</div>

1. Once we dismiss the alert, the alert cannot be seen again, it is closed for all website

🡪We can also overcome this problem by not having the close button on the alert and setting the time out method and hence the alert will go away itsef

1. We can make some changes in the title name of the website, website logo and etc things related to the website by going into the :

Public🡪index.html🡪and change the relative things of the site and this change will affect the sco rating of the site

🡪Change carefully the content which is in the meta tag which acts as the keyword for which the site is related to…

🡪We can make an favicon by simply going into the google search and searching the favicon generator and uploading the our icon and by downloading the .zip file we can get various icon of it

🡪We use the .ico file in the zip file for the favicon of the website

1. We can change the title of the page by the help of the following code :

document.title=”Page Name”;

🡪By the help of this function, we can make the change in the title like, Gmail-(2)unread by the help of the JavaScript

1. We can make the title of the page change after some time, like the virus add in the chromes by making this code in some button :

setInterval(() => {

document.title = “Virus Detected”;

}, 2000)

setInterval(() => {

document.title = “Install this app now”;

}

1. **React router** is used to not make the whole page reload but some part of the page to be reload and some part of the page to remain intact and this facility is not provided in the simple js or html page, i.e. our website will work on the principal of the youtube in which the only content page is reloaded but not the index of menu part
2. For making the react application we can follow the following steps :

🡪The steps for this are: (for the cocumentation [React Router | Quick Start (reactrouterdotcom.fly.dev)](https://reactrouterdotcom.fly.dev/docs/en/v6/getting-started/overview))

1. Paste the following statement which is in the point 2 in the terminal of the vs code
2. **npm install react-router-dom**
3. It will take some time and then done, don’t give any attention to the error that comes like vulnerabilities
4. Make the change in the App.js as followed :
5. add the following import statements below the other imported statements :

import React from "react";

import {

BrowserRouter as Router,

Switch,

Routes,

Route,

Link

} from "react-router-dom";

1. Then add the router tags on the starting and ending of the return of the render function and the tags are like <Router>…</Router>, This tags are after and before <>or <div>
2. Then add the <Routes> tags between the div components
3. And in that write the code for linking, the demo is as followed :

<Routes>

<Route exact path=”/about” element={<About mode={mode}/>}/>

<Route exact path=”/” element={<TextForm heading=”xyz” showAlert={showAlert}/>}/>

</Routes>

1. Now, we have to replace the ‘<a> and href’ tags by the ‘<Link> and to’ in the **navbar.js OR any other component in which there is link** we have used the linking process
2. We also have to import the following line or it may get imported automatically in the navbar.js(OR **any other component in which there is link**):

import {Link} from ‘react-router-dom’;

1. In navbar.js (or any other component with link) we have to set the text which is showed in the navbar as button and the link by clicking which the navbar follows it

🡪While in the app.js we have to give the link and the alias of the class or file where to go in route tag

1. If there is error while running the app like **'react-scripts' is not recognized as an internal or external command,**

**operable program or batch file.**

🡪Then use the following code i.e the error will be gone : npm install react-scripts

1. The router of the react does not work good with the github pages, it is okay if you deploy the site on the server
2. Steps for uploading the app to the github pages
   1. Make an repository on the github
   2. Open package.json and add the code on the top bellow the {

"homepage": “<https://neeldevenshah.github.io/Text-Utils>”,

\*Here the username and the myapp name is to be changed and my-app name is the name of the repository on the github in which we want to deploy the project

\*Here the username should be written in all the small characters as the link do not follow the uppercase characters (spent 2 days to find it), And if used it will give the blank page

* 1. Then install the things by entering the following code in terminal of vs code:

npm install --save gh-pages-

* 1. Then add the following code in the scripts of the package.json

"scripts": {

"predeploy": "npm run build",

"deploy": "gh-pages -d build",

}

* 1. Then run the code in the terminal of the vs code :

npm run deploy

* 1. And then go to the github depository in which the project is saved and there go to its setting🡪pages🡪set the brach as the gh-pages and /root and save🡪After some time the website will be live till some time it will show up 404 page error
  2. Tada the project will be live

1. The blank page can also be seen while development, if we are not running the server in the src folder of the project, As the terminal by default do not go into the src folder, To change the path we can use cd command
2. In react js the vulnerabilities comes do not take care of them and follow the instructions given any thing
3. If we want to deploy the **react-router website to the paid server** than we do not have to follow the above command for deploying websie on the github

🡪And if done than:

🡪Delete the installed --save gh-pages by write the following code to uninstall it :

**npm uninstall gh-pages**

🡪We have to also remove the sentences we have added in the package.json file

🡪And we can easily use the react-router technology in the paid server, So if want to use than use it

