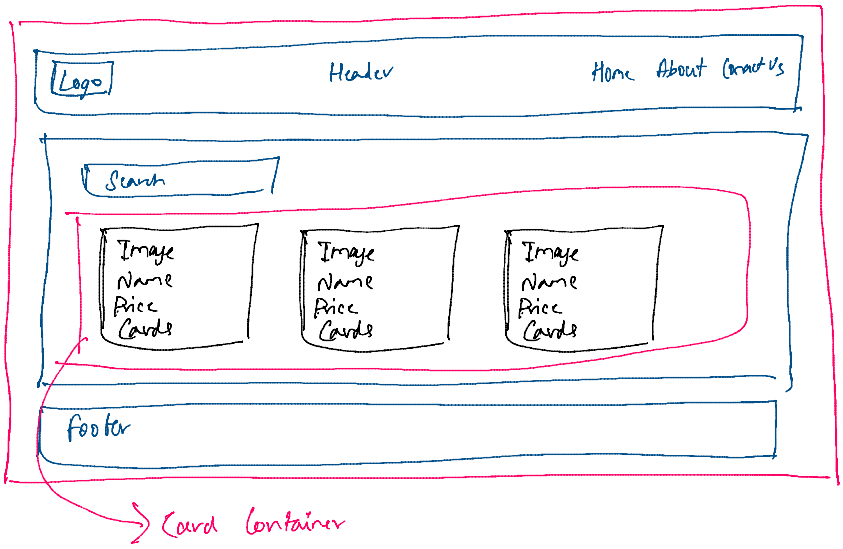
4 - Talk is Cheap, Show me the Code

Food Delivery App



App layout

```
const AppLayout = () => {
  return <div class="app">
    // Header
    // Body
    // Footer
  </div>;
};
```

This component to build.

Header Component Code :

```
const Header = () => {
  return (
    <div className="header">
      <div className="logo-container">
        <img
          className="logo"
          src="https://www.logodesign.net/logo/smoking-burger-with-lettuce-3624ld.png"
        />
      </div>
      <div className="nav-items">
        <ul>
          <li>Home</li>
          <li>About Us</li>
          <li>Contact Us</li>
          <li>Cart</li>
        </ul>
      </div>
    </div>
  );
};
```

App Layout Code

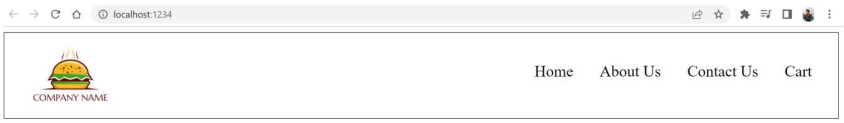
```
const AppLayout = () => {
  return (
    <div className="app">
      <Header /> // Header is called here (this is known as composition component)
    </div>
  );
};
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<AppLayout />);
```

App Layout is rendered.

CSS file

```css
.header{
    display: flex;
    justify-content: space-between;
    border: 1px solid black;
}

.logo{
    width: 200px;
}

.nav-items{
    padding: 0px 20px;
}
.nav-items > ul {
    font-size: 24px;
    display: flex;
    list-style-type: none;
}

.nav-items >ul > li {
    padding: 10px;
    margin: 10px;
}
```

Output



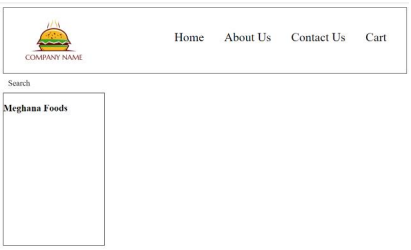
Building Restaurant Card :

```jsx
const Body = () => {
  return (
    <div className="body">
      <div className="Search">Search</div>
      <div className="res-container">
        // Restaurant Card for this seperate component
      </div>
    </div>
  );
};
```

```jsx
const RestaurantCard = () => {
  return (
    <div className="res-card">
      <h3>Meghana Foods</h3>
    </div>
  );
};

const Body = () => {
  return (
    <div className="body">
      <div className="Search">Search</div>
      <div className="res-container">
        <RestaurantCard />
      </div>
    </div>
  );
};
```

