Ep-11 Data is the New Oil

Date:

↳ Higher Order Component

- Higher order component is a function that takes the component as input and return another component as output
- Normal Javascript Function
- Takes component (already built) as input and enhances it or add extra feature into it, and returns it back

- **Promoted Label Feature Using Higher Order Component**

e.g.] export const withPromotedLabel = (Restaurant) => {

```
return (props) => {
    return (
      <div>
          <label> Promoted </label>
        <RestaurantCard {...props} />
        </div>
    );
  };
};
```

=> const RestaurantWithPromotedLabel = withPromotedLabel(RestaurantCard);

{ res.info.promoted ? (RestaurantWithPromoted
Label resData = {resList}/>)
: (<RestaurantCard resData = {resList}/>)
}

↳ Uncontrolled & Controlled Components

Uncontrolled components manages their own state and control their behaviours with the help of DOM

Uncontrolled components do not relay on react state and are handled by DOM

Controlled component are the components whose state and behaviours are managed by another React component (parent component) using states.

↳ Lifting the state up in react

Managing the state across different components.

It is common for multiple components to need access to the same piece of data or state

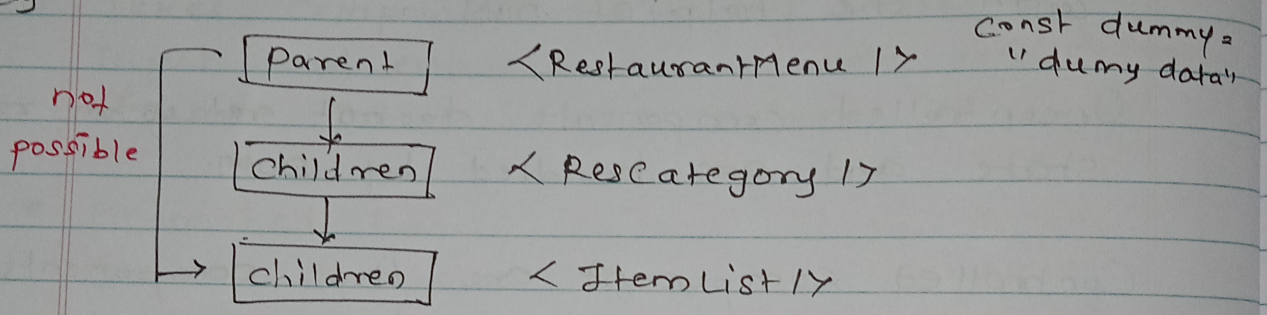
Instead of duplicating state across multiple components. React allows to "lift" the state up to common an cestor

↳ props drilling

If we know how to manage data layer properly, react application is very easy to build.

There is one way data flow
e.g parent → children → to their childrens

e.g



He can't pass data directly to the childrens of children.

To pass data, we need intermediate parent

But in real world application, there is large numbers of nested component is there

So it is so much complicated to handle the operation.

This problem is known as props drilling.

Date:

→ React Context

To pass or access data anywhere in the app, without relying on anyone, we use React context

To avoid problem like props drilling we use React context

If some information we need all across the app, can be access anywhere in our app, for that data we hold it inside a context that context can be used anywhere in app.

Building central store, so we don't need to drill down the props.

It is kind of central global object

e.g
```
const UserContext = createContext({
    loggedInUser : "Default User"
});

export default UserContext;
```

- context created by using createContext() which is given by react

eg Using context
```
const data = useContext(UserContext);
```

- useContext() is a hook given by react to use created context

- there is multiple context we can create in our app.

- In class component there is no hooks to use create context

eg import UserContext from '/path';

```
<div>
    <UserContext.Consumer>
        {
        (data) =>
    <h1> {data.loggedInUser} </h1>
        }
    </UserContext.Consumer>
```

- 2 ways to use create Context

1) using useContext() hook
2) using <contextName.Consumer/>

→ Context Provider

eg const [userName, setUserName] = useState();

```
const data = {
    name: "Akshay Saini"
};
```

Imp Interview Que: What is difference between context & redux?

```
useEffect(() => {
        setUserName(data.name);
}, []);

return (
    <UserContext.Provider value={{ loggedIn
            User : userName }}>
        <div>
            <Header />
            <Outlet />
        </div>
    </UserContext.Provider>
```

↳ modifying UserContext

```
<input  value={loggedInUser}
    onChange={(e) => setUserName(e.target
                value)}
    />
```

App.js

```
<UserContext.Provider value={{ loggedInUser:
        userName , setUserName }}>
    </UserContext.Provider>
```

↳ Data managment library or state managment library

- Redux
- Simplify Context Concept (replacment)