🡪For the paid server we don’t have to add the all the folders of the project we only have to add the build folder

🡪For it follow the bellow step(Baki)

1. We can also make use of an filter to anything in js, Example:(**In jsx, if in js than do not use {}**)

**<p>{text.split(“ ”).filter((element)=>{return element.length!=0}).length}</p>**

🡪Filters are used to filter any thing

1. In index.html in the public folder we have to define the app name and app content and which is relevelent to the app, So app can make up in the SCO Ranking
2. We can also disable an button or any function, If the nothing is written in it by :

<button disabled={text.length===0} className=”btn btn-primary mx-1 my-1 onClick={handleUpClick}”>Convert to uppercase</button>

🡪If the value of the statement inside the braces is true than the disabled will be on

🡪If the value of the statement inside the braces is false than the disabled will be off

1. We can also use the logical condition in the output like :

<p className=”container”>{text.length>0?text : “Please enter some text”}</p>

1. The following code returns the value of the target or changed value that triggered it:

setText(event.target.value)

1. In onclick method we can not give the function call, we can only give an function
2. In react till here we have work with the function based components but now we will work with the class based components
3. We can make the class based components in the App.js by first removing the inbuit function and export and then write rcc and enter we will get it by the help of the ES7 React Snipit extension in vs code
4. The render method in the App.js file compiles the html code in JSX and then make the html to render
5. For the **props for react class-based component(RCC):**
6. In this type of the components we have to add the statement starting from the let keyword then in curly braces and in it the variables that are coming in the form of the props and which we want to access and then equals to the this.props and then we can use it in the rcc and example of it is as follow(**Alternative Method exists and it is easy check it after the second point**) :

**let {title, description}=this.props;**

1. The second change that is the usage of the props variable that is different from that of the RFC:

{props.title} 🡪RFC(React function based component)

{title} 🡪RCC(React class based component)

🡪In the RCC we can also fetch the value of the variables by : {this.props.title}

1. Some times in the JSX if there the error is coming than we have to use the double curly braces as shown in the example :

<div className="card" style={{width: "18rem"}}>

🡪This is an part of the JSX of RFC OR RCC and in it if it gives error that one have to write in the following format

🡪In which the first brace indicates the js and the second one html

1. State is used when we want to change the value of any thing many times in an program
2. While the props is used when we want to set the value of any thing one time and do not make it change again
3. The work of the props and state is some what same but they are used for different purposes as told earlier
4. State is a plain javascript object used by the react to represent an information about the component’s current situation

🡪We can set the state of any variable by :

this.state={

articles : this.articles

}

1. We can also set the state of the object or variable by the help of the this.setState :

🡪Example: this.setState={

article:this.article2  
}

1. By the help of the following code we can get the value which is stored in the articles named variable and which is stored by the help of the state

{this.state.articles.map((element)=>{

            return <div className='col-md-4'>

              <NewsItem target="\_blank" key={element.url} title={element.title} description={element.description} imgUrl={element.urlToImage} url={element.url}/>

              </div>

            })}

1. componentDidMount() is an life cycle method which is executed after the render method is executed, So first constructor is executed than render and at last the componentDidMount() is executed
2. The Fetch API provides an interface for fetching resources (including across the network). It will seem familiar to anyone who has used [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest), but the new API provides a more powerful and flexible feature set.

Eg. of the both fetch and the async function and also componentDidMount()

async componentDidMount(){

    let url="https://newsapi.org/v2/top-headlines?country=in&apiKey=90e7f116117d4e6aa91f10765c013282&pageSize=21";

    let data=await fetch(url);

    let parsedData=await data.json();

    console.log(parsedData);

    this.setState({articles: parsedData.articles, totalArticles: parsedData.totalResults})

  }

1. And as in java and js both the data should be in the form of the json after doing the transportation from the one to another than only the language can read it and hence due to that the data in the above example is converted to the json form
2. An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.
3. If we use the react class based component than for calling the function or method we have to used the **this.<method-name>**

🡪Here the &larr; is used to get the symbol of the left arrow and &rarr for next arrow

Eg. <button type=”button” onclick={this.handlePrevClick}>&larr; Pervious</button>

<button type=”button” onclick={this.handlePrevClick}>&rarr; Back</button>

1. We can also disable an button by :

<button disabled={this.state.page<1} class=”btn btn-primary onClick={handleCli}”></button>

1. In function the async cannot be written in the normal way, It can be written as :

handleNextclick=async ()=>{

    console.log("Next");

    if(this.state.page+1<=Math.ceil(this.state.totalArticles/20)){

    let url=`https://newsapi.org/v2/top-headlines?country=in&apiKey=90e7f116117d4e6aa91f10765c013282&page=${this.state.page+1}&pageSize=21`;

    let data=await fetch(url);

    let parsedData=await data.json()

    console.log("Parsed Data");

    this.setState({

      page: this.state.page + 1,

      articles: parsedData.articles

    })