CSS file for the same

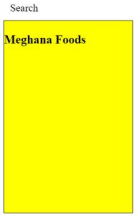
```css
.res-card{
    width: 200px;
    height: 300px;
    border: 1px solid black;

}

.search{
    padding: 10px;
}
```



Inline style in React :

+ It's given using javascript object.

```jsx
const styleCard = {
  backgroundColor: "yellow",
};

const RestaurantCard = () => {
  return (
    <div className="res-card" style={styleCard}>
      <h3>Meghana Foods</h3>
    </div>
  );
};
```



But this is not recommed way

Other way of doing

```jsx
const RestaurantCard = () => {
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "yellow",
      }}
    >
      <h3>Meghana Foods</h3>
    </div>
  );
};
```

1st bracket is javascript

2nd bracket is object

Restaurant Card Building

```jsx
const RestaurantCard = () => {
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img
        className="res-logo"
        alt="res-logo"
        src="https://res.cloudinary.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_508,h_320,c_fill/oycavthjdkphqevhzk7t"
      />
      <h3>Meghana Foods</h3>
      <h4>Biryani, North Indian, Asian</h4>
      <h4>4.4 stars</h4>
      <h4>38 mins</h4>
    </div>
  );
};
```

CSS file

```css
.res-logo{
    width: 100%;
}
```

Output

Meghana Foods

Biryani, North Indian, Asian

4.4 stars

38 mins

Completing the CSS:

```css
.res-card:hover{
    cursor: pointer;
    border: 1px solid black;
}
```

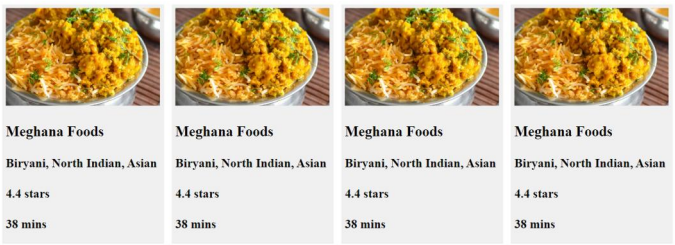
But we can have many restaurant cards,

```jsx
const Body = () => {
  return (
    <div className="body">
      <div className="search">Search</div>
      <div className="res-container">
        <RestaurantCard />
        <RestaurantCard />
        <RestaurantCard />
        <RestaurantCard />
      </div>
    </div>
  );
};
```

CSS file:

```css
.res-container{
    display: flex;
    flex-wrap: wrap;
}
```

Output:



Here the data is hard coded.

To make the card as dynamic,

like 2nd card as kfc, etc.

Here comes the new concept,

* Props :- It means properties

  - It can be passed to the component.

- It can be passed to the component.

- To pass the data as dynamic we use props.

- Props is normal argument. (it's like passing a argument to the function)

```jsx
const Body = () => {
  return (
    <div className="body">
      <div className="search">Search</div>
      <div className="res-container">
        <RestaurantCard
          resName="Meghana Foods"
          cuisine="Briyani, North Indian, Asian"
        />
        <RestaurantCard resName="KFC" cuisine="Burger, Fast Food" />
      </div>
    </div>
  );
};
```

props

props (passing props to the component)

basically

passing argument to the function.

- React take the props and it will pass to it. (It's an object over there)

```jsx
const RestaurantCard = (props) => {
  console.log(props);
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img
        className="res-logo"
        alt="res-logo"
        src="https://res.cloudinary.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_500,h_320,c_fill/oycavthjdkphqevhzk7t"
      />
      <h3>Meghana Foods</h3>
      <h4>Biryani, North Indian, Asian</h4>
      <h4>4.4 stars</h4>
      <h4>38 mins</h4>
    </div>
  );
};
```

Object

to make this card dynamic read the props. (this props is object) So we can pass it inside of {}

