

React? - JS Library - UI create

only understand
↓
Custom HTML elements

↳ React is all about components.

↓
A Reusable code

why? → only JS Imperative Approach.

React only end step need
↑ easy, less code write
↑ Declarative approach

all step are need
↑ more code write.

SPA Approach

↳ Single Page Application
→ Dynamic change

React Alternative

↳ Angular
↳ Vue

→ core syntax & Setup & JSX
→ component
→ data.

Setup

- 1) install node js
- 2) vs code
- 3) create a new folder
- 4) move to new folder
- 5) npx create-react-app app name
- 6) move to app
- 7) npm start

entry point = index.js

JSEX ← special syntax

src → app.js ←

index.css X

index.js

First File

→ react-root

Public → index.html → 'root'

→ render(APP)

Proper :-

```
import './App.css'
// import Item from './componentes/Item'
import User from './componentes/User'
import ItemDate from './componentes/ItemDate';

function App() {
  const twoName = "surffe";
  return (
    <div>
      <User name = "Nirama"></User>
      <ItemDate day = "20" month = "june" year="1994"></ItemDate>
      <User name = {twoName} ></User>
      <ItemDate day = "25" month = "jan" year="2003"></ItemDate>
      <User name = "dhadi" ></User>
      <ItemDate day = "05" month = "dec" year="2015"></ItemDate>
      <div className="App">
        Hello Jee
      </div>
    </div>
  );
}
export default App;
```

```
import './User.css'
function User(props){
  const ItemName = props.name;
  return(
    <p className="user">{ItemName}</p>
  );
}
export default User;
```

```
import './ItemDate.css'
function ItemDate(props){
  const day = props.day;
  const month = props.month;
  const year = props.year;
  return (
    <div className="date">
      { /* <span>20</span>
      <span>june</span>
      <span>1998</span> */ }
      <span>{day}</span>
      <span>{month}</span>
      <span>{year}</span>
    </div>
  );
}
export default ItemDate;
```

```
import './App.css'
// import Item from './componentes/Item'
import User from './componentes/User'
import ItemDate from './componentes/ItemDate';

function App() {
  const twoName = "surffe";
  const response = [
    {
      itemName : "Nireama",
      itemDay : "20",
      itemMonth : "JUne",
      itemYear : "1994"
    },
    {
      itemName : "Nireama",
      itemDay : "20",
      itemMonth : "JUne",
      itemYear : "1994"
    },
    {
      itemName : "Nireama",
      itemDay : "20",
    }
  ]
}
```

```

    itemMonth : "JUne",
    itemYear : "1994"
  }
];
return (
  <div>
    <User name = {response[0].itemName}></User>
    <ItemDate day = {response[0].itemDay} month = {response[0].itemMonth}
    year="1994"></ItemDate>
    <User name = {twoName} ></User>
    <ItemDate day = "25" month = "jan" year="2003"></ItemDate>
    <User name = "dhadi" ></User>
    <ItemDate day = "05" month = "dec" year="2015"></ItemDate>

    <div className="App">
      Hello Jee
    </div>

  </div>
);
}
export default App;

```

↑
 props.children :- throw any component between
 print that time used.

used in ex <Item> Hi, heeloogi </Item>
 ↑
 any component Item.js
 in any component ↳ {props.children}
 display that time
 used in parent component.

```

import './User.css'
function User(props){
  const ItemName = props.name;
  return(
    <div>
      <p className="user">{ItemName}</p>
      {props.children}
    </div>
  );
}
export default User;

```

```

<User name = {response[0].itemName}>
  hello me hu don
</User>

```

props.className = css add a parent

Event → props.throw event handle

↳ onClick = {

↑
 functionname

on keyword start thay.

↑ not a
 bracket ()
 used
 because the
 used them

because the
used them
automatic
run.

UI update & states used

useState :- react-hook (utility function)

useState(↘ initialise value

↑ return two value (value, a function for
of the updating
variable. the value.