    }

    else{

    }

1. The Math.ceil function makes the small or the point value to be roundeoff and make it similar to the highest integer of it

Eg. if(this.state.page+1<=Math.ceil(this.state.totalArticles/20))

1. And if we are using the state than we can take the value from the state by using the method like this.state.<variable-name> and it’s example is as shown in above point
2. Backticks are very useful to make the changes like this in the url by the help of the template literal and this facility is not been provided in the “” type. Eg

let url=`https://newsapi.org/v2/top-headlines?country=in&apiKey=90e7f116117d4e6aa91f10765c013282&page=${this.state.page-1}&pageSize=21`;

1. In the react class based component we can change the value of the variable by the help of the this.setState method

Eg. this.setState({mode: “dark”});

🡪While in react function based component(rfc) we don’t have the methods like the state and setState and we have to make the variable like :

Const [state, setState]=useState(“default-value-of-state”)

🡪For changing the state we use: setState(“<value>”);

🡪The methods for both the rfc and rcc are different and we cannot use the one in another

1. We can make the condition like when the first condition is true than only the second condition will be implemented without using the if else by the help of the && operator and example of it is as follow :

**{this.state.loading && <Spinner>}**

🡪In this type of the operator the id the first condition is proved right than only the second will be implemented otherwise not

1. We can set the download spinner in the website, For it steps are:
2. Download the spinner from google by writing free spinner gif
3. Then add it to the component folder of the project
4. Then make an file named the spinner.js in the component folder
5. And in it import the file by an name example:

**import spinner from ‘src\components\spinner.js’**

1. And then make the <div></div> tags after the return and in between them add:

**<img src={spinner} alt=”loading”>**

1. Then add the following tag where you want to make the spinner to roll all the time:

**{this.spinner.loading && <Spinner>}**

1. Then on the constructor put the following things

**constructor(){**

**this.state={**

**loading : false;**

**}**

**}**

1. And then where we want to play the spinner there add the following line:

this.setState({loading : true});

1. And then add the following statement where you want to stop loading

this.setSet({loading : false});

1. Then to make the old content go away while loading new one we will use another logic operator like

{**!**this.state.loading && this.state.articles.map((element))…}

**\*Here the first statement should be as it is written and the second one should be customized as per the task we want to make after loading**

🡪Hence while the loading is true the old data will be gone

1. Remount/re-render
2. In the react-router there is a problem that if we are using the same alias for more than one time in the route having the props to send then there is an high possibility that it do not re-mount or re-render to give the different information that we want each time we click the different link, But in this problem the clicking of the link will have no effect to the page but only url will be changed

🡪This problem occurs as the router would think that the all the alias do the same thing and hence it will not re-render or re-mount

🡪This problem can be solved by using the keyword key inside the alias which would be unique for each of the same alias we use

<Route exact path="/" element={<Heading key="heading" mode={this.state.mode} mode2={this.state.mode2} newsSize={6} country={'in'} akey="90e7f116117d4e6aa91f10765c013282"/>}/>

1. React Component lifecycle : The sereies of events that happens from the mounting of a React component to its Unmounting
   1. Mounting : Birth of your component
   2. Update : Growth of your component
   3. Unmount : Death of your component
2. The main methods in react component lifecycle :
   1. The render() method is used to render HTML of the component in react. This method is required for a class based component to render the DOM. It runs during the mounting and updating of your component. render() method should be pure i.e. you cannot modify state inside it!
   2. The componentDidUpdate() method runs after the component output has been rendered to the DOM.
   3. The componentDidUpdate() method is invoked as soon as the updating happens. The most common use case for the componentDidUpdate() method is updating the DOM in response to prop or state changes
   4. The componentWillUnmount() lifecycle method method is called just before the component is unmounted and destroyed. Usally used to perform cleanup
3. The process for adding the infinite scroll to the web-app
4. First search the infinite scroll in the google and open the link of the website of the npmjs.com
5. First we have to install the npm package by following the instruction given in the documentation
6. It is hard to implement the external package in the project but if the example is given in that external package than see it and make some copy paste from that code and it will be easy to implement in the project
7. So if there is any example project for that package than open it an then copy the code from it as it will be easy to be implement in the our project and if any does not exist then make it from the sample code given
8. In this case there is an sample project named CodeSandbox and we will enter it and copy the alias named InfiniteScroll in which there are various props and will also copy the ending alias but inbetween this we will put our code which we want to be make to see while loading
9. We also have to import the package named infinitescroll, But we will copy the import line from the sample project into the our project
10. The we have to make some changes in the props of the alias Infinite scroll
    1. dataLength should be changed as per our data
    2. in next there is the function to be called while scrolling to get more data, We have to make a new function named as it is the next i.e. fetchMoreData and in it we have to put the logic to load the data
    3. Now, there is the hasMore prop in which we have to tell the program that if we have the more data or we have extinguish the data, And if there is more than only it will make the alias to work otherwise not
    4. And the last one is the loader in which we will change the text in it to the Spinner alias
    5. And now we have an choice that we can also put an line or text when there is no data to load, For this we have to add the following prop to it :

