Talk is Cheap, Show me the Code!(Namaste-React)

Please make sure to follow along with the whole” Namaste ***React"*** series, starting from Episode-1 and continuing through each subsequent episode. The notes are designed to provide detailed explanations of each concept along with examples to ensure thorough understanding. Each episode builds upon the knowledge gained from the previous ones, so starting from the beginning will give you a comprehensive understanding of React development.

I've got a quick tip for you. To get the most out of these notes, it's a good idea to watch **Episode-3** first. Understanding what ***”Akshay”*** shares in the video will make these notes way easier to understand.

So far, here's what we've learned in the previous episode

- We learned what’s JSX.
- We explored what is transpilation and Babel.
- We got to know the difference between Class Based Componentsand Functional Components.
- We also explored the concept of bundlers.
- We learned what is component composition.

Part-1

In this episode, we will start actual coding by starting a new project. Our app is going to a Food Ordering App.

Planning for the UI

Before we start coding, plan things out. Planning will make things easier to understand. We should know exactly what to build:

- Name the App
- UI Structure

```
Header
 - Logo
 - Nav Items
```

```
Body
 - Search
 - Restaurant Container
   - Restaurant Card
     - Dish Name
     - Image
     - Restaurant Name
     - Rating
     - Cuisines
     - Time to Deliver

Footer
 - Copyright
 - Links
 - Address
 - Contact
```

- Keep that as a reference and start coding the app.

Let's start coding!

It is recommended that you code on your own but for some examples, we have mentioned some pieces for you along with the component name.

Main components = AppLayout

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

Header Component

```
const Header = () => {
 return(
  <div className="header">
    <div className="logo-container">
      <img className="logo" src="url" />
    </div>
    <div className="nav-items">
      <ul>
        <li>Home</li>
        <li>About Us</li>
        <li>Contact Us</li>
        <li>Cart</li>
      </ul>
    </div>
  </div>
 )
 }
```

Inline Styling

Writing the CSS along with the element in the same file. It is not recommended to use inline styling. So you should avoid writing it.

```
<div
className="red-card"
style={{ backgroundColor: "#f0f0f0" }}
>

 <h3> Meghana Foods </h3>
</div>
```

In '*style={{ backgroundColor: "#f0f0f0" }}*', first bracket is to tell that whatever is coming next will be JavaScript and the second bracket is for JavaScript object

Internal Styling

or you can store the CSS in a variable and then use it

```
const styleCard = { backgroundColor: "#f0f0f0" };

<div
className="red-card"
style={styleCard}
>
 <h3> Meghana Foods </h3>
</div>
```

Part-2

Introducing Props.

Short form for properties. To dynamically send data to a component we use props. Passing a prop to a function is like passing an argument to a function.

Passing Props to a Component

Example,

```
 RestaurantCard
  resName="Meghana Foods"
  cuisine="Biryani, North Indian"
/> >
```

'*resName*' and '*cuisine*' a props and this is prop passing to a component.

Receiving props in the Component Props will be

wrapped and send in JavaScript object

Example,

```
const RestaurantCard = (props) => {
  return(
    <div>{props.resName}</div>
  )
}
```

Destructuring Props

Example,

```
const RestaurantCard = ({resName, cuisine}) => {
  return(
    <div>{resName}</div>
  )
}
```

Config Driven UI.

It is a user Interface that is built and configured using a declaration configuration file or data structure, rather than being hardcoded.

Config is the data coming from the api which keeps on changing according to different factors like user, location, etc.

