Miracle
Pg.No :
Date: / /20

Coursera - Advanced React

* Transforming lists in Javascript

→ By .map() method.

e.g. I have data.js file that has an array of objects of a dissert menu.

```
const data = [
    {
      id: "1"
      title: "Ice Cream",
      discription: " ",
      price: " ",
      image: " "
    }
    {
      .
      .
      .
]
```

Now, I only want the title, discription and price to show on my page. Do it using .map() method.

```
const desserts = data.map(dissert => {
    return {
      content: `${dissert.title} - ${dessert.
                                    discription}
      price: {dissert.price}
    }
})
```

→ The map() method returns a new array.

→ The map() method is useful for handling third party data.

→ The map() method is a transformation operation.

✡ Render a simple list component

```
const list = data.map (dessert => {
    return <li> {dessert.title} </li>
})
```

In code:
```
<ul>
    {list}
</ul>
```

✡ Keys

→ What is the need of keys?

e.g Add element to list

```
<ul>
    <li> Beer </li>
    <li> Wine </li>
</ul>
```

If we add a new item at the end of the list, React will work well. It will

match the two Beer trees, two Wine trees and add the third new Cider.

But what if we want to add a new item at the beginning?

React will show worst performance because React will mutate every child instead of realizing it can keep the beer and wine intact. This inefficiency can be a problem.

To solve this issue, react supports key attribute.

→ What are keys?

Keys are identifiers that help React to determine which items have changed or added or are removed.

Also instruct treatment of elements when updates happen.

Use stable identifiers that is always unique among its siblings.

→ Prevents key collisions

① Math.random() : Does not preserve internal state of list items

Why?

Miracle
Pg.No :
Date: / /20

Because when re-rendering occurs, those keys will be different, resulting in React having to recreate your list from scratch.

(2) Item Index : Indexes are not recommended for keys if the order of items may change, for eg. in cases where your list has sorting capabilities or users can either add or remove items.

Why?
Negatively affect performance
User interface glitches

(3) Unique ID : e.g. UUID()