এই word file ব্যবহার করার নিয়ম:

টপিক কে .... এর মধ্যে রাখতে হবে। লেখা গুছিয়ে সুন্দর করে উপস্থাপন করতে হবে।

react-tutorial এই নামে git hub এ কোড রাখবো।

npx create-react-app my-app

npx create-react-app . দিলে সম্পূর্ন অ্যাপটি আমি যেই ফোল্ডারে আছি। তাকে ডাউনলোড হবে।

cd my-app

npm start

তবে git থেকে clone করলে nodeModule আনার জন্য ঃ npm install

git থেকে clone করার পরে npm start দিলে কাজ করে না। তখন npm i -g react-scripts লিখে enter দিলে কিছু packege ডাউনলোড হবে। তার পরে npm start ব্যবহার করবো। ইনসাআল্লাহ

html link --> style.css

html script src -->src/index.js type=”text/jsx”

reactDom.render

redner একটি মাত্র element কে নিয়ে কাজ করতে পারে।

index.js শুধু app কে render করবে।

jsx and js expression : html ও css এর মধ্যে javascript ব্যবহার করতে চাইলে js এর ভেরিয়েভল এর নাম second breack এর মধ্যে লিখতে হবে।

import React from "react";

const CodeWithHarry = () => {

  return (

    <div>

      <p>I am Ismail</p> এতোটুকু হলো jsx । একাধিক লাইন হলে ( ) ; এর মধ্যে রাখতে হয়। এই jsx এর মধ্যে কখনো ; হবে না।

    </div>

  );

};

export default CodeWithHarry;

function base component ব্যবহার করলে react কে import করার প্রয়োজন নাই।

সবধান: প্রতিটি fileName এর প্রথম word capital রাখতে হবে। নয়তো কোড কাজ করবে না।

inline css ব্যবহার :

<h1 style = {{”}}> toto app </h1>

const headingStyle = {

color: “red”, fontSize : “3rem

}

css কোড এর জন্য class = “ “ ব্যবহার করা হয়। কিন্তু react এ css কোড এর জন্য className=” “ ব্যবহার করা হয়।

className = “first-class-name second-class-name ....... “

public folder এ css ফাইল কে html file এর সাথে link করে দিলে তাকে সব জায়গা থেকে access করা যাবে।

src folder এ css ফাইল তৈরি করলে তাকে App থেকে access করার নিয়ম:

import ‘./fileName’

React এর অবজেক্ট হিসাবে CSS ব্যবহার করা যায়।

কোন ফাংশন রিটান করলে তাকে component বলা যায়।

export and import theory.

export default Card.

import ImportName from ‘./location’

index.js --> App.js --> controll all components

Props and destructuring

<a herf = “ “ > </a>

<Card titleText = “Call Mother” />

titleText, herf এগুলো হলো attributes Names.র

Card component এ props এর মাধ্যমে value রিসিভ করবো।

const {titletext, descText} = props;

Mapping data to components

json file কে নিয়ে আমার এই ভিডিওতে কিছুটা question তৈরি হলো।  
Data নামে import এবং maping

Data --> map --> item

map এক লাইনে লিখলে { ও return দেওয়ার প্রয়োজন নেই।

তবে একাধিক লাইনে লিখলে { ও return দিতে হবে।

অনেকগুলো data --> map --> যতগুলো ডাটা আছে সবগুলোই আসবে একটা একটা করে ।

for loop ব্যবহার করলে key={x} দিলে সমাধান হয়ে যাবে কিন্তু ম্যাপে কি করতে হবে:

{ Data.map((item, index) => <Card key = {index} titleText={item.title} descText= {item.desc} />}

map করলে একটা key ={index}

mapping data with a unique id

npm install uuid

import {v4 as uuidv4} from ‘uuid’; ES6 এখন ES7 চলছে।

<div key = {uuidv4()}>

বর্তমানে require ব্যবহার করা হয়।

nested map ব্যবহার :

Data --> map --> map

es7

rafce.............................. অ্যারে ফাংশন

rafcp ............................. অ্যারে ফাংশন with propsType

rcc..................................

rconst ....................... for class component

react developer tools chrome extension

How to use bootstrap use in react

class component state.

constractor--> initialState && use this every state.

disable={count ===0 ? true : false} html এর মধ্য javaScript code লেখা হয়েছে।

conditional rendering | ternary, short Circuit

{isLoggedIn && <HomePage />}

Event Handler | onChange |onClick ‍স্টেটকে value হিসাবে ডিক্লেয়ার করলে ।

react এর নিয়ম অনুসারে event handler এর জন্য যেই function ব্যবহার করবো তাকে parametar হিসাবে e দিতে হবে।

<button onClick={handleClick}> + </button>

const handleClick = (e) => {

setText = e.target.value;

}

Event handler binding

ES6 এর ফাংশন ব্যবহার করলে binding করতে হয় না। তবে normal function তৈরি করলে তার সাথে binding করতে হবে।

React Hooks | useState hook

useState Import করতে হয় react থেকে।

const [count, setCount] = useState(0);

stateফাংশন component এর ভেতরে লিখতে হবে।

dynamic styling in React

const error = false;

return (

<div className = “App”>

<h1 style={{color : error ? “red” : “green” }} > wellcome </h1>

</div>

useEffect এর ডিপেনডেন্সি হলো কোন state এর value পরিবর্তন হলে এটা সাথে সাথে কাজ করবে।

className = {`${validInput ? “valid” : “invalid”}`}

Controlled Component | Collect form data

FORM --> FORM.js

form

label htmlFor

input

**Name** --- **Email--password**

**button**

**একটি Component এর js ফাইল এর জন্য css ফাইলকে import : import style from “./form.module.css”; ব্যবহার করার নিয়ম: className={style.formGroup}**

**onchange ব্যবহার করলে ফাংশনের parameter হিসাবে e নিতে হবে।**

**e.preventDefault();**

**পরবর্তী ভিডিও টিউটিয়ালে আমরা শিখবো কিভাবে state এ object নিয়ে কাজ করতে হয়।**

**useState with object**

**const [user, setUser] = useState({name: ‘ ‘, email: ‘ ‘, password: ‘ ‘})**

**const {name, email, password}= user;**

**setUser({name: e.targe.value, email, password});**

**এখন handleSubmit এ user কে প্রিন্ট করবো।**

**এই ভিডিও থেকে কি কি শিখলাম?**

**১. useState with object**

**২. প্রতিটি input tag এ value এর যেকোন পরিবর্তন একটি function এর মধ্যে যাবে।**

**৩. ‍spreed oparetor use && name এর উপর ভিত্তি করে কিভাবে input এর value নিতে হয়।**

|  |  |  |
| --- | --- | --- |
|  |  | export default function USESTATE\_OBJECT() { |
|  |  | const [user, setUser] = useState({ userId: "", fullName: "" }); |
|  |  | const { userId, fullName } = user; |
|  |  | const handleChange = (e) => { |
|  |  | setUser({ ...user, [e.target.name]: e.target.value }); |
|  |  | }; |
|  |  | return ( |
|  |  | <div> |
|  |  | <form> |
|  |  | <input |
|  |  | type="text" |
|  |  | name="userId" |
|  |  | value={userId} |
|  |  | onChange={handleChange} |
|  |  | /> |
|  |  | <input |
|  |  | type="text" |
|  |  | name="fullName" |
|  |  | // value={user.fullName} |
|  |  | value={fullName} |
|  |  | onChange={handleChange} |
|  |  | /> |
|  |  | <h1> |
|  |  | User: {userId}, {fullName} |
|  |  | </h1> |
|  |  | </form> |
|  |  | </div> |
|  |  | ); |
|  |  | } |