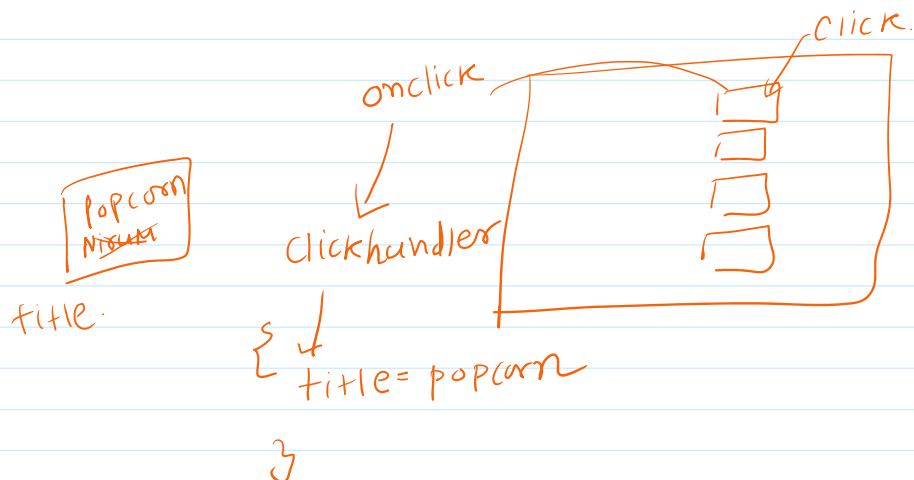
const [title, setTitle] = useState(props.title)

↑ update function

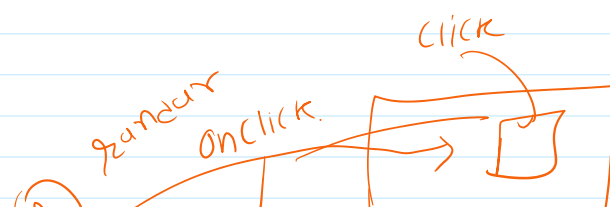
↑ old = "Normal"

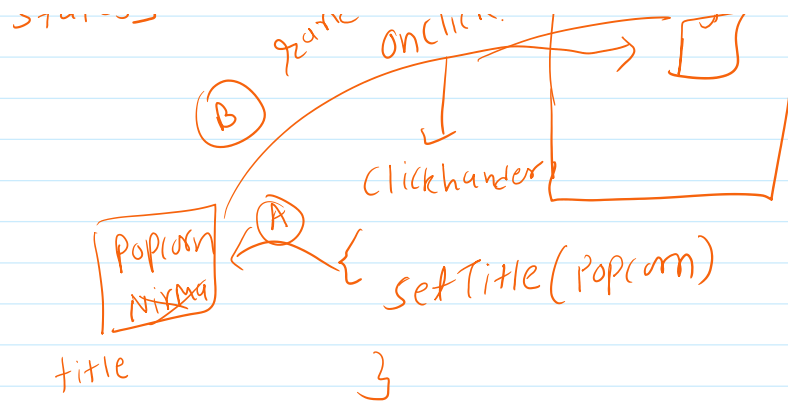
setTitle("popcorn") ;

Before → [we don't know about state]



after = [we know
states]