endMessage={

        <p style={{ textAlign: 'center' }}>

          <b>Yay! You have seen it all</b>

        </p>}

* 1. And in the app if the horizontal scroll bar appears than make the <div className=”container”></div> around the main maping logic and the scroll bar will be removed and if the articles are not aligned in rows than also use the <div className=”row”></div> around the maping logic
  2. We have to also set the page number +1 by the help of the setStat,

This.setState(

{page: this.state.page+1}

)

1. From the www.npmjs.com we can take any component for the free and use in the our website, eg. infinite scroll, react-top-loading-bar, etc
2. Now the steps for the react-top=-loading-bar
   1. Open this search on the npmjs.com
   2. Then install the package by the help of the link given in the website
   3. If the error comes like unable to resolve dependency tree while installing the package than first use the following statement :

**npm config set legacy-peer-deps true**

* 1. And then again use the install statement and the error will be gone and we can easily use the package
  2. In some of the packages,In sample code there is option that we want to write the code with ref OR with state in this situation give the priority to the **with state** one
  3. Then import the needed packages on the top of the file
  4. Then we have to insert the alias with the LoadingBar name from an sample file and make some changes in it
  5. In the sample alias that we downloaded the function was written inbuilt but we will remove it and then make a new function on the out of the render function like :

 setProgress=(progress)=>{

    this.setState({progress : progress})

  }

* 1. And then we can also pass this function as the props to the other component and there we can handle it by : **this.props.SetProgress(50);**

🡪And we can set the value of the bar different at the different positions

* 1. And also we have to change the state of the progrees in the alias like :

**Progress={this.state.progress}**

1. What are the react hooks?

🡪React hooks are features of class based components in function based components

🡪It allows you to use state and other React features without writing a class

🡪Hooks are the functions which “hook into” React state and lifecycle features from function components

🡪Commonly used react hooks are :

* 1. **useState**

**const [state, setState] = useState(initialState)**

🡪During the initial render, the returned state (state) is the same as the value passed as the first argument (initialState).

🡪The setState function is used to update the state. It accepts a new value and enqueues a re-render of the component

🡪The setState can be used as :

**setState(newState);**

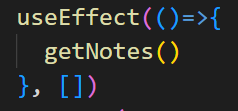
* 1. **useEffect**

**useEffect(didUpdate);**

🡪Accepts a function that contains imperative, possibly, effectful code

🡪By using the useEffect, you **tell React that your component needs to do something after render**

🡪The example of the useEffect is as follow, And here in the square or rectangle brackets we can write the condition at which we want to make the effect to happen, and this bracket can be empty if one want to make the function to implement one time, And it will be implemented after the render function is implimented



🡪By default, useEffect runs after every completed render, but you can choose to fire them only when certain values have changed

🡺Cleaning up an effect

🡪Often, effects create resource that need to be cleaned up before the component leaves the screen, such as a subscription or time ID. To do this, the function passed to useEffect may return a clean-up function. For example, to create a subscription:

**useEffect(()=>{**

**const subscription = props.source.subscribe();**

**return ()=>{**

**//Clean up the subscription**

**Subscription.unsubscribe();**

**};**

**});**

🡪The clean-up function runs before the component is removed from the UI to prevent memory leaks. Additionally, if a component renders multiple times (as they typically do), the previous effect is cleaned up before executing the next effect. In our example. This means a new subscription is created on evey update. To avoid firing an effect an on every update, refer to the next section.

🡺Timings of the effect

🡪Unlike componentDidMount and componentDidUpdate, the function passed to useEffect fires after layout and paint, during a deferred event. This makes it suitable for the many common side effect, like setting up subscriptions and event handlers, because most types of work shouldn’t block the browser from updating the screen

* 1. **useContext**

**const value = useContext(MyContext);**

**[Not have enough information]**

**🡪**

* 1. **useRef**

**[Not have enough information]**

* 1. **useMap**

**[Not have enough information]**

* 1. **useIsFirstRender**

**[Not have enough information]**

1. In an react app there can be both the class based component and the function based component at the same time
2. To make an app the function based component from the class based component, We have to remove all the this.props and have to also remove the render method from it

🡪And instead of the **export class Navbar extends Component** we have to write **const Navbar=()=>** & if there are the props in the component than put the props in the curved bracket of the function and if one want to access the any item from the props than do it by: **props.item-name**

**🡪**And at the end add the following statement **: export default <component-name>**

🡪In this process we have to change the value of the constructor to this type of thing and this would be different for each of the element which is in the constructor, Eg.