**passing data from child to parent component**

**props এর মাধ্যমে যেই নামে data পাঠাবো সেই নামেই রিসিভ বা ডিসট্রাকচার করতে হবে।**

**child to parent data pass**

**const handleChildData = (childData) => {**

**মেথোড এর মাধ্যমে data আনার জন্য আমাদের একটা paramiter লাগবে যার মধ্যে আমরা data কে ‍store করে রাখতে পারবো।**

**data কোথায় থেকে আসতেছে? child.js থেকে।**

**}**

**<Child onChildData={ handleChildData } /> এটা parent.js ফোল্ডার এর কোড।**

**card component code**

**const data = “I am from child Data”;**

**props.onChildData(data);**

***এই ভিডিও থেকে আমারা কি শিখতে পারলাম:***

***কিভাবে props এর মাধ্যে methode pass করে child থেকে data parent এ আনতে হয়।***

***onযুক্ত props name ব্যবহার করা ভাল।***

***create a basic todo app:***

app.js--> HOme.js(todos=[“item1”, “item2”]) -> todos -> todo

components -> NewTodo

toggle.js

<p>

show and hide

useEffect

import করতে হবে এবং একটি অ্যানোনিমাস ফাংশন লাগবে।

প্রতিবার যখন রেন্ডারিং হবে তখন useEffect প্রতি বার কাজ করবে।

তবে [ ] ডিপেনডেন্সি সেট করলে সেই ‍state এর value পরিবর্তন হলেই কেবল useEffect কাজ করবে।

তবে একাধিক ডিপেনডেন্সি সেট করার জন্য (,)কমা ব্যবহার করা হয়।

react এর জন্য html tag এর মধ্যে onযুক্ত function লিখার নিয়ম:

<button onClick={handleClick}> + </button>

setCount((count) => count +1);

fetch data

fetch কে বলবো তোর ‍res কে json formate এ আমার সামনে পেশ কর। এই res এর মধ্যে থাকা data গুলোকে Todos এর মধ্যে পাঠিয়ে দে।

fetch(“url”)

.then(res) => {

return res.json()

})

.then(data) => {

setTodos(data);

})

setTimeOut function এর মধ্যে যে কোড থাকে সেগুলো background এ run হয়। নির্ধারিত সময় শেষ হলে তা front এ চলে আসে।

এখানে নির্ধারিত সময় এর মধ্যে data লোড হতে না পারলে কি হবে?

front-end এ error আসবে কি/ নাকি সময় নিয়ে তার পর প্রিন্ট হবে।

.catch((error) => {  
setError(error.message)

});

error কিভাবে আসবে। throw করলে।

how to create a custom hook

diff diff api এর জন্য hook কে ব্যবহার করা।

hook এর নামের ফাস্ট লেটার small ব্যবহার করলে সমস্যা হবে কি? আনিছুর স্যার ব্যবহার করেছে।

DataFetch ---> useFetch

useFetch এ আমারা url দিবো কিন্তু আমার useFetch থেকে কি কি পাব?

রিটান করলে আমরা যা যা রিটান করা হবে সবকিছু DataFetch এ ব্যবহার করতে পারবো।

useEfftect : component এর প্যারামিটার হিসাবে url দিতে হবে।

এখানে

১. const [data, setData] =useFtech(“https:..............................”);

১.১. url pass করা হলো।

২. পরে useEfftect থেকে object আকারে

return { data, isLoading, error }; রিটান করা হয়। তাই DataFetch এ

 const { data, isLoading, error } = useFetch(

    "https://jsonplaceholder.typicode.com/todos"

  );

রিসিভ করা হয়।

৩. ডিপেনডেন্সি হিসাবে এখানে url ব্যবহার করা যায়।

৪. hook এর সাহায্যে এভাবে data pass করার জন্য first letter small case হয়।

ধন্যবাদ custom hooks কে।

fragment

<>

life cycle of a class component

Mounting --> constructor --> render --> componentDidMount

updating --> state/props – ‍shouldcomponentUpdate --> render {শুধু render call হবে}

ref in a class component with example.

ref directed dom কে ম্যনিপুলেট করতে সাহায্য করে।

ref তৈরি করা:constructor এর মধ্যে  this.userNameRef = createRef();

এবার লিংক করে দিতে হবে।

<input type="text" id="userName" ref={this.userNameRef} />

console.log(this.userNameRef.current.value);

    this.userNameRef.current.style.color = "green";

**React Hooks | useRef Hook**

**react থেকে import করতে হবে।**

ref ব্যবহার করা হয় class component এর জন্য । কিন্তু useRef ব্যবহার করা হয় function component এর জন্য। ধন্যবাদ।

useref ব্যবহার করে dom কে access ও modify করতে পারি।

তৈরি করা: const userNameRef = useRef();

<input type="text" id="userName" ref={userNameRef} />

এখানে ref attribute হিসাবে কাজ করবে।

useRef দিয়ে শুধু মাত্র dom এর value খুঁজে নিবো। এটা দিয়ে dom কে ম্যনিপুলেট করবো না। তাহলে প্রোসেস slow হয়ে যাবে। এটাকে control করা হয় না।

const handleSubmit = (event) => {

    event.preventDefault();

    const userName = userNameRef.current.value;

    const password = passwordRef.current.value;

    userNameRef.current.style.color = "red";

    console.log({ userName, password });

  };

useReducer Hooks

index.js ফাইল এর সুবিধাঃ একটি পাথ কম হবে। index.js পাথ না দিলেও react auto নিয়ে নিবে। স্যার পরে অন্য ফাইল এর চেন্স করলেন তাতে কোন সমস্যা নাই। UseReducer

books, modalText, isModalOpen তিনটি স্টেট তৈরি না করে একটি মাত্র reducer দিয়ে তিনটির কাজ করতে পারি। আরে বেশি থাকলে তাও করতে পারি।

প্রথমে ‍state ব্যবহার করে কাজটি করবো। তারপর useReducer

booksData

books -> id, name

books --> book --> name

bookName

form get book name

map এর রিটার্ন এর পূর্বে distracture করা যায়।

handleSubmit\*\*\*

setBookName এর মাধ্যমে BookName এর মধ্যে value দিয়ে দিছি। এখন এই value কারো প্রয়োজন হলে সে BookName থেকে নিবে।

setBookName এর মধ্যে অ্যানোনিমাস ফাংশন ব্যবহার করলে একলাইনে করলে return ও { দিতে হবেনা একাধিক লাইন ব্যবহার করলে দিতে হবে।

const handleSubmit = (e) => {

e.preventDefault ();

setBooks((prevState) => {

const newBook = {id: newDate().getTime().toString(), name: bookName};

return [...prevState, newBook];

});

setIsModalOpen(true);

setModalText(“book is added”)

}:

পূর্বের value গুলো ‍spread operator এর মধ্যে থাকবে এবং নতুন কোন value আসতে ‍singe package হয়ে যুক্ত হবে।

প্রশ্নঃ কোন value গুলো spread operator এ থাকবে?

