# React.js

* React was first designed by Jayden Walke, a software engineer at Facebook.
* It was first deployed for Facebook News Feed around 2011.
* In 2013, React was open sourced at JS conference.

## About React:—

(i) Component based approach:— A component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier.

(ii) It uses a Declarative Approach:—
Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow.

(iii) DOM updates are handled gracefully.

(iv) Reusable code

(v) React is designed for speed, speed of implementing the application simplicity and scalability. :(

## ☆ Create React App :—

(i) npm install -g create-react-app
(ii) create-react-app --version
(iii) create-react-app <projectname>

## ☆ index.js :—

```
const React = require('react');
const ReactDOM = require('react-dom');
ReactDOM.render(<h1>Hello World</h1>,
    document.getElementById('root')
);
```

⇒

```
import React from 'react';
import ReactDOM from 'react-dom';
```

① import react
② import react-dom
③ ReactDOM.render (क्या दिखाना है, कहा दिखाना है)

## ☆ JSX :— Javascript XML or Javascript extension

Javascript Syntax Extension

```
ReactDOM.render(<h1>Hello World</h1>,
    document.getElementById('root')
);
```

⇒

```
ReactDOM.render(
    React.createElement("h1",
        null,
        "Hello World"),
    document.getElementById('root')
);
```

## ☆ Without ReactDOM and JSX :—

```
var h1 = document.createElement("h1");
h1.innerHTML = "Hello World";
document.getElementById('root').appendChild(h1);
```

## ★ Render Multiple Elements inside ReactDOM in ReactDOM.render():-

(ii) By using one main tag or using div tag.

⇒
```
ReactDOM.render(
    <div>
      <h1> Hello World </h1>
      <p> Hello 1 </p>
      <h1> Hariom Patidar </h1>
    </div>,
    document.getElementById('root')
);
```

(ii) By using array of elements (seprated by comma (,))
For React V16 it is supported.

⇒
```
ReactDOM.render(
    [
      <h1> Hello </h1>,
      <h2> World </h2>,
      <p> Hariom </p>
    ],
    document.getElementById('root')
);
```

★ (iii) Using React.Fragment

```
ReactDOM.render(
    <React.Fragment>
      <h1> Hello </h1>
      <h1> World </h1>
    </React.Fragment>,
    document.getElementById('root')
);
```

OR

```
ReactDOM.render(
    <>
      <h1> Hello </h1>
      <h1> World </h1>
    </>,
    document.getElementById('root')
);
```

## ★ JavaScript Expression in JSX :—

JSX के अन्दर JavaScript use करने के लिए { } का use करते है

```
const name="Hariom Patidar";
ReactDOM.render(
  <>
    <h1> My Name is {name} </h1>
    <h1> 2 plus 3 is {2+3} </h1>
  </>,
  document.getElementById('root')
);
```

---

## ★ Challenge 2 :— Get Current Date and Time using Java script and print in JSX

```
const currDate = new Date().toLocaleDateString();
const currTime = new Date().toLocaleTimeString();

ReactDOM.render(
  <>
    <h1>Hello My name is {name} </h1>
    <p> Current date is {currDate} </p>
    <p> Current Time is {currTime} </p>
  </>,
  document.getElementById('root'),
);
```

---

## ★ Attributes in JSX :—

(i) Attribute should follow camel case

Ex :- contentEditable="true"

(ii) Most contain important attributes

Ex :- <img src="https://picsum.photos/200/300"
alt="Image not found" />

## ★ Add CSS :-

★ index.css :-

```
.heading {
  color: yellow;
  background: black;
}
```

index.js :-

★
```
import './index.css';
ReactDOM.render(
  <>
    <h1 className="heading"> Hello </h1>
  </>
  , document.getElementById('root')
);
```

(ii) Inline CSS :-

```
ReactDOM.render(
  <>
    <h1 style={{ color: 'green', backgroundColor: 'black',
                 textTransform: 'capitalize' }}>
      My Name is Hariom Patidar
    </h1>
  </>,
  document.getElementById('root')
);
```

OR

```
const heading = {
    color: 'yellow',
    backgroundColor: 'black',
    textTransform: 'capitalize'
}
ReactDOM.render(
  <>
    <h1 style={heading}> My name is Hariom Patidar </h1>
  </>,
  document.getElementById('root')
);
```

यहा पर object बनाओ और उसे {{ }} में pass कर दो style के अन्दर

# ★ Components :—

★ Heading.jsx :—
```
import React from 'react';
function Heading(){
    return <h1> Hello This is Haviom Pathdan </h1>
}
export default Heading;
```

★ Para.jsx :—
```
import React from 'react';
function Para(){
  return <p> This is my fav movies </p>
}
export default Para;
```

★ List.jsx :—
```
import React from 'react';
function List(){
  return <ol>
    <li> Movie one </li>
    <li> Movie two </li>
    <li> Movie three </li>
  </ol>
}
export default List;
```

★ index.js :—
```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
ReactDOM.render(<App/>, document.getElementById('root'));
```

★ App.js :—
```
import React from 'react';
import Heading from './Heading';
import Para from './Para';
import List from './List';
function App(){
  return (
    <>
      <Heading/>
      <Para/>
      <List/>
    </>
  );
}
export default List;
```

# ★ Import and Export :—

**App.js :—**

```
const fname="Harsom";
const lname="Patidar";
function fun1(){
   const name="Mr. Patidar";
   return name;
}
function fun2(){
   const bike="HRR";
   return bike;
}
export default fname;
export {lname, fun1, fun2};
```

Default में एक ही चीज़ को export कर सकते है।

बाकि को export करने के लिए { } के अन्दर लिखना है।

**index.js :—**

```
import React from 'react';
import ReactDOM from 'react-dom';
import fname, {lname, fun1, fun2} from './App';
        or
import * as allData from './App';

ReactDOM.render(
  <>
  <li>Default {fname}</li>
  <li>Normal {lname}</li>
  <li>Fun one {fun1()}</li>
  <li>Fun two {fun2()}</li>
  </>,
  document.getElementById('root')
);
```

normal data को normally function को call करके

array को dot(.) से

or

```
<>
<li>Default
  {allData.default}
</li>
<li>Normal
  {allData.lname}</li>
<li>Function one
  {allData.fun1()}</li>
</>
```

# ☆ Calculator :—

⇒ Calculator.js :—

```
function add(a,b){
  const sum=a+b;
  return sum;
}
function mult(a,b){
  let mult=a*b;
  return mult;
}
function sub(a,b){
  const sub=a-b;
  return sub;
}
function div(a,b){
  let div=a/b;
  div=div.toFixed(2);
  return div;
}
export {sum, sub, mult, div};
```

⇒ App.js :—

```
import React from 'react';
import {sum, sub, mult, div} from './Calculator';
function App(){
  return (
    <>
    <ul>
      <li>Addition is {add(40,3)}</li>
      <li>Substraction is {sub(40,3)}</li>
      <li>Multiplication is {mult(40,3)}</li>
      <li>Division is {div{40,3)}</li>
    </ul>
    </>
  );
}
export default App;
```

⇒ Index.js :—

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
ReactDOM.render(<App/>, document.getElementById('root'));
```

## ★ Props :— These are the arguments passed into React components via HTML attributes.

### ⇒ Cards.js :-

```
import React from 'react';
function Card(props){
  return(<>
    <div className='cards'>
      <div className='card'>
      <img src={props.imgsrc} alt='mypic' className='card-img'/>
      <div className='card-info'>
        <span className='card-category'>{props.title}</span>
      <h3 className='card-title'>{props.sname}</h3>
      <a href={props.link} target="_blank">
        <button>Watch Now</button>
      </a>
    </div>
  </div>
</div>
</>)
}
```

### ⇒ App.js :—

```
import React from 'react';
import Card from './Card';
function App(){
  return(
    <>
    <Card
     imgsrc="——"
     title="A Netflix Original Series"
     sname="DARK"
     link="——"
    />
    <Card
      ——
    />
  </>
)}
```

### ⇒ index.js :-

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <App/>, document.
    getElementById('root'));
```

# ☆ Netflix project using map function:—

App.js :—

```
import React from 'react';
import Card from './Cards';
import Sdata from './Sdata';
function ncard(val){
  return (
    <Card key={val.id}
      imgsrc={val.imgsrc}
      title={val.title}
      sname={val.sname}
      link={val.link}
    />
  )
}
function App(){
  return (
    <>
      <h1 className='heading'>Here is the list of Movies</h1>
      {Sdata.map(ncard)}
    </>
  );
}
export default App;
```

इसमे एक warning आती है तो उसका solution यह है कि starting में एक key होनी चाहिए

Using Arrow function:—

```
function App(){
  return (
<><h1 > Here is the list of Movies </h1>
  {Sdata.map((val) => {
    return (
      <Card
        imgsrc={val.imgsrc}
        title={val.title}
        sname={val.sname}
        link={val.link}
      />
);})}</>);}
export default App;
```

# ☆ React Hooks :—

(i) Hooks are the new features introduced in the React 16.8.
(ii) 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.
(iii) It does not work inside classes.
(iv) Hooks should always be used at the top level of the React function.