To add something in the elements of array

Example, Adding “,” after every value

```
resData.data.cuisine.join(", ")
```

Good Practices

Destructuring props-

Optional Chaining

Example,

```
const {name, avgRating, cuisine} = resData?.data;
```

Repeating ourselves (repeating a piece of code again and again)-

---

Dynamic Component listing using JS map() function to loop over an array and pass the data to component once instead of hard coding the same component with different props values

Avoid ✕

```
<RestaurantCard
  resName="Meghana Foods"
/>
<RestaurantCard
  resName="KFC"
/>
<RestaurantCard
  resName="McDonald's"
/>
<RestaurantCard
  resName="Dominos"
/>
```

Follow ✓

```
const resList = [
 {
  resName: "Meghana Foods"
 },
 {
  resName: "KFC"
 },
 {
  resName: "McDonald's"
 },
 {
  resName: "Dominos"
 }
]

const Body = () => {
 return(
  <div>
   {resList.map((restaurant) => (
```

```
        <RestaurantCard resData={restaurant} />
      ))}
    </div>
  )
}
```

Unique Key id while using map-

Each item in the list must be uniquely identified

Why? When we have components at same level and if a new component comes on the first without ID, DOM is going to re-render all the components again. As DOM can't identify where to place it.

But if we give each of them a unique ID then react knows where to put that component according to the ID. It is a good optimization and performance thing. Note* Never use index as keys in map. It is not recommended.

```
const Body = () => {
 return(
  <div>
    {resList.map((restaurant) => (
       <RestaurantCard key={restaurant.id} resData={restaurant} /
    ))}
  </div>
 )
}
```

Self-Notes-

Chapter 04 - Talk is cheap, show me the code!

As a senior engineer is what you should do is you planning ?

we create any app first we will do the planning comes a ui design a layout, a wireframe, a mock . How your app should look like.so first of all we will do that and then we will be writing code.

CSS-

```
// InLine CSS in reactjs
const styleCard ={
    backgroundColor: "yellow",
}

const ResturantCard = () => {
    return (
        <div className='res-card' style={styleCard}>
            <h3>Meghana Food</h3>
        </div>
    )
}
```

```
// InLine CSS in reactjs
const ResturantCard = () => {
    return (
        <div className='res-card' style={{  backgroundColor: "yellow"}}>
            <h3>Meghana Food</h3>
        </div>
    )
}
```

,

How to make the cards dynamically-

Suppose my first card is megnafood how to make second card kfc? Coz we are reusing the cards. How we cand do that?, I want to dynamically pass data to some component. How we can do that?

We can do that by using props(properties)

Props are just normal argument to a function. Passing a props to a component is just like passing an argument to a function.

```jsx
const ResturantCard = (props) => {
    console.log(props)
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src="https://picsum.photos/200/100" />
            <h3>{props.resName}</h3>
            <h4>{props.cusion}</h4>
            <h4>4.4</h4>
            <h4>38 min</h4>
        </div>
    )
}
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName="Meghana Food" cusion="Briyani" />
                <ResturantCard resName="KFC" cusion="Chicken" />
            </div>
        </div>
    )
}
```

g/link/react-devtools

App.js:65

▼ {resName: 'Meghana Food', cusion: 'Briyani'} i
  cusion: "Briyani"
  resName: "Meghana Food"
  ▶ [[Prototype]]: Object

App.js:65

▼ {resName: 'KFC', cusion: 'Chicken'} i
  cusion: "Chicken"
  resName: "KFC"
  ▶ [[Prototype]]: Object

React will take all these props and it will wrap it inside an object and it will pass over here as props. So this props will be an object now.

When you have to dynamically pass in some data to a component you pass in as a props.

Destructuring the props-

```
const ResturantCard = ({resName,cusion}) => {
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src="https://picsum.photos/200/100" />
            <h3>{resName}</h3>
            <h4>{cusion}</h4>
            <h4>4.4</h4>
            <h4>38 min</h4>
        </div>
    )
}
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName="Meghana Food" cusion="Briyani" />
                <ResturantCard resName="KFC" cusion="Chicken" />
            </div>
        </div>
    )
}
```

OR

```
const ResturantCard = (props) => {
    const {resName,cusion} = props
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src="https://picsum.photos/200/100" />
            <h3>{resName}</h3>
            <h4>{cusion}</h4>
            <h4>4.4</h4>
            <h4>38 min</h4>
        </div>
    )
}
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName="Meghana Food" cusion="Briyani" />
                <ResturantCard resName="KFC" cusion="Chicken" />
            </div>
        </div>
    )
}
```