App.js:49

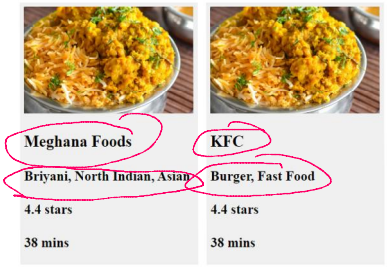
```
{resName: 'Meghana Foods', cuisine: 'Briyani, North Indian, Asian'}
  cuisine: "Briyani, North Indian, Asian"
  resName: "Meghana Foods"
  [[Prototype]]: Object
```

App.js:49

```
{resName: 'KFC', cuisine: 'Burger, Fast Food'}
  cuisine: "Burger, Fast Food"
  resName: "KFC"
  [[Prototype]]: Object
```

```jsx
const RestaurantCard = (props) => {
  console.log(props);
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img
        className="res-logo"
        alt="res-logo"
        src="https://res.cloudinary.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_508,h_320,c_fill/oycavthjdkphqevhzk7t"
      />
      <h3>{props.restName}</h3>
      <h4>{props.cuisine}</h4>
      <h4>4.4 stars</h4>
      <h4>38 mins</h4>
    </div>
  );
};
```

props is passed



Some developer they destructured on the fly:

```jsx
const RestaurantCard = ({ resName, cuisine }) => {
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img
        className="res-logo"
        alt="res-logo"
        src="https://res.cloudinary.com/swiggy/image/u
        oycavthjdkphqevhzk7t"
      />
      <h3>{resName}</h3>
      <h4>{cuisine}</h4>
      <h4>4.4 stars</h4>
      <h4>38 mins</h4>
    </div>
  );
};
```

this is javascript

More readable code

```jsx
const RestaurantCard = (props) => {
  const { resName, cuisine } = props;
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img
        className="res-logo"
        alt="res-logo"
        src="https://res.cloudinary.com
        oycavthjdkphqevhzk7t"
      />
      <h3>{resName}</h3>
      <h4>{cuisine}</h4>
      <h4>4.4 stars</h4>
      <h4>38 mins</h4>
    </div>
  );
};
```

One more way of passing the data from the backend?

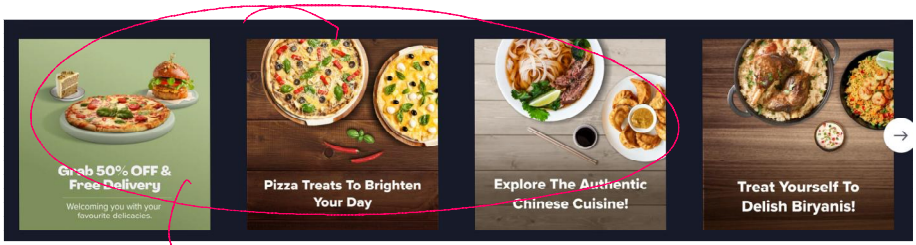
- It comes in form of json.

Config Driven UI: (For dynamic data)

What is config driven UI?

- For e.g.

When you open Swiggy, you can see some offer based on your location, now suppose there is other offer in another location, then the carousel will be different for different location (offers).



this is different for every location

- You won't make different UI for different location.

- So we use config driven UI.

- Basically controlling the UI, how the UI looks like using data (config)

- The config comes from backend.

Data:

```
// data
const resObject = [
  {
    name: "Burger King",
    image:
      "https://media.istockphoto.com/id/1206323282/photo/juicy-hamburger-on-white-background.jpg?s=612x612&w=0&k=20&c=K0DxyiChLwewXcCy8aLjjOqkc8QXPgQMAW-rwRCzqG4=",
    cusines: ["Burger", "American"],
    rating: "4.2",
    deliverTime: 36,
  },
  {
    name: "KFC",
    image:
      "https://orderserv-kfc-assets.yum.com/15895bb59f7b4bb588ee933f3cd5344a/images/items/xl/D-PR00000974.jpg?ver=27.36",
    cusines: ["KFC Burger"],
    rating: "4.5",
    deliverTime: 23,
  },
];
```