Ex :— App.js :—

```
const App=()=>{
  const[count, setCount] = useState(0);
  const IncNum=()=>{
    setCount(count+1);
  }
  return(
    <>
      <h1>{count}</h1>
      <button onClick={IncNum}>Click</button>
    </>
  )
}
export default App;
```

---

## Get Time And Update it onClick :—

App.js :—

```
const App=()=>{
  const [time, currTime] = useState(new Date().toLocaleTimeString(0));
  const GetTime=()=>{
    currTime(new Date().toLocaleTimeString());
  }
  return(<>
    <h1>{time}</h1>
    <button onClick={GetTime}>Get Time</button>
  </>)
}
export default App;
```

## ☆ Live Watch :-

App.js :-

```
const App=()=>{
  const [time,newTime]=useState(new Date().toLocaleTimeString());
  const updateTime=()=>{
    newTime(new Date().toLocaleTimeString());
  }
  setInterval(updateTime,1000);
  return(
    <h1>{time}</h1>
  )
}
export default App;
```

## ☆ Show Name of User on Main Heading :-

```
const App=()=>{
  const [name,setName]=useState("");
  const [fullName,setFullName]=useState();
  const inputEvent=()=>{
⇒   setName(event.target.value);
  }
  const onSubmit=()=>{
    setFullName(name);
  }
  return(
    <>
⇒   <h1>Hello {fullName}</h1>
    <input type="text" placeholder="Enter Your Name"
⇒     onChange={inputEvent} value={name} />
⇒   <button onClick={onSubmit}>Click Me</button>
    </>
  )
}
export default App;
```

## ★ Forms having two input fields :—

```
import React from 'react';
const App=()=>{
    const [name, setName] = useState("");
    const [lastName, setLastName] = useState("");
    const [firstNewName, setFirstNewName] = useState("");
    const [lastNewName, setLastNewName] = useState("");

    const inputEventOne=(event)=>{
        setName(event.target.value);
    }
    const inputEventTwo = (event)=>{
        setLastName(event.target.value);
    }
    const onSubmits = (event)=>{
        // for stoping form refresh
        event.preventDefault();
        setFirstNewName(name);
        setLastNewName(lastName);
    }
    return(
        <>
        <form onSubmit={onSubmit}>
            <h1> Hello {firstNewName} {lastNewName}</h1>
            <input type="text" placeholder="Enter your first Name"
                onChange={inputEventOne} value={name} />
            <input type="text" placeholder="Enter your last Name"
                onChange={inputEventTwo} value={lastName} />
            <button type='submit'> Submit Form </button>
        </form>
        </>
    )
}

export default App;
```

```
import React from 'react';
const App() => {
  const [fullName, setFullName] = useState({fname:"", lname:"", email:"", phone:""});
  const inputEvent = (event) => {
    const {value, name} = event.target;
    setFullName((preValue) => {
      if(name === "fname") {
        return {
          fname: value, lname: preValue.lname, email: preValue.email, phone: preValue.phone
        }
      } else if(name === "lname") {
        return {
          fname: preValue.fname, lname: value, email: preValue.email, phone: preValue.phone
        }
      }
    }
  }
  const onSubmits = (event) => {
    event.preventDefault();
    alert("Form Submitted");
  }
  return (
    <>
      <form onSubmit={onSubmits}>
        <h1> Hello {fullName.fname} {fullName.lname} </h1>
        <p> {fullName.email} </p>
        <p> {fullName.phone} </p>
        <input type="text" placeholder="Enter your first Name"
          name='fname' onChange={inputEvent} value={fullName.fname}/>
        <input type="text" placeholder="Enter your last Name"
          name="lname" onChange={inputEvent} value={fullName.lname} />
        <input type="email" placeholder="Enter your Email"
          name="Email" onChange={inputEvent} value={fullName.email} />
        <input type="number" placeholder="Enter your mobile number"
          name="phone" onChange={inputEvent} value={fullName.phone} />
        <button onSubmit="" type="submit">Click Here</button>
      </form>
    </>
  )
}

export default App;
```

Change this to these

```
return {
  ...preValue,
  [name]: value
}
```

## ☆ Increment Decrement challenge :-

```
const App=()=>{
  const [num, setNum] = useState(0);
  const inNum=()=>{
    setNum(num+1);
  }
  const deNum=()=>{
    if(num>0){
      setNum(num-1);
    }
    else{
      setNum(0);
      alert("Number is already Zero");
    }
  }
  return(
    <>
      <h1>{num}</h1>
      <button onClick={inNum}>Increment</button>
      <button onClick={deNum}>Decrement</button>
    </>
  )
}
```

---

## ☆ Context Api :-

1. const firstname = createContext()
2. .Provider
3. .consumer

## Why we use UseState Hook:—

React will load the page at once when we open it by, if we change any data dynamically by the user, then it will not load the data or show the add.
For showing that data we use useState hook.

Ex:—

```
import React from 'react';
export default function Check(){
    const thingArray = ['thing1', 'thing2'];
    const thingElement = thingArray.map(thing => <p>{thing}</p>)

    function addItem(){
        const newThing = `thing${thingArray.length+1}`;
        thingArray.push(newThing);
        console.log(thingArray);
    }
    return(
        <>
            <button onClick={addItem}>
                Add Item </button>
            {thingElement}
        </>
    )
}
```

Here the array is updated by it can not that new Element on display

---

Solution:—

```
const [thingsArray, setThingsArray] = useState(['Thing1', 'Thing2']);
const addItem = () => {
    setThingsArray(preThings => [...preThings, `Thing ${preThing.length}`])
}
const thingElements = thingsArray.map(thing => <p>{thing}</p>)
return(<>
    <button onClick={addItem}>Add Item</button>
    {thingElements}
</>)
```

**Props :-** Props refers to the properties being passed into a component in order for it to work correctly, similar to how a function receives parameters "from above". A component receiving props is not allowed to modify those props. (i.e. they are immutable").

**State:-** State refers to values that are managed by the component, similar to variables declared inside a function. Any time you have changing values that should be saved/displayed, you will likely be using state.