উত্তরঃ যেহেতু setBooks এ মধ্যে ব্যবহার করেছে যা books এর অংশ তাই books এর Data গুলোই থাকবে।

প্রশ্নঃ prevState ছাড়া অন্য কোন কিছু ব্যবহার করলে কি কাজ করতো?

উত্তরঃ অবশ্যই ।

এখানে প্রশ্ন হলোঃ ‍setBooks এ একাধিক লাইন ব্যবহার করা হয়েছে কিন্তু এখানে দ্বিতীয় লাইন return করা হয় নাই কেন?

উত্তর: এটা শুধু একটা ভেরিয়েভল।

প্রশ্নঃ ফাংশন এর ভেতরে ভেরিয়েভল ডিক্লেয়ার করলে তা কে কে ব্যবহার করতে পারবে?

উত্তরঃ শুধু ফাংশন ই তাকে ব্যবহার করতে পারবে। অন্য কোথায় ব্যবহার করতে চাইলে সম্ভবত ভেরিয়েভল কে return করে দিলেই হবে।

js এ ভেরিয়েভল কে return করলে তা ফাংশনে চলে যেতে। কিভাবে ?

useReducer Hooks

import form react

const [booksState, dispatch] = useReducer(reducer, {

booksরিলেটেড state নিয়ে কাজ করবে এবং action কে dispatch করবে।

তিনটি state কে একটা ‍state এর মধ্যে নিয়ে আসা হয়েছে।

books: booksData,

isModalOpen: false,

modalText: “ “

})

useReducer একটি reducer ফাংশন তৈরি করে এবং ‍state গুলোর inisital value নেয়।

const reducer = (state, action) => {

বর্তমানে initial যেই state গুলো আছে তা reducer এর এই state এ চলে আসছে ।

return state;

}

অবর্শই কোন না কোন কিছু রিটান করতে হবে। যেমনঃ ‍state.

রেডিউছার ফাংশন দুইটি প্যারামিটার গ্রহন করবে present state and action.

যখন কোন action dispatch হবে তখন reducer তাকে নিয়ে কাজ করবে।

<button

              onClick={() => {

                removeBook(id);

              }}

            >

              Remove

            </button>

অ্যানোনিমাস ফাংশন হিসাবে এর মধ্যে একটা ফাংশনকে call করা হয়েছে এবং এতে একটা id parameter হিসাবে pass করা হয়েছে।

   const filteredBooks = [...state.books].filter(

        (book) => book.id !== action.payload

      );

সেগুলোই থাকবে যাদের id ম্যাচ করবে না।

if (action.type === "ADD") {

      const allBooks = [...state.books, action.payload];

      return {

        ...state,

        books: allBooks,

        isModalOpen: true,

        modalText: "book is added",

      };

    }

প্রতিবার state কে রিটান করে দিতে হবে।

Prop drilling

com1--> com2--> com3 --> com4

এখানে com2 && com3 তে কোন ধরকার নাই com4 এ দরকার।

props এর মাধ্যমে কোন টেক্স পাঠালে propsName= “”

object পাঠালে propsName= { }

PropsName হলো যেই নামে পাঠাতে চাচ্ছি।

useContext Hook

parente এ useContext import করে ঃ import { UserContext } from "./UserContext";

তৈরি করা ঃ export const UserContext = React.createContext();

তৈরি করে UserContext ভেরিয়েভলে রেখে দিলাম।

|  |
| --- |
| step1: create context |
| // step2: from parent wrap childs with context provider |
| // setp3: state access useContext hook |

<UserContext.Provider value={{ user, text }}>

<Component2 />

</UserContext.Provider>

parente থেকে provide করলে সকল child ব্যবহার করতে পারবে।

value = { } object pass করতে হলে {এর মধ্যে অবজেক্ট গুলো দিতে হবে একাধিক হলে , ব্যবহার করতে হবে}

যেহেতু object তাই

|  |
| --- |
|  |
| import React, { useContext } from "react"; | |
| import { UserContext } from "./UserContext"; | |
| const Component4 = () => { | |
| const { user, text } = useContext(UserContext); | |
| return ( | |
| <div> | |
| <h2>{text}</h2> | |
| <h3>{user.id}</h3> | |
| <h3>{user.name}</h3> | |
| </div> | |
| ); | |
| }; | |
| export default Component4; | |

useContext কে এবং যে UserContext কে import করতে হবে।

PropType and defaultTypes

Users component

User component

userName state

userId state

User.js

import PropTypes from "prop-types";

User.propTypes ={

কমপোনেন্ট এর টাইপ বলে দিতে পারবো।

key-vlaue

userName: PropType.string,

userId: PropType.number,

}

যদি কোন টাইপ না আসে তাহলে ডিফল্ট টাইপ হিসাবে

|  |
| --- |
|  |
| // User.defaultProps = { | |
| // userName: "default name", | |
| // userId: 0, | |
| // }; | |

User.propTypes= {

user: PropTypes.object.isRequired, isRequired ব্যবহার করলে কোন এটাতে object কে এখন টাইপ হিসাবে দিতে হবে। তবে ডিফল্ট ব্যবহার করলে তখন আর error আসবে না।

};

object এর জন্য

|  |
| --- |
|  |
| User.propTypes = { | |
| // key-value | |
| user: PropTypes.shape({ | |
| id: PropTypes.number, | |
| name: PropTypes.string, | |
| }), | |
| }; | |

অবজেক্ট এর প্রোপারট্রির type সেট করার জন্য ‍shape ব্যবহার করা হয়।

rafcp ----------> propeType সহ structure.

react Routing

react router আমরা কেন ব্যবহার করি ?

কারণ route এর জন্য যা যা সুবিদা প্রয়োজন হয় আমরা তা পাই না তাই ।

reactrouter.com

npm install react-router-dom@6

*App.js*

import {BrowserRouter, Routes, Route} from “react-router-dom”;

এ return এর ভেতরে

<BrowserRouter>

<Routes>

<Route path=”/” element={}/> element এর মধ্যে <Home />

<Route path=”/blogs” element={ <Blogs />}/>

<Route path=”\*” element={ <Error />}/> \* ব্যবহার করার ফলে অন্য যেকোন path দিলে তাকে Error পেজে নিয়ে যাবে।

</Routes>

</BrowserRouter>

এখানে একটা বিষয় উল্লেখ্যঃ এই সময় react কে import করা হয় নাই।

react routing | Navigation and redirect

Routes এর উপরে nav তৈরি করতে হয়।

<a href=”/”>Home</a> এটা দ্বারা কাজ করলে প্রতিটি কাজের জন্য পুরো পেইজ কে লোড করবে।

import {Link} from “react-router-dom”;

<Link to=”/”> Home </Link> । শর্ত প্রথমে তো একটা Route তৈরি করা থাকতে হবে। Route এ যেই path দিবো ঠিক একই path এখানে দিতে হবে।

App.css থেকে App.js এর সকল child কে css style করা যায়।

<Navlink to=”/”> Home </NavLink> Route এ যেই path দিবো ঠিক একই পাঠ এখানে দিতে হবে।

Navlink ব্যবহারের ফলে যেই object টাকে ক্লিক করা হয়েছে তার সাথে .active নামে একটা ক্লাস যুক্ত করে দেয়।

nav .active{

color: red;