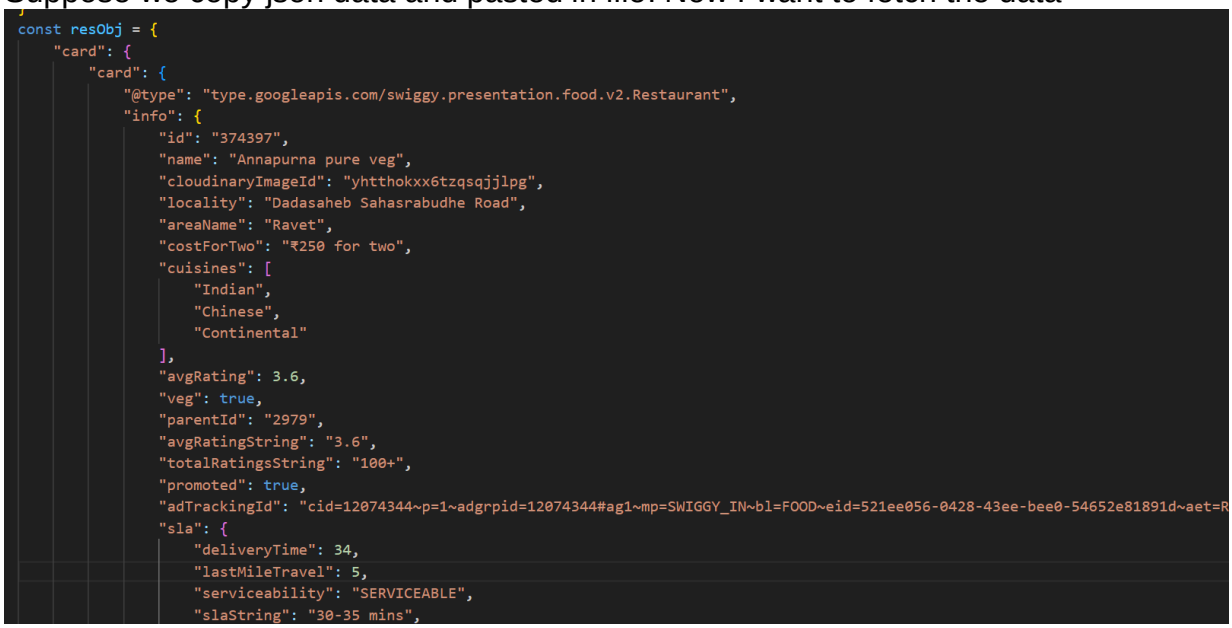
React is wrapping all these props into object passing it over here and destructing it.

How to data comes in json
Install Json chrome viewer in your system.
We can inspect -> network->refresh the page->preview

API-
https://www.swiggy.com/dapi/restaurants/list/v5?lat=12.9351929&lng=77.62448069999999&page_type=DESKTOP_WEB_LISTING
install JSON viewer extension

What is swiggy this concept is know as config driven ui
We can some offers in website but these offer available to me in banglore location. Suppose there is some other offer in Kolkata location. So there card diff. in delhi. In niwari there is no offer. How can you build a ui that you have seen sometime. Do you build the website for ever location. NO
So the website is driven by data this is known as config driven ui.
So ui is basically controlling you ui how the ui looks like using data using a config.
How the config comes from config comes from backend.
What is the config – this api this data is the config.

Suppose we copy json data and pasted in file. Now I want to fetch the data -



```jsx
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName={resObj} />
            </div>
        </div>
    )
}
```

```jsx
const ResturantCard = (props) => {
    const { resName } = props
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src={"https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/c8f1a629fb576f
              + resName.card.card.info.cloudinaryImageId} />
            <h3>{resName.card.card.info.name}</h3>
            <h4>{resName.card.card.info.cuisines.join(", ")}</h4>
            <h4>Star:- {resName.card.card.info.avgRating}</h4>
            <h4>{resName.card.card.info.sla.deliveryTime}</h4>
        </div>
    )
}
```

Cloudnary-image – it is the cdn image, all the image is hoisted in cdn
This is the url where the image are hosted and this Cloudnary-image ID is different for different restaurants. So what we are doing URL this will change but the CDN URL remains the same.