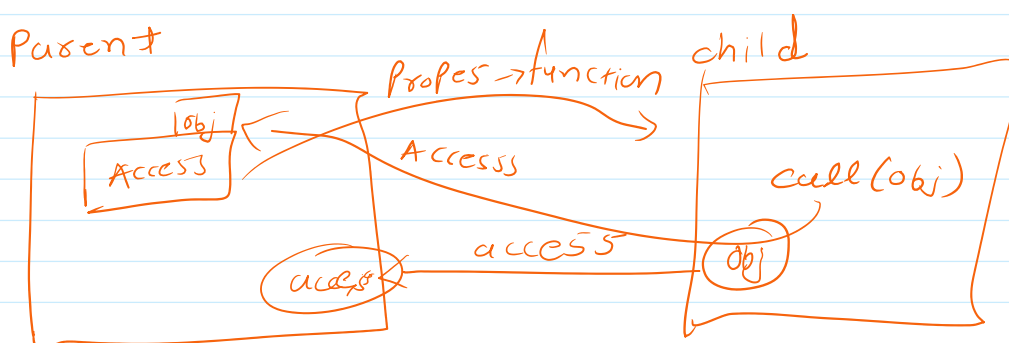
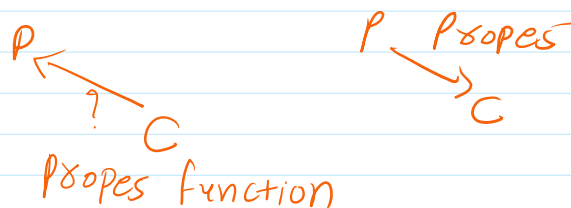
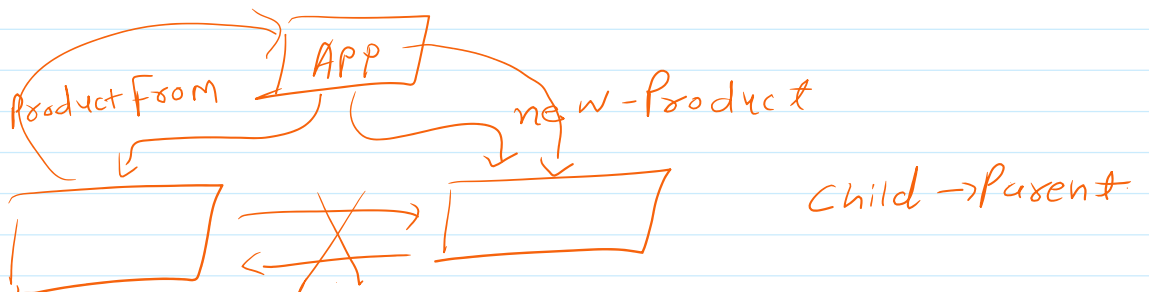
`onchange = { }`
 ↑ value in input field.