Restaurant Card

```
const RestaurantCard = (props) => {
  const { resData } = props;
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img className="res-logo" alt="res-logo" src={resData.image} />
      <h3>{resData.name}</h3>
      <h4>{resData.cusines.join(",")}</h4>
      <h4>{resData.rating} stars</h4>
      <h4>{resData.deliverTime}</h4>
    </div>
  );
};
```

called here

Body

```
const Body = () => {
  return (
    <div className="body">
      <div className="search">Search</div>
      <div className="res-container">
        <RestaurantCard resData={resObject[0]} />
        <RestaurantCard resData={resObject[1]} />
        <RestaurantCard resData={resObject[0]} />
      </div>
    </div>
  );
};
```

Output:

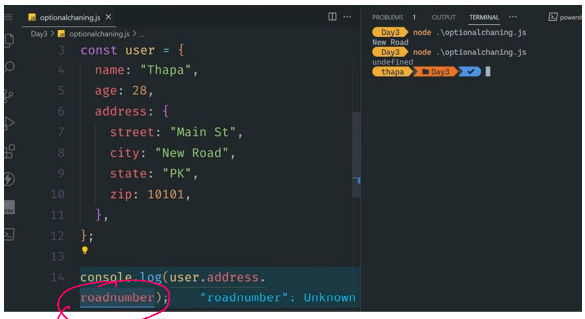
Javascript:

Read about optional chaining &

Javascript:

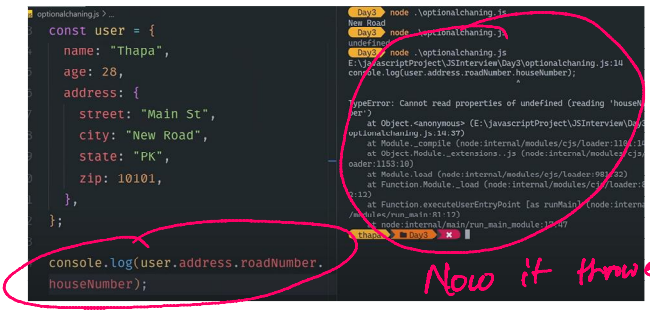
Read about optional chaining :

It's used to read the property of an object, without having to check whether the object or array is null or undefined.



Here road number is not present and still it didn't throw error but gave undefined.

Now suppose considered road number is present and accessing house number foo.



Now it throwed error.

But we don't need this error as it stop executing.

So we use optional chaining just add '?'

if it's there print housenumber
else show undefined

Now you can see no error.

You add '?' after address too.

Now let's follow coding standard:

```jsx
const RestaurantCard = (props) => {
  const { resData } = props;
  const { image, name, cusines, rating, deliverTime } = resData;
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img className="res-logo" alt="res-logo" src={image} />
      <h3>{name}</h3>
      <h4>{Array.isArray(cusines) ? cusines.join(",") : ""}</h4>
      <h4>{rating} stars</h4>
      <h4>{deliverTime}</h4>
    </div>
  );
};
```

But if now there is 1000 of data, so we will loop over the array using Map.

```jsx
const Body = () => {
  return (
    <div className="body">
      <div className="search">Search</div>
      <div className="res-container">
        <RestaurantCard resData={resObject[0]} />
        <RestaurantCard resData={resObject[1]} />
        <RestaurantCard resData={resObject[0]} />
        <RestaurantCard resData={resObject[1]} />
        <RestaurantCard resData={resObject[0]} />
        <RestaurantCard resData={resObject[1]} />
        <RestaurantCard resData={resObject[0]} />
      </div>
    </div>
  );
};
```

```jsx
// data
const resObject = [
  {
    name: "Burger King",
    image:
      "https://media.istockphoto.com/id/1206323282/photo/juicy-hamburger-on-white-background.jpg?s=612x612&w=0&k=20&c=K0DxyiChLwewXcCy8aLjjOqkc8QXPgQMAW-vwRCzqG4=",
    cusines: ["Burger", "American"],
    rating: "4.2",
    deliverTime: 36,
  },
  {
    name: "KFC",
    image:
      "https://orderserv-kfc-assets.yum.com/15895bb59f7b4bb588ee933f8cd5344a/images/items/xl/D-PR00000974.jpg?ver=27.36",
    cusines: ["KFC Burger"],
    rating: "4.5",
    deliverTime: 23,
  },
];
```

```jsx
const RestaurantCard = (props) => {
  const { resData } = props;
  const { image, name, cusines, rating, deliverTime } = resData;
  return (
    <div
      className="res-card"
      style={{
        backgroundColor: "#f0f0f0",
      }}
    >
      <img className="res-logo" alt="res-logo" src={image} />
      <h3>{name}</h3>
      <h4>{Array.isArray(cusines) ? cusines.join(",") : ""}</h4>
      <h4>{rating} stars</h4>
      <h4>{deliverTime}</h4>
    </div>
  );
};
```

```jsx
const Body = () => {
  return (
    <div className="body">
      <div className="search">Search</div>
      <div className="res-container">
        {resObject.map((restaurant) => (
          <RestaurantCard resData={restaurant} />
        ))}
      </div>
    </div>
  );
};
```