## Handle form with two or more fields:-

```
export default function Forms(){
    const [data, setData] = useState({
        fname:'',
        lname:"",
    });
    const handleEvent=(event)=>{
        setData(oldData=>{
            return{
                ...oldData,
                [event.target.name]: event.target.value,
            }
        })
    };

    return(
        <>
        <form>
            <input type='text' placeholder="Enter firstname"
                onChange={handleEvent}
                name='fname'
            />
            <input type='text' placeholder="Enter lastname"
                onChange={handleEvent}
                name='lname'
            />
        </form>
    </>
    )
}
```

# Foarm :-

```
export default function Faarms(){
  const [faarmData, setFaarmData] = useState({
      firstName:'',
      lastName :'',
      email :'',
      comments:'',
      isFriendly: false,
      employment:'',
      favColor:'',
  })

const handleEvent=(event)=>{
    const{name,value,type,checked} = event.target;
    setFaarmData( prevData=>{
      return{
        ...prevData,
        [name]: type==='checkbox'? checked: value
      }
    })
}

const submitFaarm=(event)=>{
    event.preventDefault();
    console.log(faarmData);
}

return(
  <>
  <faarm onSubmit={submitFaarm}>
    <input type='text' placeholder="Enter your name"
      onChange={handleEvent}
      name='firstName'
      value={faarmData.firstName}
    />
    <input type='text' placeholder="Enter your last name"
      onChange={handleEvent}
      name='lastName'
      value={faarmData.lastName}
    />
```

⇒ <input type='email' placeholder='Enter email'
onChange={handleEvent}
name='email'
value={formData.email}
/>

⇒ <textarea placeholder='Enter some comment'
onChange={handleEvent}
name='comments'
value={formData.comments}
/>

⇒ <input type='checkbox' id='isFriendly'
checked={formData.isFriendly}
onChange={handleEvent}
name='isFriendly'
/>

□ Checkbox में value की जगह checked लिखते है।

<label htmlFor='isFriendly'> Are You Friendly? </label>

⇒ <input type='radio' id='unemployed'
onChange={handleEvent} value={~~formData.employment~~} 'Unemployed'
name='employment' checked={formData.employment==='unemployed'}
/>
<label htmlFor='unemployed'> Unemployed </label>
<input type='radio' id='employed' onChange={handleEvent}
name='employment' value={'Employed'} checked={formData.employment==employed}
/>
<label htmlFor='employed'> Employed </label>

⇒ <select id='favColor' value={formData.favColor}
onChange={handleEvent} name='favColor'>
<option value=''>--Select</option>
<option value='Blue'>Blue</option>
<option value='Red'>Red</option>
</select>

<button>Submit</button>

</form>

—— o

# Access API:-

```
export const API=()=>{
  const [starWarData, setStarWarData] = useState({});

  fetch("https://swapi.dev/api/people/1")
    .then(res => res.json())
    .then(data => setStarWarData(data))
  Console.log(starWarData)
  return (
    <pre>{JSON.stringify(starWarData, null, 2)}</pre>
  )
}
```

Here it is rendering infinite time which is a side effect of using api.

---

```
const [allMemes, setAllMemes] = useState([])
React.useEffect(async ()=>{
    const res=await fetch("https://api.imgflip.com/get_memes")
    const data=await res.json();
    setAllMemes(data.data.memes)
},[])
```

---

# Remove the side effect of UseEffect hook :—

```
const [windowWidth, setWindowWidth] = useState(window.innerWidth);
React.useEffect( ()=>{
  const watchWidth=()=>{
    setWindowWidth( window.innerWidth);
  }
window.addEventListener('resize', watchWidth);
  return function (){
    window.removeEventListener('resize', watchWidth);
  }
}
```

This will execute normally

This will execute when the state is not used or not rendered or if our component died this return will remove the side-effect of the useEffect ~~stat~~ hook.

# React

JS (Library) — use → UI create करना

Component based architecture

The piece of code which we does not write but we can use it by invoking or calling it.

React is all about Components

User code — invoke → Library

Ex:- int ans = max(a,b)

Components → A reusable piece of code

Components → or → Custom HTML element

---

React → JS Library

→ Why react

Just JS :-
- HTML
- CSS
- Functionality

Tell only End state

after All things are handled by react

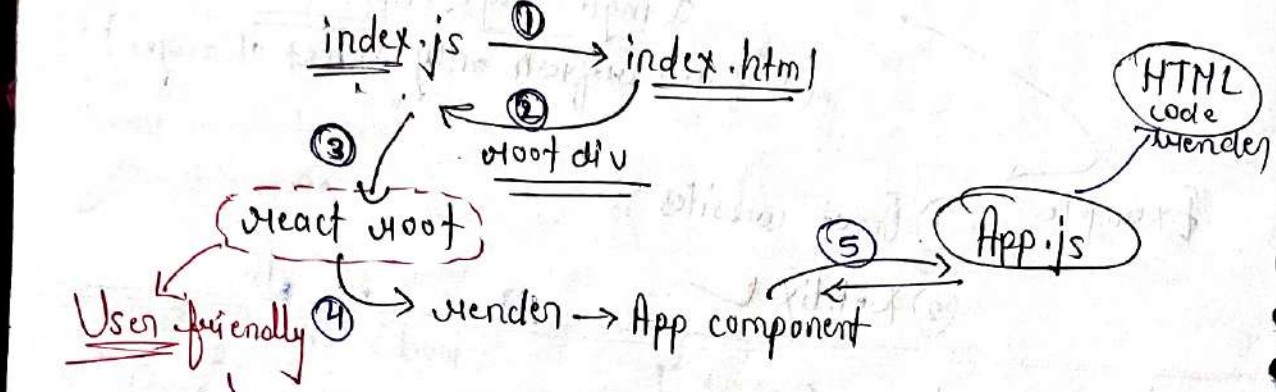
This approach is known as Declarative approach.

Imperative Approach

इसमें ये बताना पड़ता है कि
1. Body Tag को लाओ
2. Create New child
3. Insert content in it
4. Append that tag at that place

Step by step
Line by line

## Execution:-




Component :- ① Export ② import

Props :-

```
=> function App() {
    return (
      <div>
        <Item name="Nirmal" />
        <ItemDate day="20" month='Jan' year='1990'/>
        <Item name="SurfExcel"/>
        <ItemDate day='30' month='Feb' year='1998'/>
      </div>
    )
}
export default App;
```

```
=> function Item (props){
    const name = props.name;
  return (
    <p>{name}</p>
  )
}
export default Item;
```

```
=> function ItemDate (props){
    const day = props.day;
    const month= props.month;
    const year = props.year;
  return (
    <span>{day}</span>
    <span>{month}</span>
    <span>{year}</span>
  )
}
export default ItemDate;
```

# React ⟹ SPA approach

Single Page Application

(Does not refresh only content changes)

Example → ① Gfg out websites ✗

② Netflix ✓

## Why React:—

① Reusability

(ii) Less Duplicacy or Do Not Repeat Yourself Principle (DRY)

(3) Readability.

(4) Separation of Concern.

(5) Maintability.

(6) Clean code

## Setup:—

(i) Install Node js

(ii) VS code

(3) Create a new Folder

(4) Change directory to new Folder

(5) npx create-react-app <name/>

(6) Change directory to <name/>

(7) npm start.

⟹ o

Theprangupta /insta

~~const~~ function App() {

const response = [

{ name: 'Nivma', date: '20', ~~month~~: 'Jan', year: '1990' },

{ name: 'Surf', date: '25', month: 'Feb', year: '1993' },

{ name: '555', date: '30', month: 'Feb', year: '1999' }

];

const showItem = () => {

~~return~~

const data = response.map( newData => {

return (

<Item name = newData.name />

<ItemDate

day = newData.day

month = newData.month

year = newData.year

/>

)

}

)