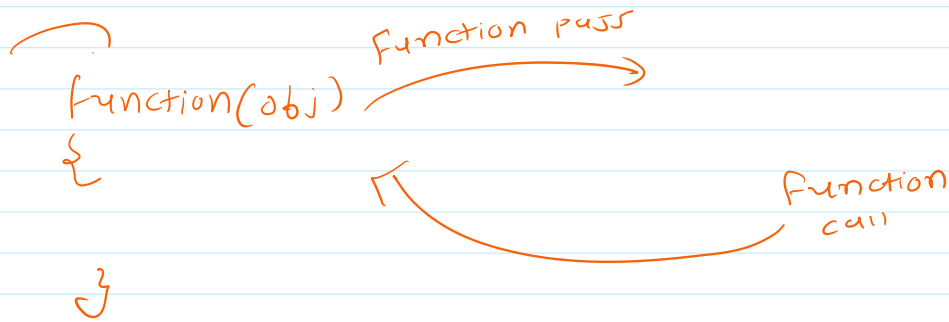
`event.target.value`

↳ console me print.
 ↳ only single

default behavior avoid :- `event.preventDefault()`

`onSubmit = { }`

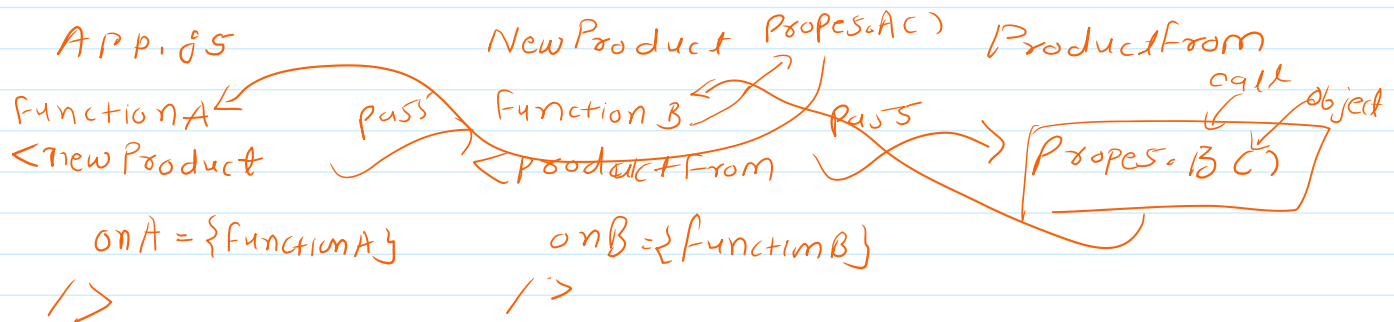




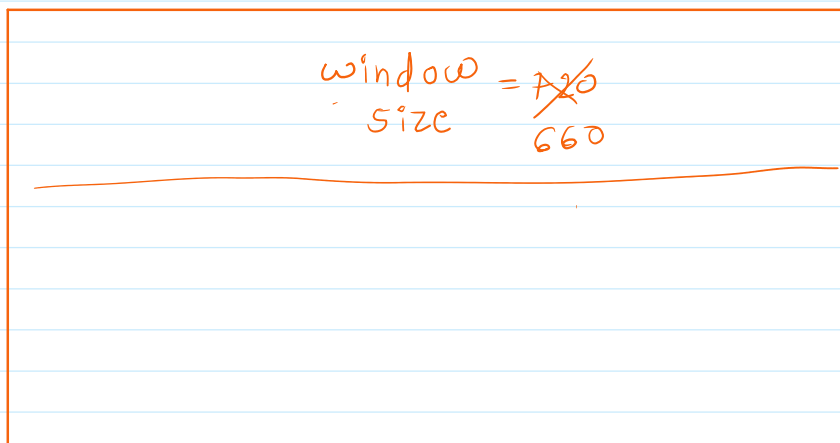
function object create a parent.

Function pass a child throw a props.

child are called a function throw a obj parameter
Access the object a parent.



useEffect :- Manage side effects



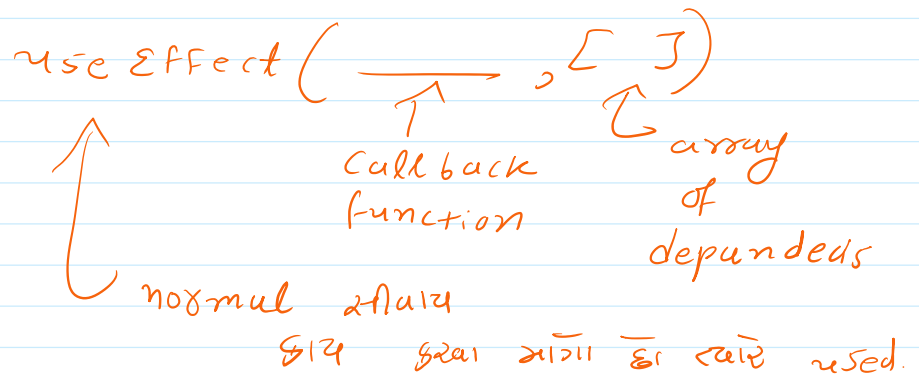
if any component are change then other any place are change that handle that used.

component render and use useEffect used.

Component render aur useEffect used.

↳ side task.

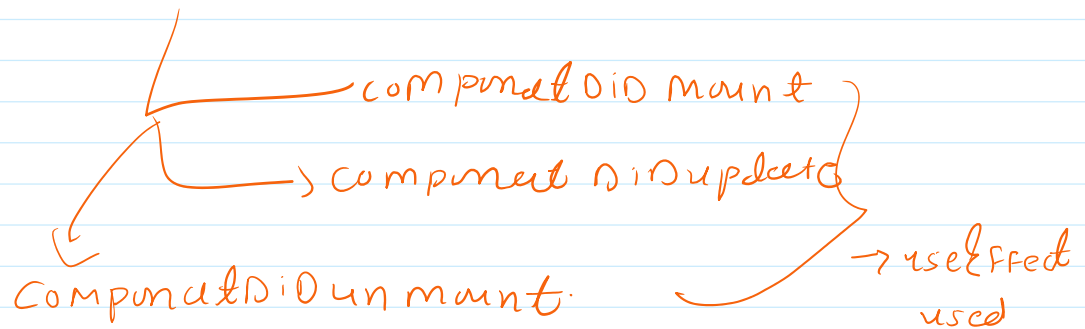
↳ API call, HTTP call,
title change

useEffect ( , [])
normal ahlaz array of dependencies
Bta kya kya use kiye hain use.