}

আমার চাচ্ছি contact page এর নিচে থাকা Go to home page বাটনে কেউ ক্লিক করলে তাকে Home page এ নিয়ে যাবে। এই জন্য react-router-dom থেকে useNavigate Hooks ব্যবহার করা হয়।

react-router-dom থেকে কোন কিছু import করলে তাকে { } এর মধ্যে লিখতে হয়।

const navigate =useNavigate(); contact page এর মধ্যে ব্যবহার করার জন্য তাকে ভেরিয়েভলে রেখে দিলাম।

<button onClick={() => {

navigate(“/”);

}}>

Go to Home Page

</button>

react routing | dynamic routing using useParams

src folder এ data.js file তৈরি করে তাকে pages folder এ কোন একটি ফাইল থেকে import করার নিয়ম।

data.js কে এই ভাবে import করলো কেন?

import {blogsData } from “../data”;

const [blogs, setBlogs] = useState(blogsData);

<Link to = {title}> Learn More </Link> ব্যবহারের ফলে ঃ blogs/title value| blogs/html | blogs/css | blogs/js একে handle করার জন্য একটা Route তৈরি কেরতে হবে ‍route App.js এ তৈরি করবো। {app.js এর টাইটেলে চলে যাবে}

বা বর্তমান path এর সাথে /html বা /css এভাবে যুক্ত হবে।

localhost:3000/blogs/html

const truncateString =(str, num) => {

}

<p> {truncateString(body, 100)}</p>

blogs/html ->Blog(html)

blogs/css->Blog(css)

blogs/react ->Blog(react)

এই রকম হবে।

<Route path = “/blogs/:title” element={<Blog />} />

localhost:3000/blogs/title এর value

/blogs/html বা /blogs/css বা /blogs/react এগুলো :title এর মধ্যে থাকবে। :name দিলে :name এর মধ্যে থাকবে বা যাবে। অর্থাৎ যা দিবো তার মধ্যেই থাকবে।

Blog পেইজে এসে useParams কে react-router-dom থেকে import করবো।

তাকে খুজার জন্য const params = useParams(); যে কোন নামে রাখতে পারি। তবে blogs/:title নামে পাঠিয়েছিলাম বলে {title } দিবো।

usePararmsআমরা কেন ব্যবহার করতেছি?

**App.js থেকে blogs/:title এর value পাওয়ার জন্য useParams ব্যবহার করতেছি।**

আমার মনে হচ্ছে params এ ব্যবহার করা title কে ; Learn More এ ক্লিক করলে এর মধ্যে শুধু html বা css আসবে। **কিন্তু {title } এভাবে লিখলে এর মধ্যে /html বা /css যাবে**

**তাহলে useParams এর মাধ্যেমে Learn More যেইটাতে ক্লিক করা হচ্ছে তার টাইটেল পাচ্ছি। পাচ্ছি App.js এর পাথ থেকে blogs/:title**

useLocation Hook

useLocation ব্যবহার করে state এর সাথে Data পাস করা যায়।

Blog Component এ useLocation কে import করবো।

যেই জায়গাই আমরা ‍state গুলোকে access করতে যাচ্ছি যেই জায়গাই আমরা useLocation কে import করবো।

const location =useLocation();

এটাকে clg করলে hash, key, pathname, search, state এগুলো পাওয়া যায়।

Blogs component এ Link এ ‍state ব্যবহার করবো।

এখানে উল্লেখ্য যে map এর ফলে যেহেতু ‍single packege Data আসে। তাই এই এই Artical এ যেই id, title, body আসছে সেগুলো state এর মাধ্যমে useLocation এ দেওয়া হলো।

<Link to = {title} state={{id, title, body}} > learn More </Link>

Blog Component এ <h1> {location.state.body} </p>

route parameter, query parameter

useParams এর মাধ্যমে Route এর path /:...... এর value খুজেঁ নিতে পারি। ‍App.js থেকে। তবে এখান ভেলু টা location bar এ দেওয়া হয়েছে।

query Parameter হচ্ছেঃ localhost:3000/user?id=101

যে hooks প্রয়োজন তা হলোঃ useSearchParams এবং একে ব্যবহার করার জন্য একটা স্টেট নিতে হবে।

এটা ব্যবহারের জন্য path এ **/user/:userid দেওয়ার প্রয়োজন নাই শুধু /user দিতে হবে বাকিটা query params করে নিবে।**

{searchParams.get(“id”)} যে key দিয়েছি তাই লোকেশন বারে দিতে হবে। এই value গুলো আমার User.js এ প্রিন্ট করবো।

এতক্ষণ আমারা location bar এর মাধ্যমে সেট করে তা print করলাম। কিন্তু এখন আমরা input এর মাধ্যমে value গুলো location bar এ সেট করবো।

input tag এর মাধ্যমে

<input type =”text” value={name} onChange=((e) => বাকি... />

const [name, setName] =useState(“”)

useSearchParams ব্যবহার করার জন্য একটি state প্রয়োজন তাইঃ

setSearchParams({name: name, age: age}}; key: value value input থেকে আসবে।

http method | how to get data from an API.

api থেকে data fatch করার পূর্বে আমরা চেক্ করে নিবো যে ‍api এ কোন data আছে কি নাই।

by default fetch api এ কোন method উল্লেখ না করলে তা get methode.

fetch data

fetch কে বলবো তোর ‍res কে json formate এ আমার সামনে পেশ কর। এই res এর মধ্যে থাকা data গুলোকে setUsers এর মধ্যে পাঠিয়ে দে।

fetch(“url”)

.then(res) => {

if(!res.ok){

throw Error(“could not fetch”);

}

return res.json()

})

.then(data) => {

setUsers(data.users);

})

.catch((err) => {

setError(err.message);

})

.finally(()=> {

setIsLoading(false);

});

const getAllusers = () => {

এখানে fetch টাকে দিয়ে দাও।

}  
useEffect(() => {

getAllUsers();

}, []}

user কে delete করার জন্য আমাদের শুধু মাত্র একটি id প্রয়োজন।

ডিলিট হয়ে গেলে আবার ও ডেটা টাকে fetch করতে হবে।

যাকে ডিলিট করতে চাচ্ছি তার বাটনের সাথে onClick Event এ handleDelete নামে একটি ম্যাথোড এ id পাস করে দিবো। এবং মেথোড টা তৈরি করে নিবো এবং paramieter হিসাবে id receive করবো।

প্রশ্নঃ মেথড কি?

উত্তরঃ মেথড হলো একটি ফাংশন যা Event এর মধ্যে ব্যবহার করা হচ্ছে তাই একে method বলা হচ্ছে।

ডিলিট করার জন্য fatch এর সাহায্য নিতে হবে।

method হিসাবে শুধু delete দিতে হবে।

fetch(URL + `/${id}`, {

method: “DELETE”,

})

.then((res) => {

if(!res.ok) {

throw Error(“could not delete”);

}

getAllusers(); অর্থাৎ যদি ডাটা ডিলিট হয়ে যায় তাহলে আবার fetch এর কাজ চলবে।

});

এখানে dataset করার কোন প্রয়োজন নেই।

**How to create data in an API**

UserForm.js এ rafcp দিয়ে PropType সহ snippets নিয়ে নিবো।

একটি form তৈরি করতে হবে userName and Email দিয়ে এবং এগুলোকে রাখার জন্য একটি user নামের state নিবো।

input এ value দিবো username and onChange এ handleChange method নিয়ে নিবো।

কোন ফিল্ডের value পরিবর্তন করিতেছি তার নাম ও ভ্যালু পাওয়ার জন্য handleChange এঃ

const selectedField = e.target.name;

cosnt selectedValue = e. target.value;

setUser(prevState => {

return {...prevState, [selectedField]}

}

handleChange এ কোন value গেলে তা আমরা user এ আপডেট করে রাখবো।

  const handleChange = (e) => {

    const selectedField = e.target.name;

    const selectedValue = e.target.value;

    setUser((prevState) => {

      return { ...prevState, [selectedField]: [selectedValue] };

    });

  };

আমরা যেই e ব্যবহার করি তার মানে হলো event অর্থাৎ ঘটনা

প্রশ্ন ঃ user কখন তৈরি হইতেছে?