this is how you call object

Important.

Now we have warning,

```
▸Warning: Each child in a list should have a unique "key" prop.   index.js:1

Check the render method of `Body`. See https://reactjs.org/link/warning-keys for more information.
    at RestaurantCard (http://localhost:1234/index.7826abd7.js:3025:13)
    at Body
    at div
    at AppLayout
```

Each of these should be uniquely represented

```
68 const resObject = [
69   {
70     name: "Burger King",
71     id: "01",
72     image:
73       "https://media.istockphoto.com/id/1206323282/photo/
        juicy-hamburger-on-white-background.jpg?s=612x612&w=0&k=20&
        c=K0DxyiChLwewXcCy8aLjjOqkc8QXPgQMAW-vwRCzqG4=",
74     cusines: ["Burger", "American"],
75     rating: "4.2",
76     deliverTime: 36,
77   },
78   {
79     name: "KFC",
80     id: "02",
81     image:
82       "https://orderserv-kfc-assets.yum.com/
        15895bb59f7b4bb588ee933f8cd5344a/images/items/xl/
        D-PR00000974.jpg?ver=27.36",
83     cusines: ["KFC Burger"],
84     rating: "4.5",
85     deliverTime: 23,
86   },
87 ];
```

Give unique Id to the data.

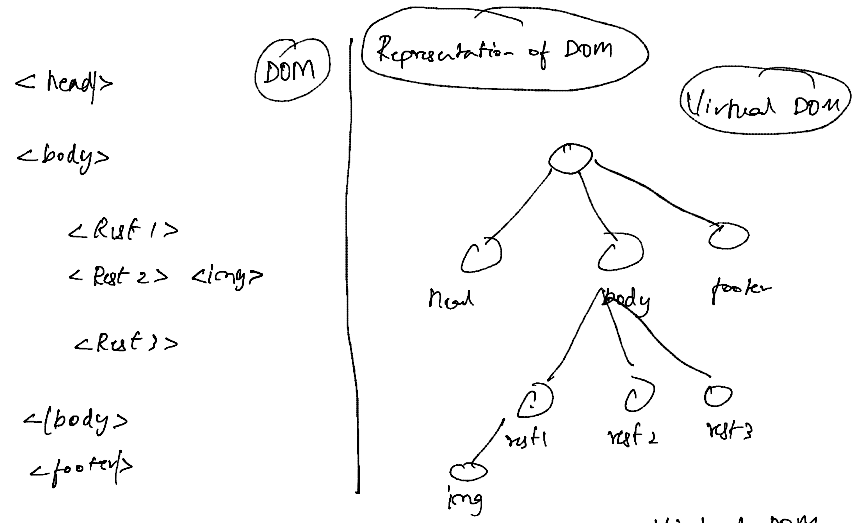
and now give the key to it.

```
89 const Body = () => {
90   return (
91     <div className="body">
92       <div className="search">Search</div>
93       <div className="res-container">
94         {resObject.map((restaurant) => (
95           <RestaurantCard key={restaurant.id} resData=
             {restaurant} />
96         ))}
97       </div>
98     </div>
99   );
100 };
```

Now no error, so when ever you create a map always create a key.

why key? (concept comes Virtual DOM)

what is virtual DOM?