mount → render

unmount → render remove

class based life cycle

 component did mount
component did update
component did unmount. → useEffect used

useEffect (() => {
 → Every render
});

useEffect (() => {
 → First render
}, []);

useEffect(c) => {

{ [name] } ;
 ↑
 dependency list

→ first render +
name value change.

useEffect(c) => {
 return c => {
 y → (A) }
 }
 } remove or
 add event
 listener.

* Controlled Components → [Maintain state
inside component]

abcde → onchange → handleChange() { setFormOutput() }

Every input field
state are given

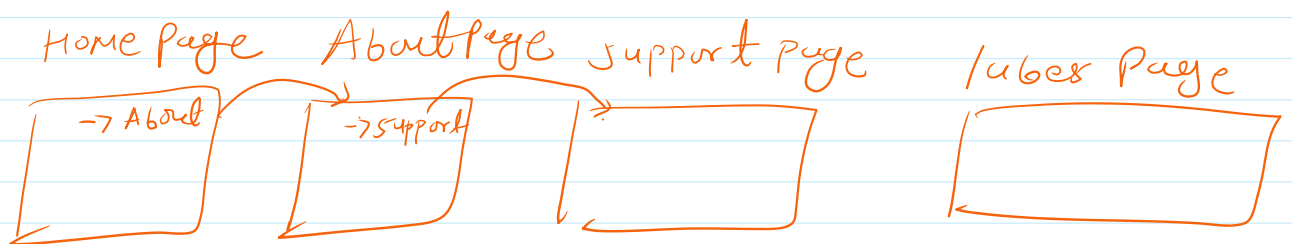
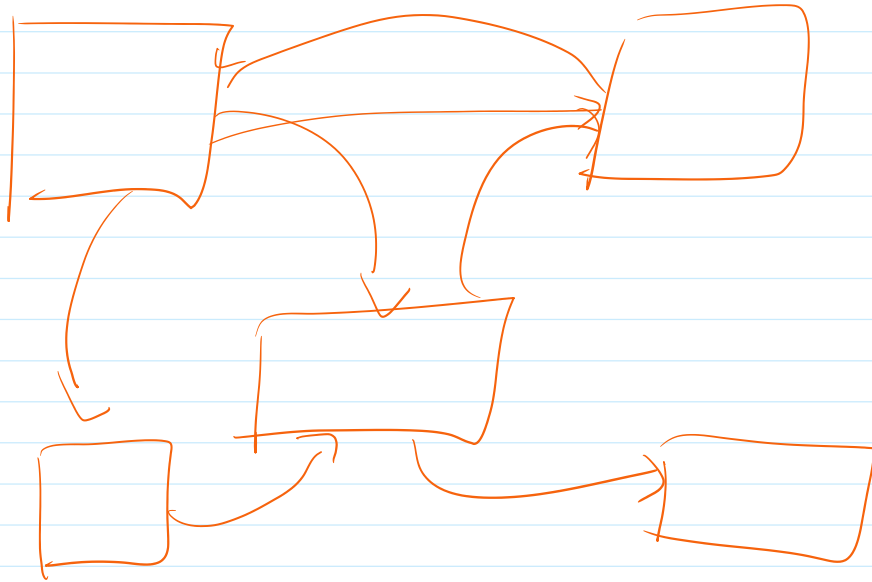
from data update
↓ use state
↓ update
↓ re-render

JSX Code
↓
<input
value =
{f.firstname}

→ React Router :-

↳ navigation without reference.

