Monday, December 18, 2023 11:18 AM

Parent component and a child component

Communication between Parent and child --- props
Communication between Child and Parent ----LIFTING STATE UP !!!

Ex1 --- Write a Parent component
arr
show the array elements in

It includes a Child component
Enter element of array : Textfield
Button --- clicked --- the element be added to the array
And the array shown in parent component should be updated

1. To **lift state up** --- pass the function of parent to child as a props
child will pass the data to the function and call the function
- in this way child data will be available to parent component

2. To use this.setState or setobj for objects - we must create copy otherwise the setters will not rerender the component
obj.property1 = "newvalue"
this.setState({obj:obj} //NOT RERENDER as address of obj is same
this.setState({obj:{…obj} } //This will RERENDER

arr.push(newelement)
setarr(arr) //NOT RERENDER as address of arr is same
setarr([…arr]) //This will Rerender

3. This.setState or setobj , or setarr = state setter functions are asynchronous
--their execution is delayed
so if you write console.log(see the changed state) ---NOT SEEN

4. When we do not need data rerendering We may not use STATE !!!

Parent component data ---props------>Child component
Child component data to be sent to Parent component
Send parent function as props to Child
Child calls the function and PASSES the data

Scenario --- Two components C1 and C2 want to share data / pass data to
Each other but they are not NESTED as parent and child

Functional component = useRef
Class component = createRef

Ref variable vs State variable

Ref	**State**
Maintains the value of the variable across Renders	**Maintains the value of The variable across**

	renders
When ref changes components is NOT rerendered	**When state changes-component Is RERENDERED**
Ref is also used to access DOM elements directly	**State is not available for this**
to acces value of ref ref.current	

When we want to SHARE data between components

1. Props

2. Redux

REDUX Store ----------Sharing DATA between components !!!

We shared data using Props ---------- Parent ----props -----> Child
--- the components must be in a Parent child composition !!!
What to do when components are not in Parent child composition ?
INDEPENDENT STORAGE = REDUX }} Predictable state container
state is immutable

C1 ----------Redux STORE ----------C2
|
|
|
C3

PROPS DRILLING is problematic if the components is too DEEP
So instead of using PROPS --- REDUX is used !!!

C2---------- STORE (DATA) ----------C1
|
|
|
C3

Store DATA is immutable ------ cannot be changed
To modify the STATE of the data ---REPLACE it

The redux store can be OPERATED by DISPATCHERS that dispatch **"action"**

Observer --Observable pattern ---
All components will act as Observers /Subscribers

npm install redux react-redux @reduxjs/toolkit

1. Create a STORE
REDUCER
---- function that holds the STATE + Operations that can be performed on the state

2. Access the store from C1 and C2
   --- subscribe to the Store !!!

Responsive View / UI

Depending on the viewport size the UI has the ability to change the look

1. The html MUST MUST have

```
<meta name="viewport" content ="width=device-width, initial-scale=1.0">
```

2. Define the change of VIEW in the @media of the CSS
   By specifying different max -min ranges

A huge CSS library = BootStrap

Visit the site = getBootStrap

Routing --- create routes on a SPA to get a feeling that multiple Pages are accessed
At a time we see only one route

Step1 --- npm install react-router react-router-dom
Step2 --- WRAP in <BrowserRouter >
Step 3 ---- Define routes
Step 4 --- provide <Link> ---- <Home> to routes
Step 5 ---- define <Outlet> == the component will be rendered here
Step 6 ---The path of the <home> component should enclose other paths
The menu always remains on the top

Deeper --

1. On button click navigate to another route
2. Pass data between routes