}

return (

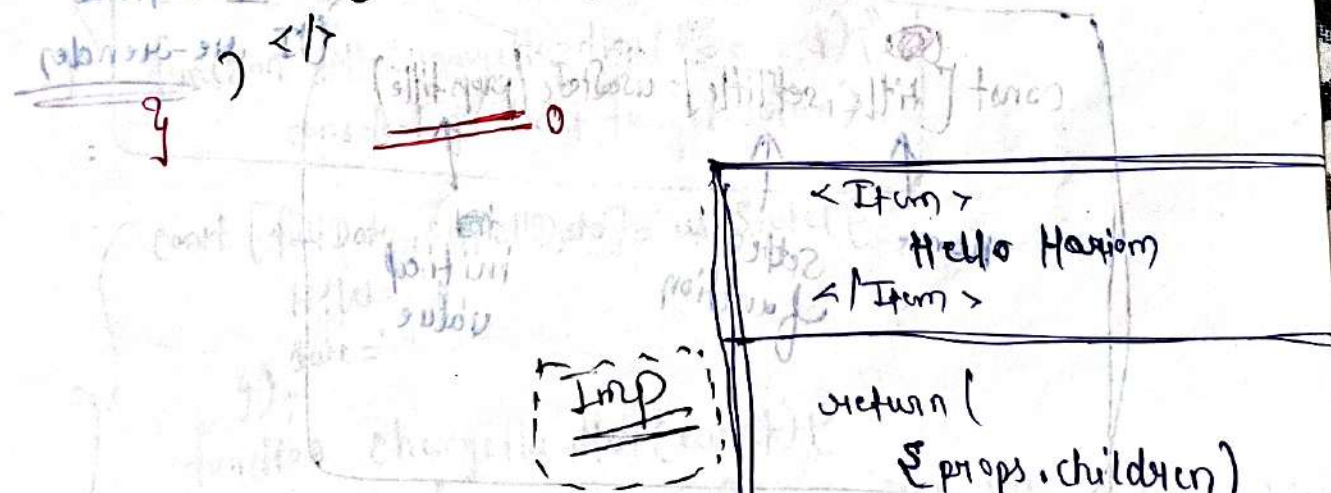
<>

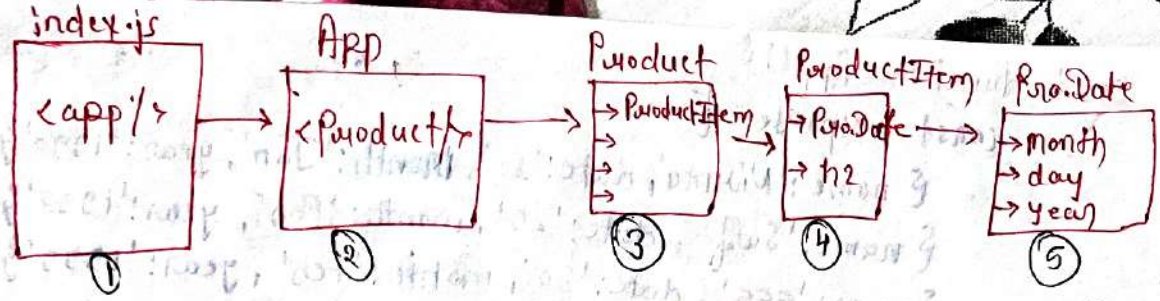
{ showItem }

</>

)

}

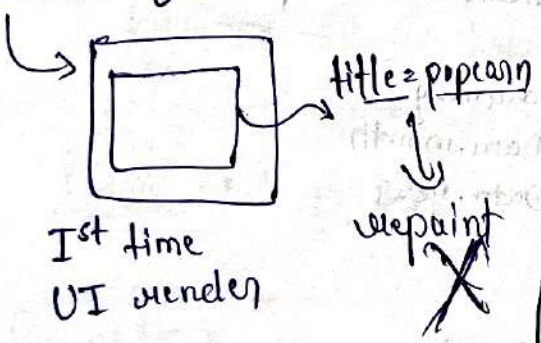





Event handling ⇒ Props (props addition in tag)

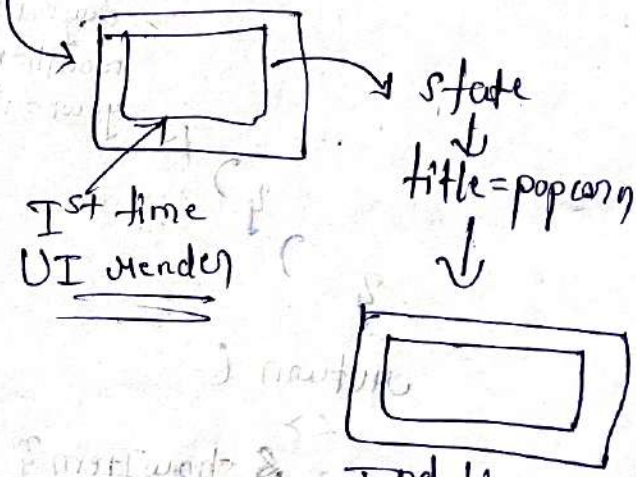
onClick={clickHandler}

Normal flow :-




Using State :-

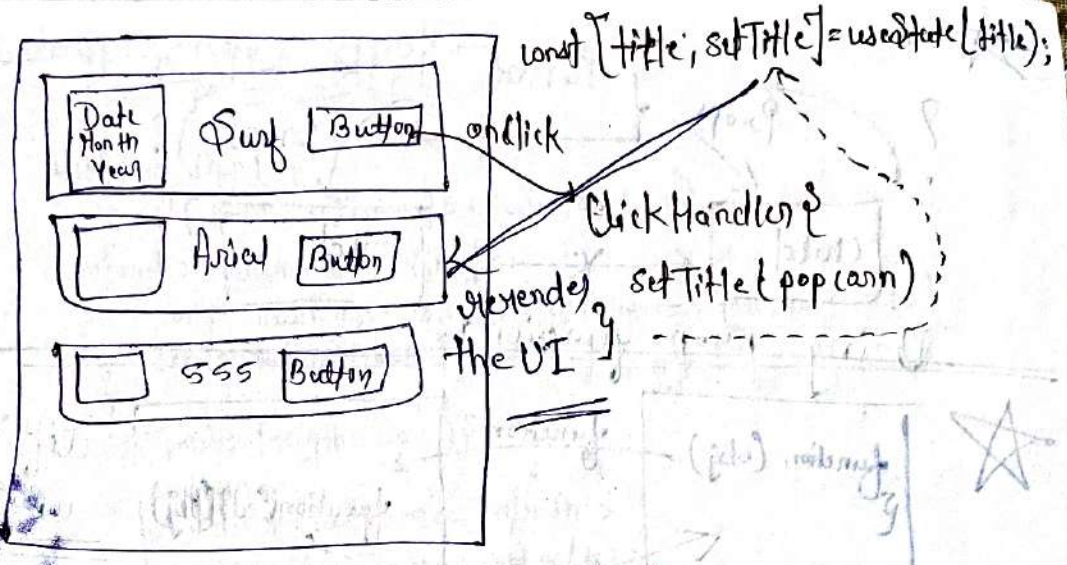
const [title, setTitle] = useState()




const [title, setTitle] = useState (props.title)

name — setter function — initial value

⇒ UseState is a utility function which rerender the UI when its variable value changes.



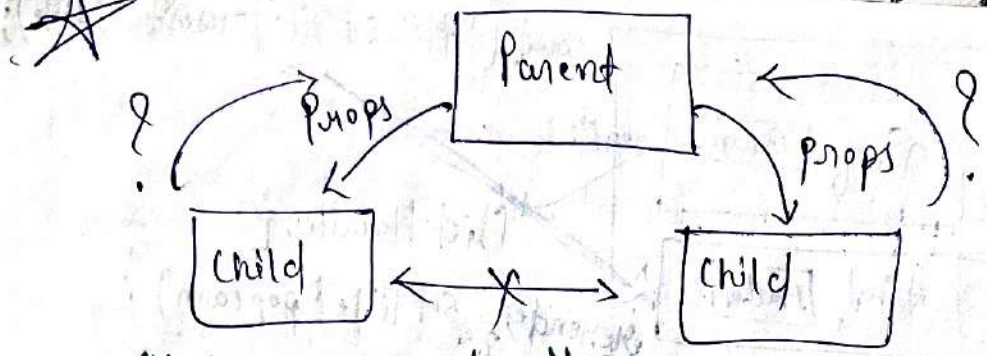

Q.1 Why usestate variable are constant.

Q.2 set___ will set value immediately or it will schedule it.
It will schedule the changes.