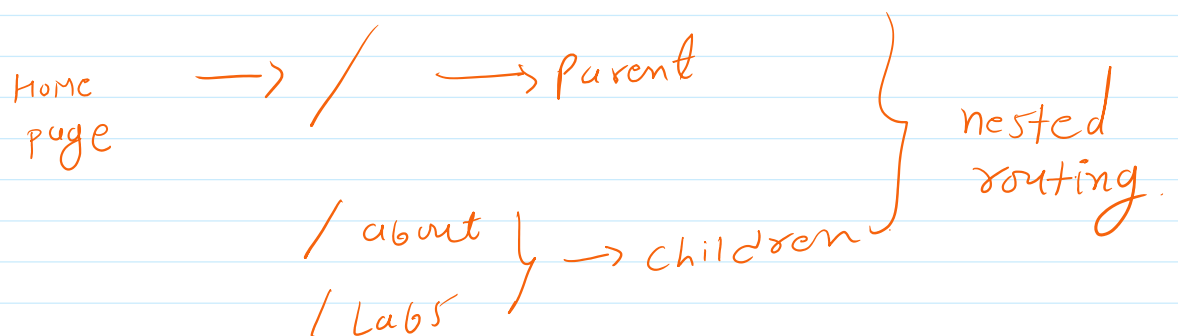
single index.html. Dynamic change
one page to move other page
without reference.



→ Same page per difference components loads,

npm react-router-dom

browser-router tag used.



parent used in = `<Outlet>` ↙ cand of permission

Home page → remove.

↳ create a other file.

that file return a Home page content.

default route ↗ `<Route index element={<Home>}>`

* useNavigate hook → different route navigate using this hook.

Context api :-

rules for used

① Context create.

② Component per used Component used and their children also used that data.
(provided)

③ Consuming

↙
useContext used for context data used.
hook

useSearchParams() hook

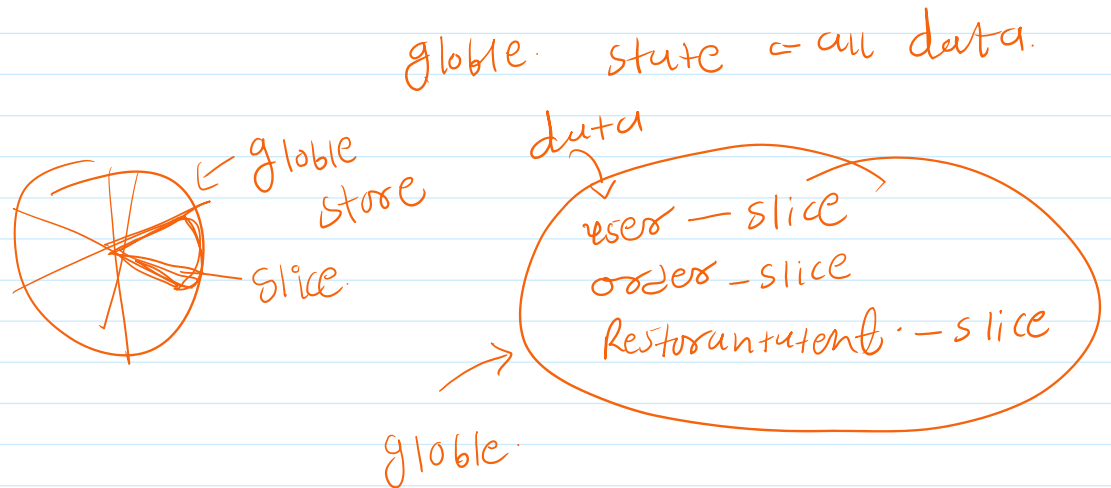
query parameter ↙
update query parameter ↗
`const [searchParameter, setSearchParameter] =`
`useSearchParams()`

useLocation() hook → used the current

location. `pathname` `search` `hash` `state` `key`

useLocation() hook → used the current
↳ path, hash, key, search. location find in component.

redux JS → State Management
↳ need → ↳ slice — 1 type



Function also add a slice.

This called reducers.

need. → { name of slice.
initile state of slice
reducer

hooks



```
const count = useSelector((state) =>  
state.counter.value);  
const dispatch = useDispatch();
```

state access

Function access.