**const [articles, setArticles]=useState([])**

🡪And in this useState we have to put the initial value that we had put in the constructor

1. **MERN** stands for MongoDB, Express, React, Node, after the four key technologies that make up the stack

🡪MongoDB – document database

🡪Express(.js) – Node.js web framework

🡪React(.js) – A client-side javascript frameeork

🡪Node(.js) – The premier javascript web server

🡪There exist the other stacks like MERN which are MEAN, MEVN, etx

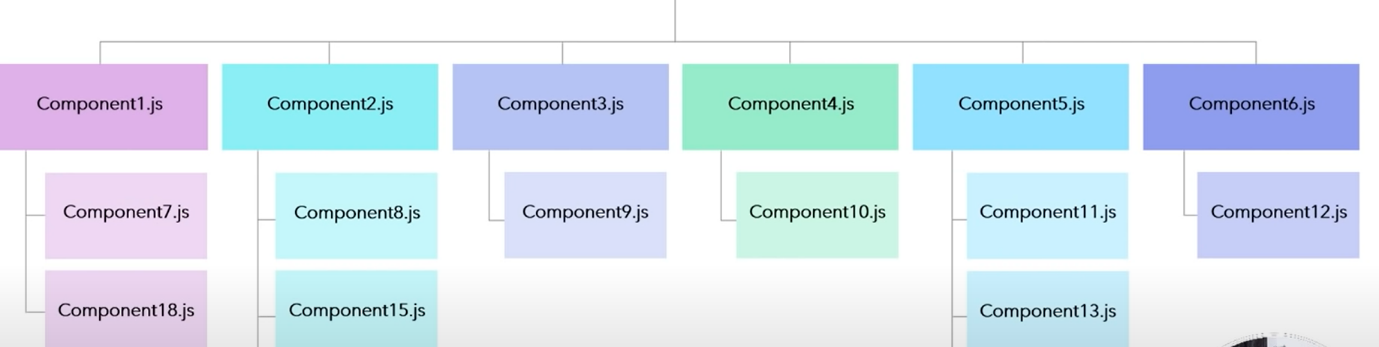
1. In react we can run the two servers together by making following changes in the outer package.json which is on the project folder not in the backend folder

🡪In it we have to add the following lines under the scripts section, originally we have to add the codes that we have to run but in this project we are running the two servers for it the command is:

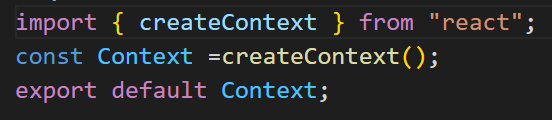
"both": "concurrently \"npm run start\" \"nodemon backend/index.js\""

1. And we can run both the servers together by help of the **npm run both**
2. Whenever we make any application for the production then make sure that we keep the frontend and backend of that application on the different folders only
3. The javascript follows the asynchronized function and it works on the un-blocking nature, hence some time occurs that if we write an statement first and it will be executed after some other statements as that first statement would have taken more time than the other and hence the statement after it will be executed first than the first one will be executed
4. While using the react-router technology the error comes if we had written the alias of the navbar outside the Router tag and we are using the link and to in the navbar
5. In a typical React application, data is passed top-down (parent to child) via props, but such usage can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application. **Context** provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.
6. And by the help of the context api we can pass the component of one to another easily without any pain of making of the props

🡪For example we can easily send an thing form the component 12 to component 25 without the use of the props and without passing any value via component2 and component 8



1. As we make the new folder for making the components in the react, It is also suggest that **we have to also make an new folder in the project named context and it is good practice to make it**
2. **The steps for making the context and using it in a file:**
   * + 1. We will make a file named Context.js
       2. And in it we will put the following things :



**Context.js**

* + - 1. And the **next step** is to implement the usage of the context is that we have to add the **<Context.Provider value={state-or-variable-name}>…</Context.Provider>** alias in the starting and the ending of the return function in the **App.js**
      2. Now we have to import the following statement in the file where we want to use the context :

**import { useContext } from ‘react’**

* + - 1. And then we have to import the **Context** file(If other file than use that’s name) in which we had configure the context, Eg.