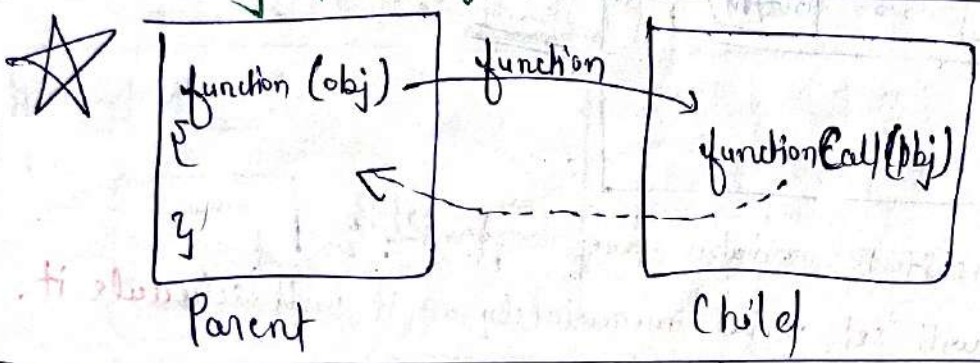
```
<input type='text' onChange={titleChangeHandler}
    value={title.title}
/>
```

```
const [title, setTitle] = useState('');
function titleChangeHandler(event){
    console.log(event.target.value)
```

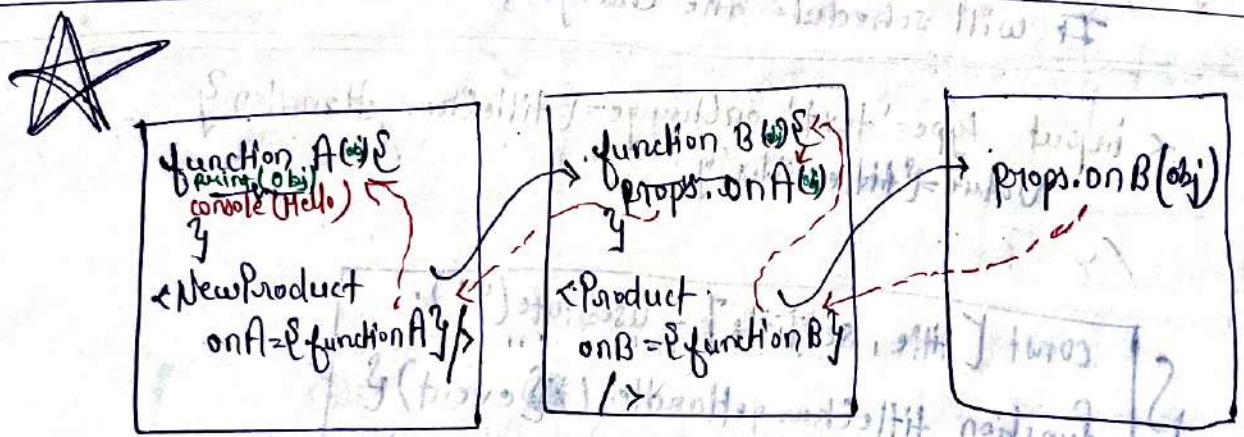
```
const [fullDate, setFullDate] = useState({
    title:'',
    date:''
});
function changeHandler(event){
    return (prevState => {...prevState, title: event.target.title})
}
```




Using prop function :-




lifting up




Counter App

## Card.js :-

```
function Card ({ id, image, info, price, name, removeTour }) {
  const [readmore, setReadMore] = useState(false);
  const description = readmore ? info : `${info.substring(0,150)}...`;
  function readmoreHandler() {
    setReadMore(!readmore);
  }
  return (
    <div className='card'>
      <img src={image} className='image' />
      <div className='tour-info'>
        <div className='tour-details'>
          <h4 className='tour-price'> ${price} </h4>
          <h4 className='tour-name'> {name} </h4>
        </div>
        <div className='description'>
          {description}
          <span  className='read-more' onClick={readmoreHandler}>
            {readmore ? 'Show less' : 'Read More'}
          </span>
        </div>
      </div>
      <button className='btn-red' onClick={() => removeTour(id)}>
        Not Interested
      </button>
    </div>
  )
}

export default Card;
```

# ☆ holidaypackage App :—

App.js :—

```
function App(){
  const[tours, setTours] = useState(data);
  function removeTour(id){
    const newTours = tours.filter(tour => tour.id !== id)
    setTours(newTours);
  }
  if(tours.length===0){
    return(
      <div className='refresh'>
        <h2> No Tours Left </h2>
        <button onClick={setTours(data)}> Refresh </button>
      </div>
    )
  }
  return(
    <div className='App'>
      <Tours tours={tours} removeTour={removeTour} />
    </div>
  )
}
```

Tours.js :—

```
function Tours({tours, removeTour}){
  return(
    <div className='container'>
      <div className='title-container'>
        <h2 className='title'> Plan With Harjot </h2>
      </div>
      <div className='cards'>
        {
          tours.map(tour => {
            return <Card key={tour.id} {...tour}
              removeTour={removeTour} />
          })
        }
      </div>
    </div>
  )
}
```

# ★ USE EFFECT Hook :—

* manage side effect

A side effect is a change that affect something outside the component being rendered.

If we clicked on one component and it cause change in some other component also then that is side effect.

(अगर UI render होने के बाद हम किसी Task को करवाना चाहते है और UI पर दिखाना चाहते है तो उस Task को useEffect के अन्दर डाल देंगे जिससे बस वही Task re-render हो और बाकि की UI फिर से render ना हो)

→ useEffect( —— , [] )

- —— : callback function
- [] : Array of Dependencies

Component mount हो चुका है मतलब component is Rendered.

Component Unmount हो चुका है मतलब component DOM से remove हो चुका है

Extra :

## Life Cycle methods :—

Class-based components

- componentDidMount
- componentDidUpdate
- componentWillUnmount

# Four Ways to use useEffect Hook:-

① useEffect( () => {

});

⇒ Every Render

② useEffect( () => {

}, []);

⇒ First Render

③ useEffect( () => {

}, [name]);

⇒ First render + when the value of name is changed

Dependency Array — Dependency

④ useEffect( () => {

return ( () => {

})

});

First render पर ~~Add~~ normal

⇒ After that return वाला पहले फिर normal wala execute होगा

## useEffect hook usage Example:-

```
function App(){
    const [text, setText] = useState('');
    function changeHandler(event){
        setText(event.target.value);
        console.log(text);
    }
```

① case 1 :- Every Render

```
useEffect( ()=>{
  clg("UI rendering done");
});
```

② case 2 :- First Time Rendering

```
useEffect( ()=>{
  clg("UI rendered 1st time");
}, [] );
```

↑ use empty array

③ case 3 :- When Dependency (value) changes

```
useEffect( ()=>{
  clg("Change observed in Dependency");
}, [text] );
```

④ case 4 :- to handle the unmounting of a component :-

```
useEffect( ()=>{
    clg("Listener Added");
    return (
      clg("Listener Removed");
}, [] );
```

```
return (
  <div className='App'>
    <input type='text' onChange={changeHandler} />
  </div>
);
}
```

# ★ React Router:—

React router is used when we want to move from one page to another page in a SPA without Refreshing. It dynamically changes the content in the page

[इससे ऐसे लगता है कि हम दूसरे Page पर गये है पर वह पर एक component की जगह दूसरा component load होता है बस]

It is a framework ~~which~~ from which we can navigate from one page to another page without refreshing the page

① install react-router-dom :—

⇒ npm install react-router-dom

---

App.js / index.js :—

```
import App from './App';
import { BrowserRouter } from 'react-router-dom';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App/>
  </BrowserRouter>
)
```

App.js :-

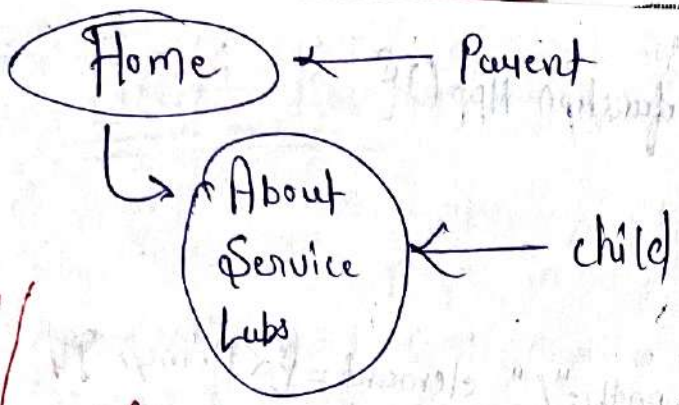
```
expaort default function App(){
return (
 <div>
  <Routes>
    <Route path="/" element={<Home/>}/>
    <Route path='/about' element={<About/>}/>
    <Route path='/support' element={<Support/>}/>
    <Route path='/labs' element={<Labs/>}/>
    <Route path='*' element={<NotFound/>}/>

  </Routes>
 </div>
 <div>
  <nav>
    <Link to="/">Home</Link>
    <Link to="/about">About</Link>
    <Link to="/support">Support</Link>
    <Link to="/labs">Labs</Link>
  </nav>
```

Mapped these path to here use to=" "

* using NavLink tag which adds Active class in tag :-

```
<div>
  <nav>
    <NavLink to='/'>Home</NavLink>
    <NavLink to='/about'>About</NavLink>
    <NavLink to='/support'>Support</NavLink>
    <NavLink to='/labs'>Labs</NavLink>
  </nav>
</div>
```




App.js :-

```
<Routes>
  <Route path="/" element={<Home/>}>
    <Route path='/about' element={<About/>}/>
    <Route path='/service' element={<Service/>}/>
    <Route path='/labs' element={<Labs/>}/>
    <Route path='*' element={<NotFound/>}/>
  </Route>
</Routes>
```

यहा पर Parent route element child को दिखने नही देता है या render नही होने देता है।

And if we want to give permission of rendering a child element inside parent then we have make use of `<Outlet/>` tag in the parent.

