In react, to avoid nesting of lot of <div> tags we may use <Fragment></Fragment>, this can be also written as <></>

In react props and state are 2 special objects.

props object is used for transfer data from parent to child component, or child to parent communication

Props vs state

props	state
props are used to transfer data from parent to child	It is used to store data within component
props are read only	Value of state can be changed within component by using this.setState() method
props are used for parent to child and child to parent communication	state is used for components data, for internal usage

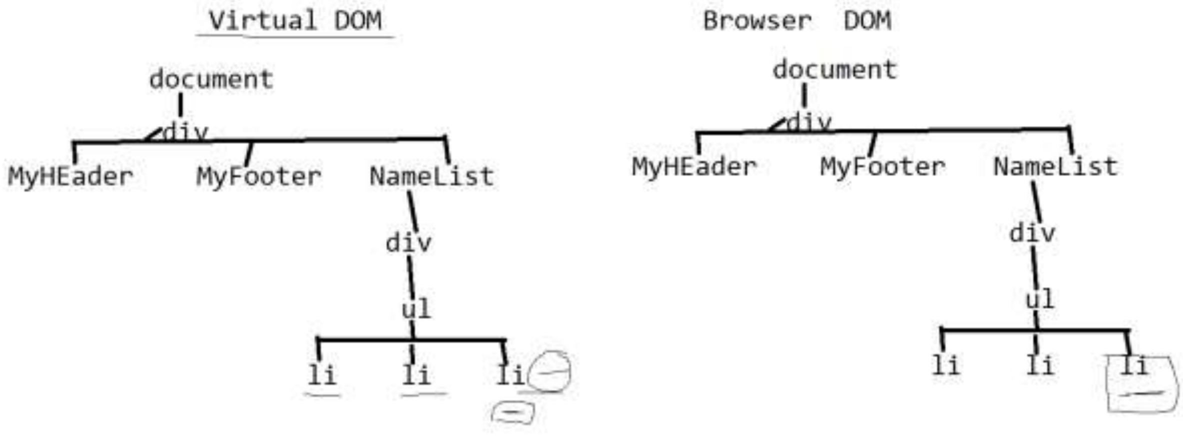
To install bootstrap

npm install bootstrap@4.6

In react if we are using map function then assign key property with unique value,

This helps virtual DOM to search the element in physical DOM (browser DOM), while rerendering.

It improves performance efficiency of the code, because search by using key will be faster.



To transfer data from parent to child

<NameList arr={searcharr} val=”23”></NameList>	Namelist is a component which will receive props object from parent.

	and props object has arr and val, to access searcharr stored in parent component, we can use props.arr in NameList component and props.val in NameList component
{props.arr.map(ob=><li>ob</li>)}	convert every value from the array into li tag or tr tag, the use map function

To use textbox in a form
Name: <input type="text" name="add" id="nm"
value={name}
onChange={changename}></input>
//to assign the value from the textbox to name state variable
const changename=(event)=>{
setname(event.target.value)
}
	when we will change the value of text box, onchange event will occur and the changename function will be called, it will assign the new value to the variable name

In namemanagement , how to add search

1. When you change value of search text box(App.js), then onChange event occurs
2. OnChange will call changesearch function
3. In changesearch function we are using setSearchtxt function to change the value of state searchtxt
4. As soon as searchtxt changes it will call function written in useEffect.

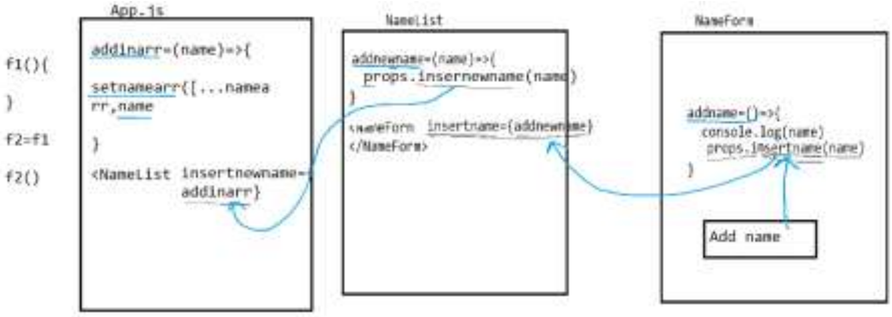
```
            useEffect(()=>{
              if(searchtxt==="")
              {
                setsearcharr([...namearr])
              }else{
                let newarr=namearr.filter(nm=>nm.includes(searchtxt))
                //since the memory of newarr is different so we may pass its
            reference otherwise use spread operator [...newarr]
                setsearcharr(newarr);
              }
  },[searchtxt])
```

5. This useEffect will change searcharr
6. This searcharr we are passing as props to NameList
7. Hence NameList component will be rerendered and displays only names in the searcharr

Add name

1. Assign value and onChange to the text box name
2. Assign onClick event to add name button
3. It will call addname function, in which we are using props.insertname(name);
4. In NameList component(Parent of Nameform), insertname is assigned to addnewname function reference, hence props.insertname(name) in Nameform component will call, addnewname function in NameList component
5. In addnewname of NameList component, we are using props.insertnewname(name)
6. In App component(Parent of NameList), insertnewname is assigned to addinarr function reference, hence props.insertnewname(name) in NameList component will call, addinarr function in App component
7. addinarr function will change namearr, by adding name at the end Setnamearr([...namearr,name])
8. this will call useEffectFunction based on namearr

```
        useEffect(()=>{
          console.log("in searchtxt useEffect")
          if(searchtxt==="")
          {
            setsearcharr([...namearr])
          }else{
            let newarr=namearr.filter(nm=>nm.includes(searchtxt))
            //since the memory of newarr is different so we may pass its
        reference otherwise use spread operator [...newarr]
            setsearcharr(newarr);
          }
  },[searchtxt,namearr])
```

9. this useEffect will change searcharr, which inturn will change arr props of NameList, hence NameList will be rerendered.
10. 

```
f1(){
}
f2=f1
f2()
```

```
addinarr=(name)=>{
setnamearr([...namea
rr,name
}
<NameList insertnewname=
          addinarr}
```

```
addnewname=(name)=>{
  props.insernewname(name)
}
<nameForm  insertname={addnewname}
</NameForm>
```

```
addname=()=>{
  console.log(name)
  props.insertname(name)
}
```

To use routing

1. Install react-router-dom library
2. Install bootstarp@4.6
   npm install react-router-dom bootstrap@4.6