**import Context from ‘../Context’**

🡪It changes as per file name and file’s location, don’t copy it

* + - 1. Now we have to write the following statement in the main function which is named same as the file name, but make sure you write the following statement above the return method

**const a= useContext(Context)**

🡪 Here the file name used inside the curved brackets should be same as that of the imported file in the above case.

* + - 1. And then in the App.js file we have to write the object or the variable above the return that we have to send in the form of context
      2. And then where we want to use that context, there we have to write the following statement above the return function

const a=useContext(Context)

🡪Here instead of the a we can use the any name of the variable in which we want to store the value

🡪And instead of the context, if there is another file used then we have to use that file name after importing it

* + - 1. And then we can use the context inside the return function of that file, Example

return(

<div>

My name is {a.name} {a.surname}

</div>

)

* + - 1. And now we can easily use the context’s value

1. We can highlight the name of the page in the navbar according to which page we are by the help of the **useLocation** hook

🡪And we can **highlight the text in the navbar** by the help of the **active** keyword in the className of the Link

🡪The steps for using this functionality are as follow :

1. First in the navbar.js, we have to import the useEffect from react

**import {useEffect} from ‘react’**

1. And we have to also import the useLocation from the react-router-dom

**import {useLocation} from ‘react-router-dom’**

1. Write the following statment before the return method

**let location =useLocation();**

1. And then we have to use the location.pathname for changing colour by use the conditional operator, for example

**<Link className={`nav-link ${location.pathname===”/about”?”active”:””}`} to=”/about”>About</Link>**

🡪Here we had used this function for the about, but we can also use for many other things also by changing the url and the Text name

1. If we want to make the change the pointer to the hand like pointer on hovering anything (for example we consider doing this effect on the I tag)

🡪For doing this we will add the following code in the index.css

**i{**

**cursor : pointer;**

}

1. For getting the value while writing any text we can use the following function :

**const onChange=(event)=>{**

**//As there are three dots, it is said to be the spread operator**

**setNote({…note, [event.target.name]: event.target.value})**

**}**

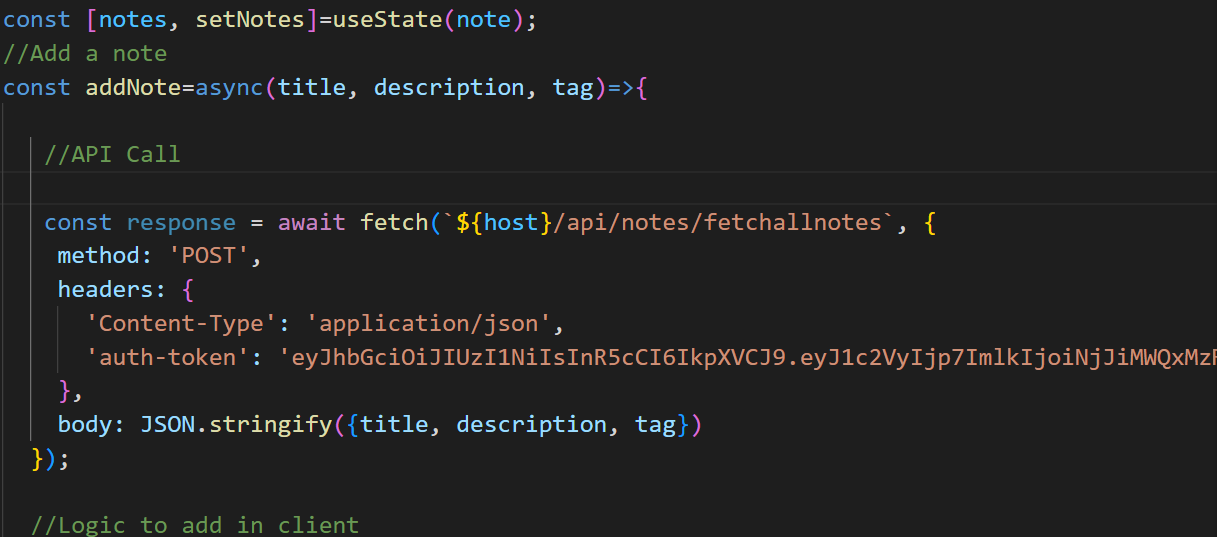
**🡪And where we are writing we have to put the following function in the input tag**

**<input type=”text” className=”form-control” id=”tag” name=”tag” onChange={onChange}/>**

🡪And we can store this values in an const state example:

const [note, setNote]=useState({title:””, description:””, tag:””})

1. We can use the api to get or fetch the data from the database, And for we have to follow the following steps
2. Search the **fetch with headers** in the google, And open the **documentation link of the developer.mozilla.org** which had commed during the google search
3. And we can use the documentation also **OR** the other way is to write the following statements(Recommended)