Home.js

```
const Home=()=>{
  return (
    <div>
      <Outlet/>
      This is Home page
    </div>
  )
}
```

But the problem after using Outlet tag also is that when we click on other page then also the Home page will be rendered.

It's solution is:—

① Create one more component and put all data of Home page into that page ⇒ MainHeader ⟸ Home

② Create default route using index

③ Make main route to that new component.

```
<Routes>
  <Route path="/" element={<MainHeader/>}/>
    <Route index element={<Home/>}/>   ⟸ default Route
    <Route path='/about' element={<About/>}/>
    <Route path='/service' element={<Service/>}/>
    <Route path='/labs' element={<Labs/>}/>
    <Route path='*' element={<NotFound/>}/>
  </Route>
</Routes>
```

Home.js

```
const Home=()=>{
  return (
    <div>
      This is Home page
    </div>
  )
}
```

MainHeader.js

```
const MainHeader=()=>{
  return (
    <Outlet/>
    This is Home page
  )
}
```

Same Content

## ✱ For going Go back on time :—

```
import { useNavigate } from 'react-router-dom';
const Page = () => {
  const navigate = useNavigate();
  function goBackHandler(){
    navigate(-1)
  }
  return(
    <button onClick={goBackHandler}>
    Simand Go Back
    </button>
  )
}
```

## ★ useNavigate Hook :—

This hook is used for navigating from one page to another page from that current page.

```
import { useNavigate } from 'react-router-dom';
const About = () => {
    const navigate = useNavigate();
    function clickHandler(){
        navigate("path");        // navigate("/about")
    }
    return (
    <div>
        <Button onClick={clickHandler}>
            Go to that page
        </Button>
    </div>
    )
}
```

# ★ Custom Hook: (एक component बना दो use आगे लिख कर)

**React hook:** — "It is a utility function used to perform certain task."

Random Gif generator:—

```
const API_key = process.env.GIPHY_API_KEY;
const Random = () => {
  ⇒ const [gif, setGif] = useState('');
  ⇒ const [loading, setLoading] = usestate('false');
⇒ async function fetchData() {
    setLoading(true);
    ⇒ const url = 'https://api.giphy.com/v1/gifs/random?api_key=${APIkey}';
    ⇒ const {data} = await axios.get(url);
    ⇒ const imageSource = data.data.images.downsized_large.url;
    ⇒ setGif(imageSource);
    setLoading(false);
  }
⇒ useEffect( ()=>{fetchData()}, []);
⇒ function clickHandler() {
    fetchData();
  }
return (
  <div>
    <h1> Random GIF </h1>
    {
    loading ? (<Spinner/>) : (<img src={gif} width="450" />);
    }
   <button onClick={clickHandler} >
      Generate GIF
   </button>
) </div>
}
```

## * useGif.js :—

```
const useGif = (tag) => {
    const [gif, setGif] = useState('');
    const [loading, setLoading] = useState(false);

    async function fetchData(tag) {
        setLoading(true);
        const {data} = await axios.get(tag ? (`${url}&tag=${tag}`)
                                           : (url));
        const source = data.data.images.downsized_large.url;
        setGif(source);
        setLoading(false);
    }

    useEffect(() => fetchData('car'), []);

    return {gif, loading, fetchData}
}
export default useGif;
```

---

```
const Tag = () => {
  const [tag, setTag] = useState('car');
  const {gif, loading, fetchData} =
                        useGif(tag);

  return (
    <div>
      <h1> Random {tag} Gif </h1>
      {
        loading ? (<Spinner />) : (<img src={gif} />)
      }
      <input ——— />
      <button onClick={() => fetchData(tag)}>
        Generate </button>
    </div>
  )
}
```

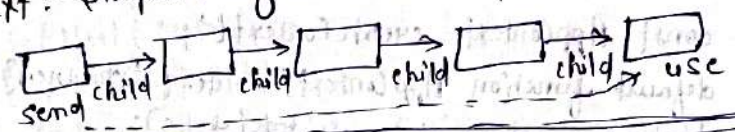
```
const Random = () => {
  const {gif, loading, fetchData} =
                        useGif();
  return (
    <div>
      <h1> Random Gif </h1>
      {
        loading ? (<Spinner />) : (<img ——— />);
      }
      <button onClick={() => fetchData()}>
        Generate
      </button>
    </div>
  )
}
```

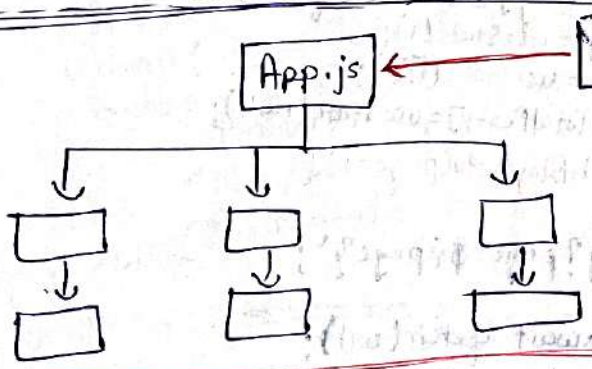
GIF generate according to tag:—

```
const Tag = () => {
  const [gif, setGif] = useState('');
  const [loading, setLoading] = useState(false);
  const [tag, setTag] = useState('');
  async function fetchData() {
    setLoading(true);
    const url = 'https://api.giphy.com/v1/gifs/random?api_key=${API_key}
                                                &tag=${tag}';
    const {data} = await axios.get(url);
    const imageSource = data.data.images.downsized_large.url;
    setGif(imageSource);
    setLoading(false);
  }
  useEffect(() => fetchData(), []);
  function clickHandler() {
    fetchData()
  }
  return (
    <div>
      <h1> Random {tag} GIF </h1>
      {
      loading ? (<Spinner/>) : (<img src={gif} width="450" />);
      }
      <input
        onChange={(event) => setTag(event.target.value)}
        value={tag}
      />
      <button onClick={clickHandler}>
        Generate GIF
      </button>
    </div>
  );
}
```

# ✱ Context API :—

⇒ context :— Snapshot of data at a perticular time.




all the child of App.js can use the data written inside the context.

## Context API :—

↳ (i) createContext() → Creation

↳ (ii) Provider ✱ → context provide

↳ (iii) Consume

Folder :— Context

↳ AppContext.js

↳ (i)

```
import { createContext } from 'react';
export const AppContext = createContext();
```

(ii)

```
function AppContextProvider({children}) {
  (1) {
    const [loading, setLoading] = useState(false);
    const [posts, setPosts] = useState([]);
    const [page, setPage] = useState(1);
    const [totalpages, setTotalPages] = useState(null);
  }
  (2) { const value = { loading, setLoading, posts, setPosts, page,
        setPage, totalpages, setTotalPages } }
  (3) {
    return <AppContext.Provider value={value}>
      {children}
    </AppContext.Provider>
  }
}
export default AppContextProvider;
```

(1) Context creation →

index.js

```
<AppContextProvider>
  <App />
</AppContextProvider>
```

App component AppContextProvider का child बन गया।

baseUrl.js :— export const baseUrl = "https://codehelp-apis.vercel.app/api/get-blog?"

