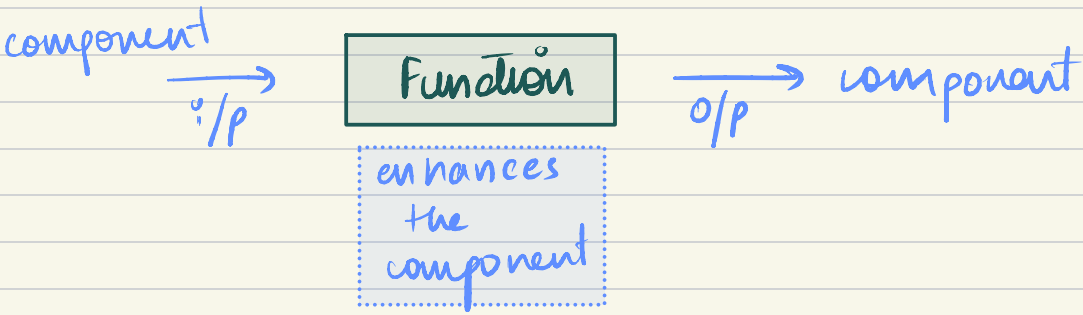
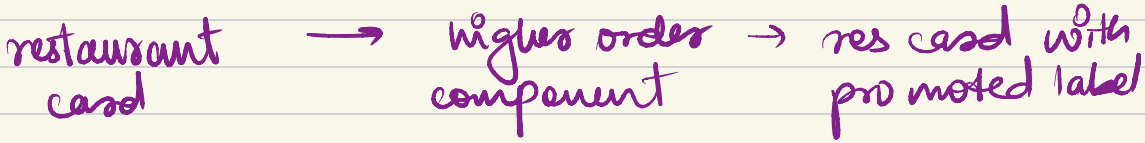

data is
the new oil.

#Higher order component



Example of promoted restaurants

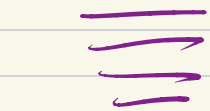
↳ for specific restaurant, you modify specific ones



const HigherOrderComponent = () => {

return () => {

return (



)

}

}

returns a component

2 layers: React

* most imp to manage data layer.

data layer

state, props, local vars

↓ powers

UI layer

jsx code

fixing restaurant menu, as api changed

→ I am getting the resId printed using useParams!

→ need to fix json object retrieval

① Text : name

Pizza hut

data.cards[0].card.card.text

② data.cards[2].card.card.info.

→ areaName , city
→ avgRating , totalRatingsString.
→ cloudinary ImageId
→ costForTwoMessage.
→ cuisines
→ "id"

The menu card is customised.

#Accordian toggle feature

① make a state variable

```
const [ ] = useState( );
```

false → default

showItems, setShowItems.

② if showItems → true

only then show ItemList

else dont show.

③ header div → on Click event.

when header is clicked, the accordian should toggle.

handle click event should do the following :-

if setShowItems = true, make it false

if setShowItems = false, make it true.

Restaurant menu → controls all categories

- should have the power to show or hide items
- Lifting the state up
  - if one category is open, others closed
  - the parent (RestaurantMenu) should have control over restaurant categories

Controlled and Uncontrolled Components

Uncontrolled components- do not rely on React state and are handled by DOM.

→ refs in React forms

- access values using refs

Controlled components- state and behaviour is managed by React component.

- controlled by the parent component
- current value is accepted as props.

can a child modify parent's state variable?

- Problem — **props drilling**
- soln → context, which is maintained throughout the appln → global

create in utils.

→ create context.

advance concept.

Lifting the state up.

→ sometimes we need to lift state up so that the children can be controlled

(note)

API call inside use effect()

PROPS DRILLING

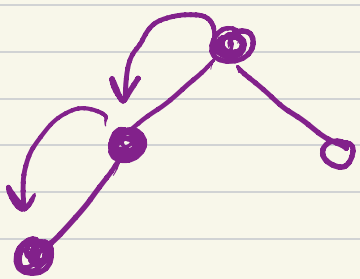
problem to be avoided

- many components with various hierarchy
- big tree structure with nested components.

parent ⟶ child

one way data stream

- data flow is one way, flows from parent to child



data has to go through all intermediate nodes, to reach the children nodes

- This is an issue in React, we don't want to pass data down to such deep levels.

Solution → React context. like a global place, which is maintained throughout the appln.

can be accessed anywhere from our appln

Example: logged in user.

- context created inside utils → seperate file.
- utility function → Object
  - → createContext()
- how to access the context within the application?
  - → useContext()
    - ↓
    - user Context
      - context that you just made using createContext

#
There can be multiple contexts within a single application.

That's why you pass in an arg inside useContext, to use a particular context

only the data that needs to be used throughout the application, needs to be in context.

Class based components do not have hooks?

how to access context.

```
<UserContext.Consumer>
  {(data) => console.log(data)}
</UserContext.Consumer>
```

this data is the context data

---

how to change the value of context?

wrap your whole app inside

```
<UserContext.Provider value={{loginUser: UserName}}>
  </Header>
  </Body>
</UserContext.Provider>
```

on the ground level.
App.js

⇒ can you name nested contexts?

whole app has some default value and the Header only has the specific content

as the inner context has overridden the default

---

you can pass functions also in the contextProvider