উত্তরঃ যখন input এ কোন value দিতেছি তখন ই user তৈরি হইতেছে।

প্রশ্নঃ তার মানে প্রতিটি letter লিখার জন্য কি একটি করে user তৈরি হইতেছে।

উত্তরঃ হ্যা। তবে setUser ব্যবহার করার ফলে name এবং value ব্যবহার করার জন্য কাজটি পরিকল্পিত ভাবে হয়েছে।

<UserForm btnText="Add User" handleSubmitData={addUser} />

chile to parent data pass

fetch(URL, {

      method: "POST",

      headers: {

        "Content-Type": "application/json",

      },

      body: JSON.stringify(user),

    })

      .then((res) => {

        if (res.status === 201) {

          getAllUsers();

        } else {

          throw new Error("could not creat new user");

        }

      })

      .catch((err) => {

        setError(error.message);

      });

হেডার ও বড়ি ব্যবহার করতে হবে।

how to deploy react app on netilfy from github

যাতে সবার সাথে শেয়ার করতে পারবো।

react memo

by default component একবার rendering হবে।

q: component render করবে কখন ?

উত্তরঃ যখন কোন state change হয় অথবা props আসে।

message নামে একটি component কে app.js িএ import করলাম। কিন্তু ‍button এ ক্লিক করলে message কে render করবে। যা একদম অপ্রয়োজনীয়।

ছোট ui এর জন্য এটা কোন বিষয় ই না। তবে যখন বড় কোন প্রজেক্ট নিয়ে কাজ করা হবে তখন এটা অনেক বড় একটি সমস্যা হতে পারে।

rerender close

function এর বড়ি memo function এর মধ্যে দিয়ে দিবো। তবে এর চেয়ে বেস্ট উপায় হলোঃ

export default memo(Message); Message Component কে memo ফাংশন এর মধ্যে export করে দেওয়া।

props এর value চেন্স হলে rerender হওয়া দরকার এবং তা render হবে।

<https://dmitripavlutin.com/use-react-memo-wisely/>

**useCallback hook**

**memo ব্যবহারের ফলে শুধু props এর value চেন্স হলেই কেবল message component কে render করবে। তবে child থেকে parent এর data কে props এর মাধ্যমে updata করার করলে message কে render করবে। আবার props এর মাধ্যমে কোন update না করলেও message কে render করবে।**

**এক্ষেত্রে Parent থেকে onযুক্ত method pass করা ভাল । এতে child এ যখন onযুক্ত method রিসিভ করবো তখন এটা বুঝা যাবে যে এই method এর মাধ্যমে parent এ props পাঠানো হবে।**

const Message = ({ numberOfMessages, onHandleIncrement }) => {

**on ছাড়া method ঃ parent থেকে props এর মাধ্যমে data আসতেছে।**

**on যুক্ত method ঃ parent এ data কে pass করে দিতেছি।**

**এখন দেখা যাচ্ছে toggle কে আপডেট করলে message render হচ্ছে । কিন্তু কেন?**

**toggle এর সাথে তো message এর কোন সম্পর্ক নাই। তাও কেন?**

**app.js toggle এর value update করলে app.js component টা render হবে।**

<Message

        numberOfMessages={count}

        onHandleIncrement={handleIncrementMessage}

      />

onHandleIncrement={handleIncrementMessage} এটা ডিফাইন করা হয়ে যাইতেছে। এবং handleIncrementMessage ম্যাসেজ কল হয়ে যাইতেছি । যদিও count এর value এর কোন পরিবর্তন হইতেছে না।

তাই আমরা চাচ্ছি count state এর কোন পরিবর্তন না হলে onHandleIncrement যেন কল করা না হয়।

useCallback যা করে প্রথমে এটা inline funciton বা callback function recive করতে পারে। এবং ডিপেনডেন্সি হিসাবে কোন কিছু state সেট করে দিতে হবে।

  const handleIncrementMessage = useCallback(() => {

    setCount(count + 1);

  }, [count]);

**child থেকে data parent এ পাঠালে তখনঃ যখন ই parent render হবে react child কে render করবে যে কোন data আসবে কিনা। তখন যে props এর method এর মাধ্যমে data আসবে তাকে কল করবে।**

**এই কলটি যেন কোন প্রয়োজনীয় state এর value মাধ্যমে হয় তাই useCallback ব্যবহার করা হয়।**

**UseMemo hook**

**কোন বড় একটা claculation কে memory তে রেখে দেওয়া। যাতে বার বার সেটাতে calculation না করে।**

  const calculatedNumber = useMemo(() => {

    let number = 0;

    for (let index = 0; index < 50000; index++) {

      number++;

    }

    return number;

  }, []);

**ব্যবহারের নিয়ম: useMemo এর মধ্যে একটা অ্যানোনিমাস ফাংশন ও ডিপেনডেন্সি নিতে হবে। অ্যানোনিমাস ফাংশন এর body এর মধ্যে for loop নিবো এবং শেষে রিটান ভেরিয়েভলকে রিটান করে দিবো যাতে তার ভ্যালু useMemo তে চলে যায়। এবং useMemo কে একটি ভেরিয়েভলে রেখে দিবো। এখন এই ভেরিয়েভলকে ব্যবহার করলে প্রতিবার আর ক্যালকুলেশন করার প্রয়োজন হবে না।**

# React Bangla Tutorial 15 : react component under the hood

একই ফাইলে Component তৈরি করলে তাকে inport করার প্রয়োজন নাই।

Component কি?

=যে কোন ফাংশন। (শর্ত ফাংশন এর মধ্যে কোন কিছুকে Return করতে হবে।)

বি: দ্র: ফাংশনাল component এর নাম এর প্রথম অক্ষর সব সময় বড় হাতের লিখতে হয়।

import React from 'react'

function Welcome () {

  return <h1> welcome</h1>

}

const App = () => {

  return (

    <div>

      <Welcome/>

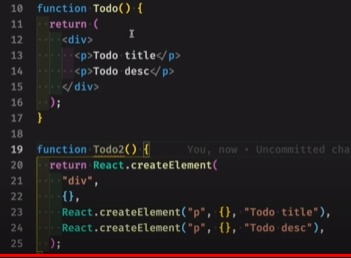
    </div>

  )