`?page=${page}`

AppContext.js :—

```
export const AppContext = createContext();
export default function AppContextProvider({children}) {
    const [loading, setLoading] = useState(false);
    const [page, setPage] = useState(1);
    const [post, setPost] = useState([]);
    const [totalPages, setTotalPages] = useState(null);
  async function fetchBlogPosts(page=1) {
    setLoading(true);
    let url = `${baseUrl}?page=${page}`;
    try {
      const result = await fetch(url);
      const data = await result.json();
      setPage(data.page);
      setPost(data.post);
      setTotalPages(data.totalPages);
    }
    catch(error) {
      console.log("Error in fetching data");
      setPage(1);
      setPost([]);
      setTotalPages(null);
    }
    setLoading(false);
  }
  function handlePageChange(page) {
      setPage(page);
      fetchBlogPage(page);
  }
  const value = { loading, setLoading, post, setPost, page, setPage,
                  totalPages, setTotalPages, fetchBlogPage, handlePageChange };
  return <AppContext.Provider value={value}> {children}
      </AppContext.Provider>
}
```

App.jsx :-

```
function App() {
  const { fetchBlogPosts } = useContext(AppContext);
  useEffect(() => {
    fetchBlogPosts()
  }, []);
  return (
    <div>
      <Header />
      <Blog />
      <Pagination />
    </div>
  );
}
```

Pagination.js :-

```
function Pagination() {
  const { page, totalPages, handlePageChange } = useContext(AppContext);
  return (
    <div>
      { page > 1 &&
        <button onClick={() => handlePageChange(page - 1)}>
          Previous
        </button>
      }
      { page < totalPages &&
        <button onClick={() => handlePageChange(page + 1)}>
          Next
        </button>
      }
      <p> Page {page} of {totalPages} </p>
    </div>
  );
}
```

Blog.js :—

```
const Blog = () => {
  const { loading, posts } = useContext(AppContext);
  return (
    <div>
      loading ? (<Spinner/>) :
      (posts.length === 0 ? (<p> No post found </p>) :
        posts.map( (post) => (
          <div key={post.id}>
            <p> {post.title} </p>
            <p> By {post.author} on {post.category} </p>
            <p> Posted on {post.date} </p>
            <p> {post.content} </p>
            <div>
              {post.tags.map( (tag, index) => {
                return <span key={index}> #{tag} </span>
              })}
            </div>
          </div>
        ))
      )
    </div>
  )
}
```

## Redux Toolkit:-

Redux → Global State

↳ Slices

Centeralize store

Slices Many

↓

Portion of Data

---

Ex:- Counter App

⇓

Counter Slice

- name → counter
- initial state → 0
- functionality / Reducers → inc → { }, dec → { }, reset → { }

By using createSlice function

---

## Installation:-

⇒ npm install @reduxjs/toolkit react-redux

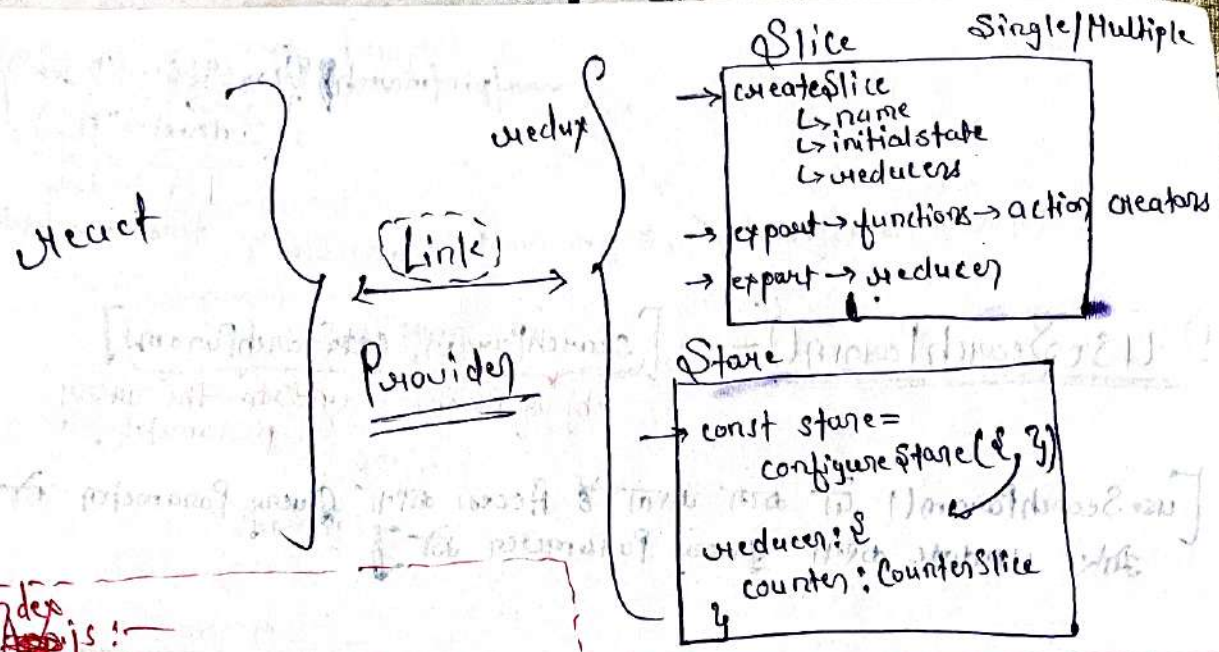
src/redux/slices/CounterSlice.jsx :—

```
import {createSlice} from "@reduxjs/toolkit";
const initialState={
    value:0,
}
export const CounterSlice= createSlice({
    name:"counter",
    initialState,
    reducers:{
        increment: (state)=>{
            state.value+=1;
        },
        decrement: (state)=>{
            state.value-=1;
        }
    }
})
export const {increment, decrement} = CounterSlice.actions;
export default CounterSlice.reducer;
```

---

src/redux/store.js :—

```
import {configureStore} from '@reduxjs/toolkit'
export const store = configureStore({
    reducer: {
        counter: CounterSlice
    }
})
```




index.js :-

```
<Provider store={store}>
  <App/>
</Provider>
```

Counter.jsx } component

```
const Counter = () => {
  const count = useSelector((state) => state.counter.value)
  const dispatch = useDispatch();
  return (
    <div>
      <button onClick={() => dispatch(increment())}>
        Increment
      </button>
      <div>{count}</div>
      <button onClick={() => dispatch(decrement())}>
        Decrement
      </button>
    </div>
  )
}
```

~/pic/march ?tag="friend" &&
level="A"

For finding these parameters

① useSearchParam() ⇒ [searchParam, setSearchParams]

searchParam → obj of params
setSearchParams → update the query parameter.