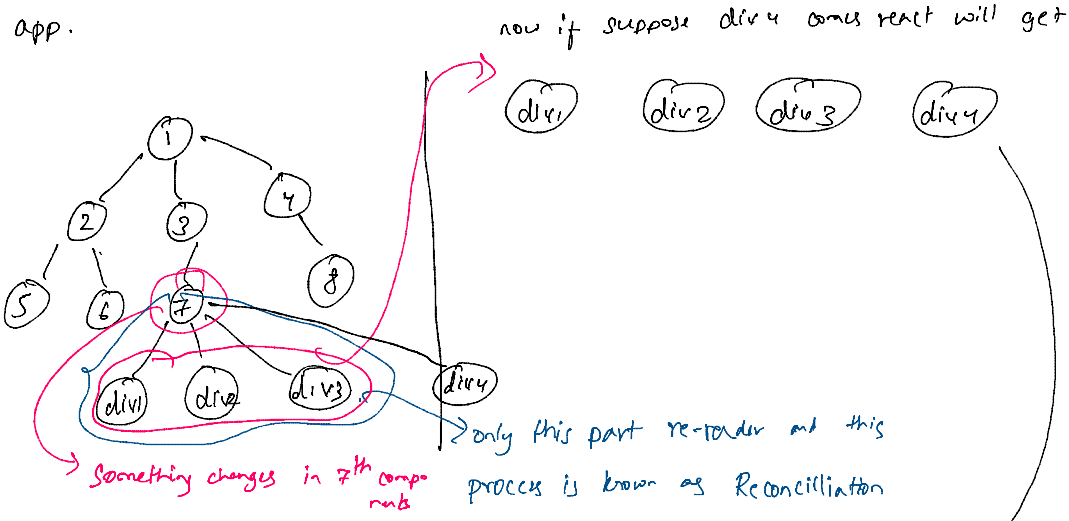
we keep representation of DOM is known as Virtual DOM.

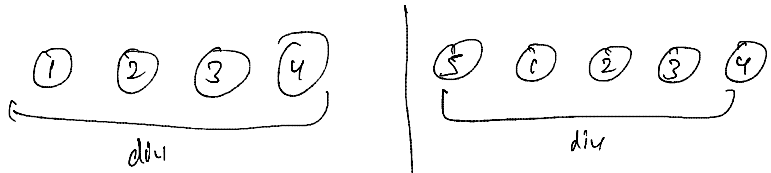
why do we need virtual DOM?

- It's need for Reconcillintion for React.

- It's a algorithm that react uses to diff one tree from another and it determine what's need to change in UI and what doesn't need for UI.

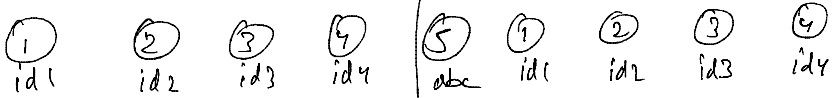
- It's find out what needed to be updated and update that portion of it, it doesn't re-render whole app.





Now if (5) comes in 1st positio react will not know which div It's, it know it has 5 div and it re-render everything.

Now if we use something known as (key) which is unique.



Now react will known that already 1, 2, 3, 4 is present, so now react will reordered which is added new.

Read more about Virtual DOM...

Why react more fast? (Interview) (2:45) time stamp.

- Explain about Virtual DOM — as written above

then talk about reconciliation.

Read more on

- Reconciliation https://medium.com/javarevisited/react-reconciliation-algorithm-86e3e22c1b40#:~:text=Reconciliation%20is%20the%20process%20by%20which%20React%20updates%20the%20UI,Model)%20to%20update%20the%20UI.
- React Fibre. (it's a new reconciliation engine) React 16.
  - ↳ Used to render a system faster and smoother

The algorithm react uses to diff one tree with another to determine which parts need to be changed.

↳ Interview Answer

- React doesn't recommend using index for keys if the order of items may change.

We don't recommend using indexes for keys if the order of items may change. This can negatively impact performance and may cause issues with component state. Check out Robin Pokorny's article for an in-depth explanation on the negative impacts of using an index as a key. If you choose not to assign an explicit key to list items then React will default to using indexes as keys.

If no key (not acceptable) <<<< index key (OK way if no unique) <<<< unique key (best practice)