}

export default App

react এ কোন Empt tag কে Close করতে হবে।



এখানে Todo সাধারণ ভাবে তৈরি। Todo2 তৈরি করা হয়েছে React এর অভ্যন্তরীণ সিস্টেমে।

Class component State & SetState

|  |
| --- |
|  |
| import React, { Component } from 'react' |
| import './style.css' |
| class STATE\_IN\_CLASS extends Component { |
|  |
| constructor(props) { |
| super(props) |
| this.state = { |
| count : 0 |
| } |
| } |
|  |
| handleIncrement = () => { |
| this.setState({ |
| count : this.state.count + 1 |
| }); |
| } |
|  |
| handleDecrement = () => { |
| this.setState({ |
| count : this.state.count - 1 |
| }); |
| } |
|  |
| render() { |
|  |
| const {count} = this.state |
|  |
| return ( |
| <div> |
| <h1>Count : {count}</h1> |
| <button onClick={this.handleIncrement} disabled={count===5 ? true : false}>+</button> |
| <button onClick={this.handleDecrement} disabled={count===0 ? true : false}>-</button> |
| {/\* <button onClick={this.handleDecrement} disabled={count===0 && true}>-</button> \*/} |
|  |
| </div> |
| ) |
| } |
| } |
|  |
| export default STATE\_IN\_CLASS |

value সাথে সাথে console এ প্রিন্ট করতে চাইলে ।

এখানে setStae এর দ্বিতীয় প্যারামিটার এর মধ্যে একটি ফাংশন তৈরি করে তার মধ্যে console.log(………) রাখতে হবে।

তানাহলে console এর value এক ক্যারেকটার পরে যাবে।

 handleOnChange = (e) => {

        // console.log(e.target.value)

        this.setState({

            chageValue : e.target.value

        }, () => {

            console.log(this.state.chageValue)

        })

# React Bangla Tutorial 21 : Event Handler | onChange | onClick (Class component)

import React, { Component } from 'react'

export default class EVENT\_HANDLER\_CLASS extends Component {

    constructor(props) {

      super(props)

      this.state = {

         chageValue : ''

      }

    }

    handleOnChange = (e) => {

        // console.log(e.target.value)

        this.setState({

            chageValue : e.target.value

        })

        console.log(this.state.chageValue)

    }

  render() {

    return (

      <div>

        <input type="text" onChange={this.handleOnChange} />

        <p> {this.state.chageValue}</p>

      </div>

    )

  }

}

|  |  |
| --- | --- |
|  | import React from "react"; |
|  |  |
|  | import Button from "react-bootstrap/Button"; |
|  | import Card from "react-bootstrap/Card"; |
|  |  |
|  | const ReactBootstrap = () => { |
|  | return ( |
|  | <Card style={{ width: "20rem" }}> |
|  | <Card.Body> |
|  | <Card.Title>Card Title</Card.Title> |
|  | <Card.Text> |
|  | Lorem ipsum dolor sit, amet consectetur adipisicing elit. Aspernatur |
|  | quibusdam non iure accusantium nemo! Aliquam. |
|  | </Card.Text> |
|  | <Button>Learn more</Button> |
|  | </Card.Body> |
|  | </Card> |
|  | ); |
|  | }; |
|  |  |
|  | export default ReactBootstrap; |

\*\*\*\*\*\*\*\*\*\*\*\*React Bangla Tutorial 22 : Event handler binding**\*\*\*\*\*\*\*\*\*\*\*\*\*\***\*\*\*\*\*\*\*\*\*\*

import React, { Component } from 'react'

export default class Index extends Component {

    constructor(props) {

      super(props)

      this.state = {

         count : 0

      }

    }

    handleClick = () => {

        this.setState({

            count : this.state.count + 1

        })

    }

  render() {

    return (

      <div>

        <h1> { this.state.count}</h1>

        <button onClick={this.handleClick}> Increase</button>

      </div>

    )

  }

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*From (useState & object)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

import React, { useState } from "react";

import style from "./form.module.css";

const Form1 = () => {

  const [user, setUser] = useState({ name: "", email: "", password: "" });

  const { name, email, password } = user;

  const handleChange = (e) => {

    setUser({ ...user, [e.target.name]: e.target.value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    console.log(user);

    setUser({ name: "", email: "", password: "" });

  };

  return (

    <div className={style.formCm}>

      <h1 className={style.formCmHd}> Register</h1>

      <form action="" onSubmit={handleSubmit} className={style.formCmForm}>

        <div className={style.formCmFormMainDiv}>

          <div className={style.formCmFormMainDivName}>

            <label htmlFor="name">Name :</label>

            <input

              type="text"

              name="name"

              value={name}

              onChange={handleChange}

              className=""

            />

          </div>

          <div className={style.formCmFormMainDivEmail}>

            <label htmlFor="email">Email :</label>

            <input

              type="email"

              name="email"

              value={email}

              onChange={handleChange}

              className=""

            />

          </div>

          <div className={style.formCmFormMainDivPassword}>

            <label htmlFor="password">Password :</label>

            <input

              type="password"

              name="password"

              value={password}

              onChange={handleChange}

              className=""

            />

          </div>

          <div>

            <button> Register</button>

          </div>

        </div>

      </form>

    </div>

  );

};

export default Form1;

### Props and destructuring: video 8

import React from "react";

import Child from "./components/Satel\_Lifting/Child";

const App = () => {

  const data = "I am Data from app.js";

  return (

    <div>

      <Child data={data} />

    </div>

  );

};

export default App;

import React from "react";

const Child = (props) => {

  const { data } = props;

  return <div>{data}</div>;

};

export default Child;

**passing data from child to parent component: video 27**

import React from "react";

import Child from "./components/Satel\_Lifting/Child";

const App = () => {

  const data = "I am Data from app.js";

  const handleChildData = (childData) => {

    console.log({ childData });

  };

  return (

    <div>

      <Child data={data} onChildDate={handleChildData} />

    </div>

  );

};

export default App;

import React from "react";

const Child = (props) => {

  const { data } = props;

  const data1 = "I am data from child component";

  return (

    <div>

      <p>{data}</p>

    </div>

  );

};

export default Child;

**redux tutorial**

state কে মেইনটেইন করার জন্য আমরা redux শিখতেছি।

state, actions

শুরুতে npm init -y এতে packeg.json file আসবে।

code লিখার জন্য index.js file নিবো।

npm install redux । এখন আমাদের App টি redux এর উপর নির্ভরশীল।

প্রথমে state কে define করে দিতে হবে।

action এর উপর ভিত্তি করে state কে update করতে হবে।

const initialCounterState = {

  count: 0,

};

কি করতে চাচ্ছি সেটাই হচ্ছে action হচ্ছে একটি object এর মধ্যে দুইটি জিনিস থাকে।action –Object –type, payload

action গুলোকে একটা ফাংশনে রেখে দিতে হবে।

const incrementCounter = () => {

  return {

    type: INCREMENT,

  };

};

1. state
2. dispath action মনে করো বাটন ক্লিক করা।
3. reducer work base on action
4. store

reducer এর কাজ হলো ‍action এর জন্য logic লিখা।

output দেখার জন্যঃ node index.js

createStore : এটা state কে হোল্ড করে রাখে। এর মধ্যে reducer কে রাখতে হবে।

reducer : এটা সবকিছুকে controll করে।

dispatch: বাটন ক্লিক করার মতো কাজ করে। এর মধ্যে action type বা যে ফাংশনে রাখছি তা বলে দিতে হবে।ফাংশন()।

subscribe : Store কে view করে। এর মধ্যে একটি অ্যানোনিমাস ফাংশন নিতে হবে।

getState: state এর অবস্থা দেখা যায়।

const store = createStore(counterReducer);

store.subscribe(() => {

  console.log(store.getState());