1. Now if the error comes regarding the **CORS POLICY** , then to fix it we have to follow the following steps :
2. We have to write the **how to fix cors policy in express**, And then we have to open the link which is from the expressjs.com and we will follow it
3. Now write the following statement in the terminal of the backend folder

🡪i.e. If you are not in the backend folder but you are in the main project folder than you can move to the backend folder if it is in the main project folder by writing the **cd .\backend\**

🡪And if you are in the backend folder only than directly write the following statement directly

**npm install cors**

1. And then open a the backend folder and in it go to the **index.js** and there write the following three statements before the **app.use(express.json()) satment**

**var cors=require(‘cors’)**

**var app=express()**

**app.use(cors())**

1. And then the app is ready to go, the error related to the cors will be removed
2. We can get the error or the special box like pop-up in the screen in the website by the help of the bootstrap’s modal
3. By the help of the useRef we can give the refrence of the one thing to the other, Example the one button is been clicked and by the help of the refrence the more than one button can be operated,

🡪Fist make const outside the return method like,

**const refExpt=useRef(null);**

🡪Now add the **ref={refExpt}** to the button or thing you want to make done automatically without clicking it

**<button type=”button” ref={refClose} className=”btn btn-primary”>Close</button>**

🡪Now add the following statement under any function, hence when the button is clicked the function will be maked worked and than the our button will also work

**const onClicle=(event)=>{**

**event.preventDefault();**

**refExpt.current.click();**

**}**

🡪Hence, by clicking the one button the function of the both the buttons is performed, this is useful in the bootstrap’s modal

1. We can set the value of the any thing in the html by the help of the value={}, Example

**<input type=”text” value={note.title} onChange={onChange}…/>**

1. The json.parse() is used for exchanging the data to/and a web server
2. The json.stringify() method is used to convert a JavaScript object or value to a JSON string
3. We can also make the taking the value of the input compulsory by the help of the **required keyword**

**<input type=”text onChange={onChange} required”/>**

1. We can also make the compulsory minimum length of the input by the help of the minLength keyword

**<input type=”text onChange={onChange} minLength={5}”/>**

1. We can also make the compulsory maximum length of the input by the help of the maxLength keyword

**<input type=”text onChange={onChange} maxLength={5}”/>**

1. The maxLength, minLength, required does not work if there is the onClick() on the submit button instead of the onSubmit(), So to use this methods we should compulsory have the onSubmit() method

🡪The main thing in the using the onSubmit method is that it can be used only in the form and the button should have the type=”submit”

🡪And the onSubmit function is used in the form tag not in the input tag, Example

**<form onSubmit={handleSubmit}>**

**🡪**Now if one want to use the onClick() and also use the conditions, than the other way is by the usage of the **disabled keyword**

**<button type=”submit” disabled={note.title.length<5 || note.description.length<5} onSubmit={handleClick}>Add</button>**

1. After clicking any button like submit or any thing if you want the page to not be redirect and be remain same as it is than use the preventDefault as the function which is called on the click:

**const handleSubmit=(event)=>{**

**event.preventDefault();**

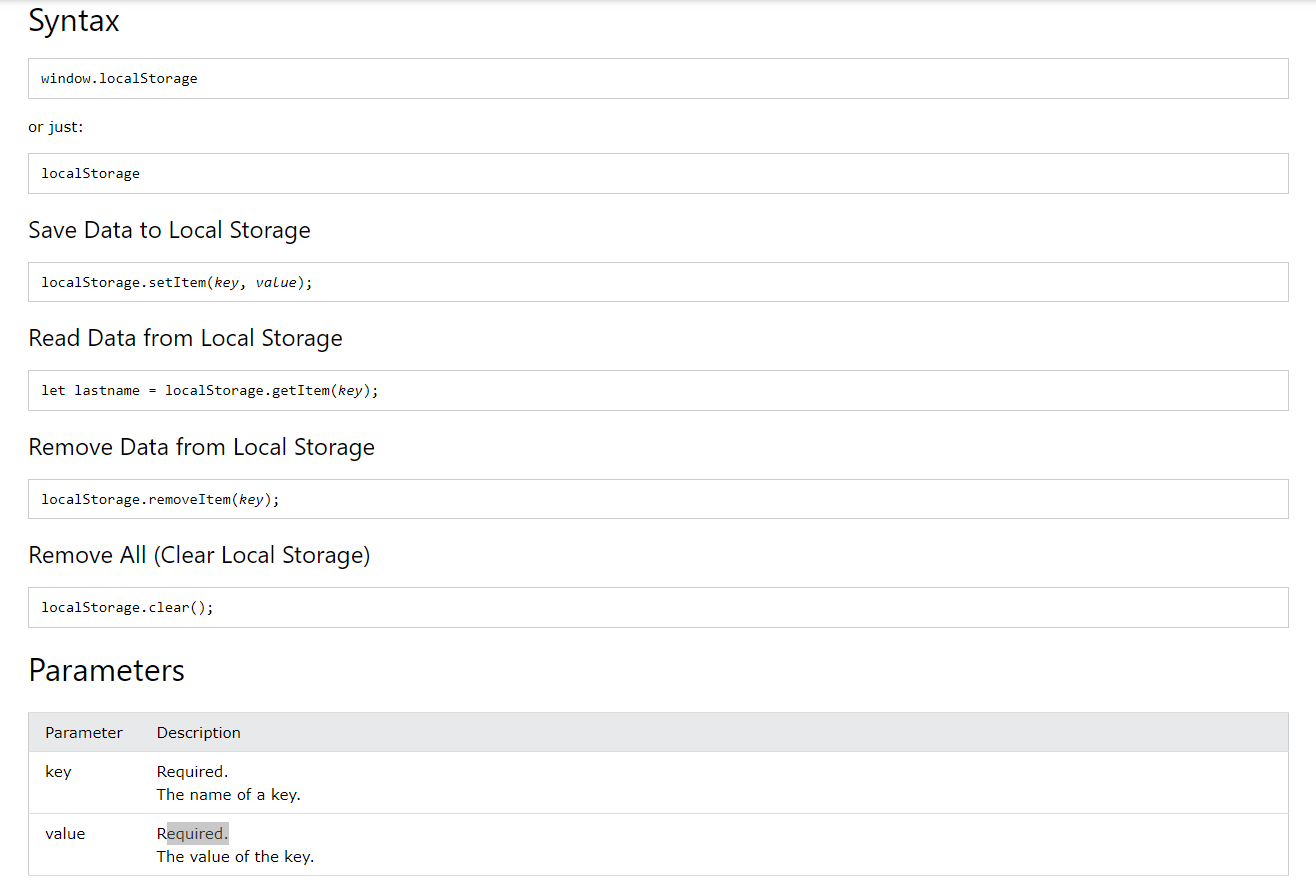
**}**

1. The fetch() method in JavaScript is used **to request to the server and load the information on the webpages**. The request can be of any APIs that return the data of the format JSON or XML
2. For converting the form data like email, password into the josn we first have to give the name of that input and then by the help of the following statement, We can store it in the json file by the help of the following statement,

**body: JOSN.stringify({email, password})**

1. localStorage is **a property that allows JavaScript sites and apps to save key-value pairs in a web browser with no expiration date**. This means the data stored in the browser will persist even after the browser window is closed.

🡪Some of the useful information regarding the local storage are :



1. The useNavigate hook **returns a function that lets you navigate programmatically, for example after a form is submitted**

🡪previously it was known as the useHistory, but nowadays it is known as the useNavigate

🡪One can use the useNavigate hook, by using the following syntax’s:

1. Import the following syntax on the top of the program with the other imports in the file where you want to navigate

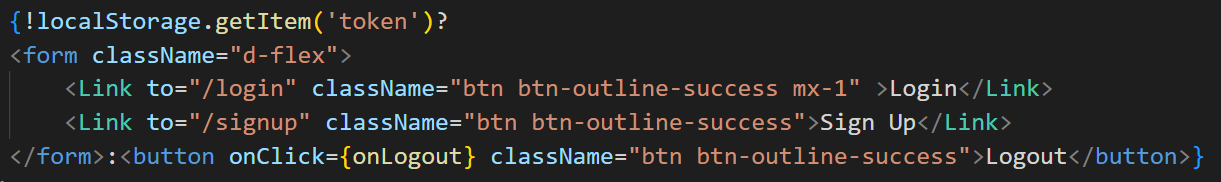
**import { useNavigate } from 'react-router-dom';**

1. Then write the following statement on the starting of the function but before the render function

**const navigate = useNavigate();**

1. Then write the following statement from which you want to navigate to the other page

**navigate('/home');**

1. We can make an page user secure by the help of the auth-token and the navigate functions
2. The example of the change in the options between the login, signup and the logout in the JSX is :

🡺**Redux**

1. **Redux** is used for managing the state of the application when it grows larger, For more information about the redux refer the redux info
2. For the more information about the **backend API making** see the inotebook backend info or the Backend API Info if maded
3. To use the imported react application from github we have to first go to the terminal and write the **npm install** to install all the dependencies required to run the application
4. We can also change the state to the opposite, if the variable is having the Boolean value by using the following logic :

setShowCart(state){

            state.showCart =!state.showCart;

        },

1. In jsx we can give the style to the tag by, example

**<img src=”map\_image.png” style={{width: ‘100%’}}></img>**