[useSearchParam() दो काम करता है Access करना Query Parameter को और Update करना Query Parameter को]

---

② useLocation() ⇒ Pathname, Hash, value, key, search value
1 2 3 4

~/pic/march/13

↳ uselocation().pathname = "pic/march/13"
.split('/') = ["pic" "march" 13]
.at(-1) = 13

add in path

↳ uselocation().search("page=1") → added this in search path

~/pic/march/13?page=1

# Shoping Cart Project :-

Folder structure :-

- src
  - components
    - CartItem.jsx, Navbar.jsx, Product.jsx, Spinner.jsx
  - pages
    - Cart.jsx, Home.jsx
  - redux
    - Slices
      - CartSlice.js
    - Store.js
  - App.js
  - Index.js

---

CartSlice.js :-

```
export const CartSlice = createSlice({
    name: "cart",
    initialState: [],
    reducers: {
        add: (state, action) => {
            state.push(action.payload);
        },
        remove: (state, action) => {
            return state.filter( item => item.id !== action.payload);
        }
    }
});
export const {add, remove} = CartSlice.actions;
export default CartSlice.reducer;
```

Store.js :-

~~import CartReducer from "./Slice~~

```
export const store = configureStore({
    reducers: {
        cart: CartSlice.
    }
});
```

} or {

```
import CartReducer from "./slices/CartSlice"
cart: CartReducer
```

index.js :-

```
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <Provider store={store}>
      <App />
      <Toaster />
    </Provider>
  </BrowserRouter>
);
```

App.js :-

```
const App = () => {
  return (<div>
    <div>
      <Navbar />
    </div>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/cart" element={<Cart />} />
    </Routes>
  </div>
};
```

Navbar.js :-

```
const Navbar = () => {
  <div>
    <div>
      <NavLink to="/">
        <img src="../logo.png" />
      </NavLink>
    <div>
    <NavLink to="/">
      <p> Home </p>
    </NavLink>
    <NavLink to="/cart">
      <FaShoppingCart />
    </NavLink>
  </div>
</div></div>
```

Product.jsx :-

```
const Product = ({post}) => {
  const {cart} = useSelector((state) => state);
  const dispatch = useDispatch();
  const addToCart = () => {
    dispatch(add(post));
    toast.success("Item added to cart");
  }
  const removeFromCart = () => {
    dispatch(remove(post.id));
    toast.error("Item removed from cart");
  }
return (
  <div>
    <div>
      <p> {post.title} </p>
    </div>
    <p> {post.description} </p>
    <img src={post.image} />
    <p> {post.price} </p>
    {
      cart.some((p) => p.id === post.id) ?
      (<button onClick={removeFromCart}>
        Remove Item
      </button>) :
      (<button onClick={addToCart}>
        Add to Cart
      </button>)
    }
  </div>
);
};
```

("root"));

Home.jsx :—

```
const Home = () => {
  const API_URL = "https://fakestoreapi.com/products";
  const [loading, setLoading] = useState(false);
  const [posts, setPosts] = useState([]);
  async function fetchProductData() {
    setLoading(true);
    try {
      const res = await fetch(API_URL);
      const data = await res.json();
      setPosts(data);
    } catch (error) {
      clg("Error aagya jee");
      setPosts([]);
    }
    setLoading(false);
  }
  useEffect(() => fetchProductData(), []);
return (
  <div>
    {
      loading ? <Spinner /> :
      posts.length > 0 ?
      (<div>
        {
          posts.map((post) => {
            <Product key={post.id} post={post} />
          })
        }
      </div>) :
      (<div>
        <p>No data found</p>
      </div>)
    }
  </div>
);
};
```

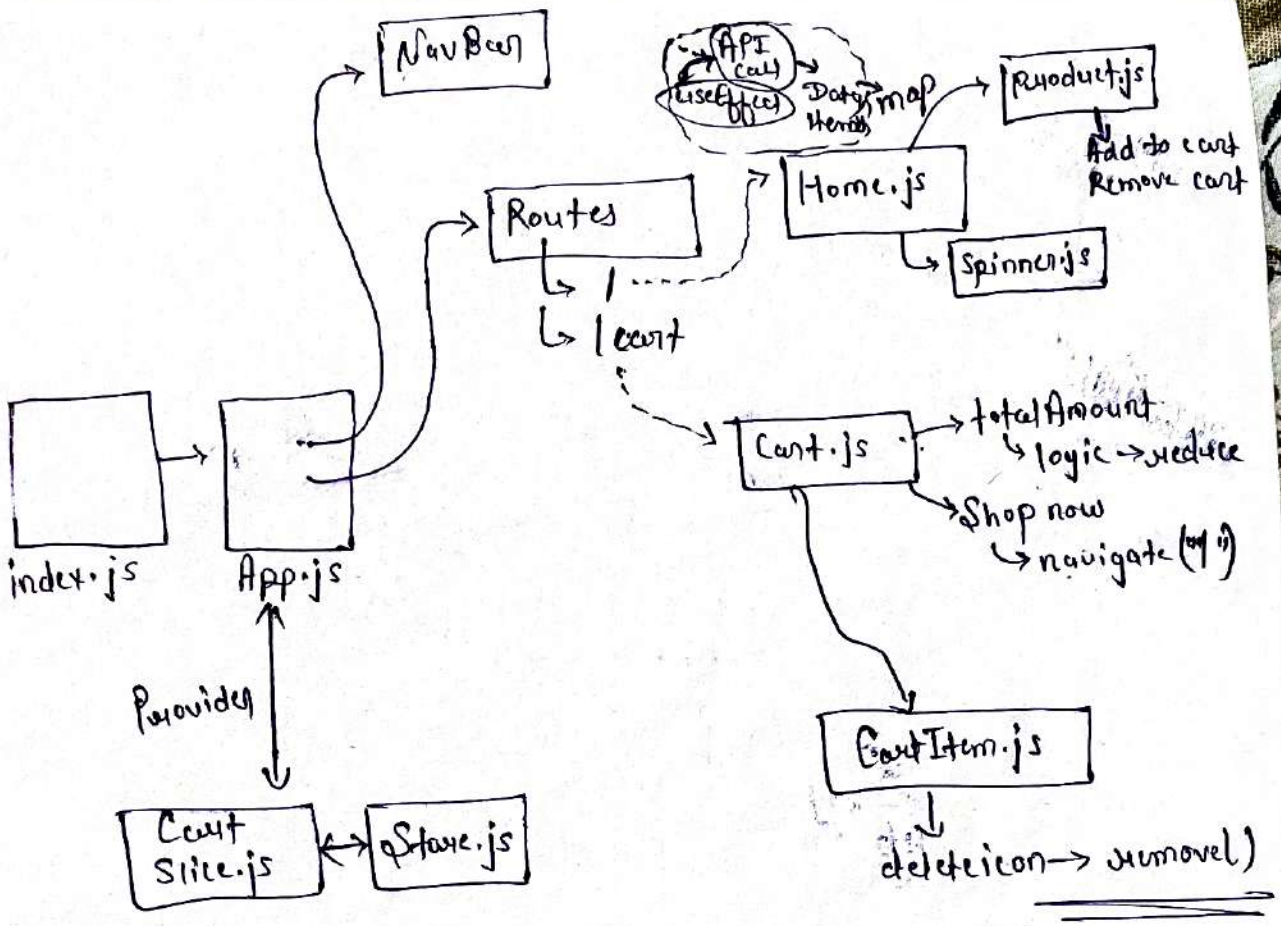
Cart.jsx :-

```
const Cart = () => {
  const {cart} = useSelector((state) => state);
  const [totalAmount, setTotalAmount] = useState(0);
  useEffect( () => {
    setTotalAmount(cart.reduce((acc, curr) => acc + curr.price, 0));
  }, [cart]);

return (
  <div>
    {
      cart.length > 0 ?
      ( <div>
          <div>
            {
              cart.map( (item, index) => {
                return <CartItem key={item.id} item={item}
                        itemIndex={index} />
              })
            }
          </div>
        <div>
          <div> Your Cart </div>
          <div> Summary </div>
          <p> Total Items : {cart.length} </p>
          <p> Total Amount : ${totalAmount} </p>
          <button> Checkout Now </button>
        </div>
      ) :
      ( <div>
          <h1> Cart Empty </h1>
          <Link to={"/"}>
            <button> Shop Now </button>
          </Link>
        </div>
      )
    }
  </div>
);
};
```

CartItem.jsx :—

```
const CartItem = ({ item, itemIndex }) => {
  const dispatch = useDispatch();
  const removeFromCart = () => {
    dispatch(remove(item.id));
    toast.error("Item removed");
  }
return (
<div>
  <div>
    <img src={item.length} /> </div>
  <div><h1> {item.title} </h1>
    <h1> {item.description} </h1>
    <div>
      <p> {item.price} </p>
      <div onClick={removeFromCart}>
        <FcDeleteDatabase/>
      </div>
    </div>
  </div>
</div>
);
};
```


NavBar
API call
useEffect()
Data
map
Items
Product.js
Add to cart
Remove cart
Home.js
Routes
Spinner.js
/
/cart
Cart.js
totalAmount
logic → reduce
Shop now
navigate("/")
index.js
App.js
Provider
CartItem.js
Cart Slice.js
Store.js
deleteIcon → removel)