});

store.dispatch(incrementCounterAction());

payload এর ব্যবহার

payload কে একশন টাইপ এর জন্য যেই ফাংশন নিতেছি সেই ফাংশনের ফাংশনে প্যারামিটার হিসাবে নিতে হবে।

store.dispatch(incrementCounterByValue(5));

const incrementCounterByValue = (value) => {

  return {

    type: INCREMENT\_BY\_VALUE,

    payload: value,

  };

};

   case INCREMENT\_BY\_VALUE:

      return {

        ...state,

        count: state.count + action.payload,

      };

store.dispatch(addUser("rafiq"));

const ADD\_USER = "ADD\_USER";

const initialState = {

  users: ["anis"],

  count: 0,

};

const addUser = (user) => {

  return {

    type: ADD\_USER,

    payload: user,

  };

};

case ADD\_USER:

      return {

        users: [...state.users, action.payload],

        count: state.count + 1,

      };

action গুলোকে ভেরিয়েবলে রেখে দিবো যাতে করে বানানে কোন ভুল না হয়।

case এর মধ্যে একশন নেম দিতে হবে। একশন যেই ফাংশনে আছে তার নাম দিলে কি হবে। উত্তরঃ না।

store এ একটি মাত্র reducer রাখা যায়। তবে reducer রাখার জন্য ভিন্ন প্রদ্ধতি অবলম্বন করতে হয়।

combineReducers রেডাক্স এর একটি ফাংশন এটি object গ্রহন করে। একে rootReducer নামের একটি ভেরিয়েভলে রেখে দিবো।

key : reducertName

এখন createStore এর মধ্যে rootReducer কে রেখে দিবো।

middleware এর জন্য আমরা redux logger কে npm থেকে install করবো।

npm i --save redux-logger

যখনই আমরার redux এ কোন middleware ব্যবহার করবো তখন redux থেকে ‍applyMiddleWare import করে নিবো।

const store = createStore(productReducer, applyMiddleware(logger));

createStore এর first paramite হিসাবে reducer name এবং second parameter হিসাবে ‍applyMiddleware ফাংশন এর মধ্যে middleware name ।

const { default: logger } = require("redux-logger");

logger middleware এর মাধ্যমে time, prevState, action , next State এর ধারণা পাওয়া যায়।

**fetch data using redux-thunk**

initialState নিবো todos, error, isLoading

action-> GET\_TODOS\_REQUEST, GET\_TODOS\_FAIL, GET\_TODOS\_SUCCESS

GET\_TODOS\_FAIL এর জন্য যেই ফাংশন তৈরি করবো তাতে প্যারামিটার হিসাবে error দিবে। এবং payload ব্যবহার করবো।

GET\_TODOS\_SUCCESS এর জন্য যেই ফাংশন তৈরি করবো তাতে প্যারামিটার হিসাবে todos দিবে। এবং payload ব্যবহার করবো।

reducer এর ‍swich এর case এ action type এর নাম দিয়ে দিবো।

reducer মূলত action এর জন্য Logic লিখবে।

reducer এ payload আসার জন্য action type এর জন্য যেই ফাংশন লিখবো তো payload কে নিতে হবে।

reducer এ todos এর মধ্যে যেই data গুলো আসবে তা action.payload থেকে আসবে।

store.dispatch(fetchData());

async function create করার জন্য

প্রথমে data কে fetch করতে হবে then আমাদের চেক করতে হবে। সেটা success বা failer কিনা অন্যন্য action গুলোকে ব্যবহার করতে হবে। চেক করার জন্য তো প্রথমে data কে fetch করতে হবে। তাই fetchdata নামে একটি ফাংশন তৈরি করে নিতে হবে। এই fetchData ফাংশন কি করবে । এই request api থেকে request করবে চেক করবে success বা failer এবং action গুলো সে genarete করবে।

const fetchData = () => {};

normale ‍একশন গুলো object return করে but এবার আমরা একটি function return করতে যাচ্ছি। যেই ফাংশন api call থেকে শুরু করে, ডিসপ্যচ এর কাজ করবে যদি data successfully পায় আমরা সেক্ষেতে todosSuccess কে dispatch করবে , যদি আমাদের successfull না হয় সেক্ষেতে todosfail কে dispatch করবে action কে । আর যদি এখানে আমরা যখন রিকুয়েস্ট করা শুরু করবে তখন এইখান থেকে dispatch করবে GET\_TODOS\_REQUEST কে। সেই জন্য আমাদের রিডাক্স থাংক যে মিডেলওয়ার আছে সেটা লাগবে সেটা আমাদের সাহায্য করবে এই ASYNC FUNCTION create করতে এবং এই fetching এর কাজ সবকিছু এখানে async ভাবে করতে।

redux-thunk থেকে আমরা dispatch এবং getState Methode কে access করতে পারবো।

[middleWare কিভাবে ব্যবহার করতে হয়?](#middleware_use_rules)

const thunk = require("redux-thunk").default;

middleWare এর নাম হলো thunk

যেহেতু আমরা রিডাক্স থাংক ব্যবহার করছি তাই এটা আমাদের একটা ফাংশন রিটান করতে সাহায্য করবে। তো এখানে আমরা একটা ফাংশন রিটান করবো এবং এই ফাংশন থেকে আমরা আমাদের একশন কে dispatch করতে পারবো। অথ্যৎ success হচ্ছে কিনা failer হচ্ছে কিনা এগুলো আমরা dispatch করতে পারবো। সবচাইতে importent কথা আমাদের async call টা ‍axios এ সেটা আমরা এখানে make করতে পারবো। তো এখানে dispatch যেটা রয়েছে সেটা আমরা অবশ্যই পাস করবো কারণ এই dispatch কি করবে আমাদের একশন কে এইখান থেকে আমরা যেই একশন তৈরি করেছি উপরের দিকে নিশ্চয় খেয়াল করেছেন।

GET\_TODOS\_REQUEST, GET\_TODOS\_FAIL, GET\_TODOS\_SUCCESS এই গুলোকে কল করতে সাহায্য করবে এই একশন গুলোকে dispatch করতে পারবো। তো একদম শুরুর দিতে আমরা GET\_TODOS\_REQUEST কে dispatch করবো। api কে call করার জন্য আমাদের axios কে install করে নিতে হবে। এবার আমরা axios কে ব্যবহার করে api কে কল দিতে পারি। তো যেই api রয়েছে সেটাকে একটা ভেরিয়েবল এর মধ্যে রেখে দিচ্ছি।

const fetchData = () => { thunk ব্যবহার করার কারনে এটা একটি ফাংশন রিটান করবে।

  return (dispatch) => { এর সাহায্যে উপরের অ্যাকশনকে dispatch করতে পারবো।

    dispatch(getTodosRequest());

    axios

      .get(API\_URL)

      .then((res) => {

        const todos = res.data;

        const title = todos.map((todo) => todo.title);

        dispatch(getTodosSuccess(title));

      })

      .catch((error) => {

        const errorMessage = error.message;

        dispatch(getTodosFailed(errorMessage));

      });

  };

};

react-redux | complete counter react app

স্যার ফাইল এর নাম ছোট হাতের অক্ষর দিতেছে তাতে কোন সমস্যা হইতেছে না। কিন্তু আমরা শুধু react এ fileName ছোট হাতের অক্ষর দিয়ে শুরু করলে কোড করতো না কেন?

default:

      return state;

এটা মনে রাখতে হবে।

store এর জন্য redux কে install করলাম।

redux কে react এ ব্যবহার করার জন্য react-redux এই প্যাকেজ টা install করতে হবে।

সবচাইতে root file এ আমরা store কে ব্যবহার করবো। যেমন: index.js

useSelector এর কাজ হলো state কে এইখানে নিয়ে আসা।

useDispatch এর কাজ হলোঃ এখানে action কে dispatch করা।

const initailCounter = { count: 0 };

এই স্টেট টাকে counter.js component এ নেওয়ার জন্য useSelector ব্যবহার করা হয়।

const count = useSelector((state) => state.count);

 return (

    <div>

      <h3>Count :{count}</h3>

        );

how to fetch data | react-redux

npm install redux react-redux axios redux-thunk

css

onclick= window.open(“url”) ব্যবহার করলে ক্লিক কররে উক্ত url এ নিয়ে যাবে।

class = “btn request-callback”

 div {

        height: 300px;

        width: 300px;

        background-color: red;

        border: 5px solid black;

        margin: 0 auto;

        margin-top: 200px;

        transition: 3s;

      }

      div:hover {

        transform: rotate(45deg);

      }

transition মেইন class এর সাথে ব্যবহার করা হয়।

transform চাইলে hover এর সাথে ব্যবহার করা যায়।

**css Specificity**

**id>class>tag**

/\* specificity calculator \*/

/\* a=number of id  \*/

/\* b=number of class, pseduo classes, attributes \*/

/\* c=number of elements, pesude elemnts \*/

pesude elemnts এর মধ্যে ডাবল (:) হবে।

   img {

        height: 200px;

        width: 200px;

        display: block;

        margin: 0 auto;

        animation-name: ismail;

        animation-duration: 3s;

      }

 @keyframes ismail {

        from{

          height: 200px;

        }

        to{

          height: 200px;

        }

      }

**css3 variables**

:root {

--primary-color: brown;

--secondary-color: red;

}

#div1{

background-color: var(--primary-color);  
}

***BootStrap***

container

container-fluid

container-xl, md, xxl এগুলো ব্যবহার করলে শুধু এই break point এ break হবে। অন্যগুলোতে Full Screen হবে।

container

row

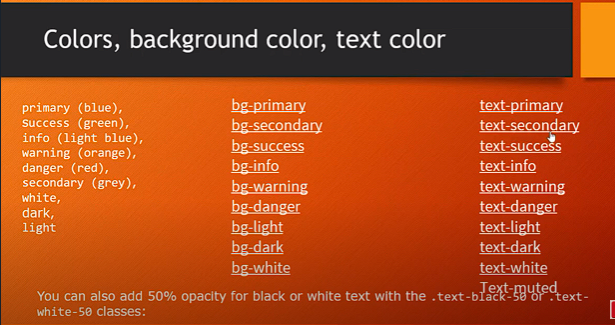
col-4 ( total 12 col is devidet this web. )

* Extra small (xs)
* Small (sm)
* Medium (md)
* Large (lg)
* Extra large (xl)
* Extra extra large (xxl)

col-lg-3

col-md-6

col-sm-12



p-0 p-1 p-2 p-3 p-4 p-5

p(t/b/l/r)-(0-5)

px- py-

m-0 m-1 m-2 m-3 m-4 m-5

m(t/b/l/r)-(0-5)

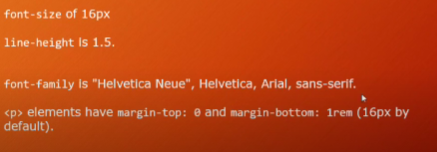
img-fluid img কে responsibe করার জন্য।

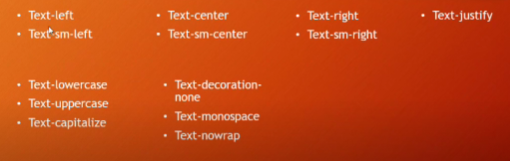
row এর মধ্যে text-center ব্যবহার করলে row এর ভেতরে যতগুলো div আছে তাদের সবার টেক্স center হবে।

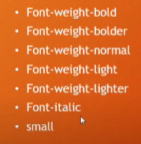
border img-fluid img-thumbnail rounded-circle

center এ নেওয়ার জন্য্ ।

font-awasame এর icon কে বড় করার জন্য fa-2x fa-3x







w-100 ব্রেক দেওয়ার জন্য

row-col-2

JavaScript Can Change HTML Content

document.getElementById("demo").innerHTML = "Hello JavaScript";

## JavaScript Can Change HTML Styles (CSS)

document.getElementById("demo").style.fontSize = "35px";

## JavaScript Can Hide HTML Elements

document.getElementById("demo").style.display = "none";

## JavaScript Can Show HTML Elements

document.getElementById("demo").style.display = "block";

## The <script> Tag

<script>  
document.getElementById("demo").innerHTML = "My First JavaScript";  
</script>

## External JavaScript

<script src="myScript.js"></script>

## JavaScript Display Possibilities

JavaScript can "display" data in different ways:

* Writing into an HTML element, using innerHTML.
* Writing into the HTML output using document.write().
* Writing into an alert box, using window.alert().
* Writing into the browser console, using console.log().

## Using document.write()

<script>  
document.write(5 + 6);  
</script>

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>My First Web Page</h1>  
<p>My first paragraph.</p>  
  
<button type="button" onclick="document.write(5 + 6)">Try it</button>  
  
</body>  
</html>

Using document.write() after an HTML document is loaded, will **delete all existing HTML**:

The document.write() method should only be used for testing.

<script>  
window.alert(5 + 6);  
</script>

<script>  
console.log(5 + 6);  
</script>

## JavaScript Print

<button onclick="window.print()">Print this page</button>

# JavaScript Statements

 let x, y, z;

      x = 5;

      y = 6;

      z = x + y;

      document.getElementById("demo").innerHTML =

        "The value of z is " + z + ".";

|  |  |
| --- | --- |
| ar | Declares a variable |
| let | Declares a block variable |
| const | Declares a block constant |
| if | Marks a block of statements to be executed on a condition |
| switch | Marks a block of statements to be executed in different cases |
| for | Marks a block of statements to be executed in a loop |
| function | Declares a function |
| return | Exits a function |
| try | Implements error handling to a block of statements |

## JavaScript Variables

JavaScript uses the keywords var, let and const to **declare** variables.

## JavaScript Operators

JavaScript uses **arithmetic operators** ( + - \* / ) to **compute** values:

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

"John" + " " + "Doe"

## JavaScript Comments

let x = 5;   // I will be executed  
  
// x = 6;   I will NOT be executed

A JavaScript name must begin with:

* A letter (A-Z or a-z)
* A dollar sign ($)
* Or an underscore (\_)

Numbers are not allowed as the first character in names.

**Hyphens:**

first-name, last-name, master-card, inter-city.

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

**Underscore:**

first\_name, last\_name, master\_card, inter\_city.

**Upper Camel Case (Pascal Case):**

FirstName, LastName, MasterCard, InterCity.

**Lower Camel Case:**

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

## Multi-line Comments

/\*  
The code below will change  
the heading with id = "myH"  
and the paragraph with id = "myP"  
in my web page:  
\*/

### 4 Ways to Declare a JavaScript Variable:

* Using var
* Using let
* Using const
* Using nothing

If you think the value of the variable can change, use let.

In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" operator.

The "equal to" operator is written like == in JavaScript.

short project tutorial link

1. Understanding State & Handling Events in React | Commlete React Course in Hindi #7(ummercase)