Suppose we have list of json , in that case we will use array-

```jsx
const ResturantCard = (props) => {
    const { resName } = props
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src={"https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/c8f1a629fb576f
              + resName.card.card.info.cloudinaryImageId} />
            <h3>{resName.card.card.info.name}</h3>
            <h4>{resName.card.card.info.cuisines.join(", ")}</h4>
            <h4>Star:- {resName.card.card.info.avgRating}</h4>
            <h4>{resName.card.card.info.sla.deliveryTime}</h4>
        </div>
    )
}
const restList = [...
]
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName={restList[0]} />
                <ResturantCard resName={restList[1]} />
            </div>
        </div>
    )
}
```

We created a reusable component restro card , we were able to pass in dynamic props inside that, Because we made a reusable card, this is how we render dynamic data.

Destructuring the code –

```jsx
const ResturantCard = (props) => {
    const { resName } = props
    const{cloudinaryImageId,name,cuisines,avgRating} = resName?.card?.card?.info
    const{deliveryTime} = resName?.card?.card?.info.sla
    return (
        <div className='res-card' style={{ backgroundColor: "yellow" }}>
            <img className='res-img' src={"https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/c8f1a629fb576f8
              + cloudinaryImageId} />
            <h3>{name}</h3>
            <h4>{cuisines.join(", ")}</h4>
            <h4>Star:- {avgRating}</h4>
            <h4>{deliveryTime}</h4>
        </div>
    )
}
const restList = [ ···
]
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* how to make dynamic cards  */}
                {/* passing the props like that */}
                <ResturantCard resName={restList[0]} />
                <ResturantCard resName={restList[1]} />
            </div>
        </div>
    )
```

Optional Chaining

Optional chaining in React.js allows you to access nested properties of an object without worrying about whether intermediate properties exist or not. It's particularly useful when dealing with props that might be undefined or null.

For example, suppose you have a component that receives props from its parent:

jsx Copy code

```jsx
const MyComponent = ({ data }) => {
  return (
    <div>
      <p>{data.user.name}</p>
    </div>
  );
};
```

If `data` or `data.user` is undefined, this would throw an error. To prevent this, you can use optional chaining:

```jsx
const MyComponent = ({ data }) => {
  return (
    <div>
      <p>{data?.user?.name}</p>
    </div>
  );
};
```

With optional chaining (`?.`), if `data` or `data.user` is undefined, the expression returns undefined instead of throwing an error. This ensures your application remains robust even when dealing with potentially missing data.

Assignment

How do I loop over this data and create this whole cards dynamically-

I will use javascript map function

```jsx
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* No need to write again and again  <ResturantCard /> */}
                {/* <ResturantCard resName={restList[0]} />
                <ResturantCard resName={restList[1]} /> */}
                {
                    restList.map(restaurant => <ResturantCard resName={restaurant}/>)
                    // it will loop all of the restaurant, for earch restaurant we have to return the <ResturantCard/>
                    // and agter that i will pass resturant data, what was the key over there was rest datav key
                }
            </div>
        </div>
    )
}
```

How to resolve this warning-

```
▶Warning: Each child in a list should have a unique "key" prop.   App.js:653

Check the render method of `Body`. See https://reactjs.org/link/warning-keys for more information.
    at ResturantCard (http://localhost:1234/index.7826abd7.js:3060:13)
    at Body
    at div
    at AppLayout
```

So unique key property means that each of these list item should be uniquely represented. Whenever you are looping on to anything you have to give a key.

```
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* No need to write again and again  <ResturantCard /> */}
                {/* <ResturantCard resName={restList[0]} />
                <ResturantCard resName={restList[1]} /> */}
                {
                    restList.map(restaurant => <ResturantCard key={restaurant.card.card.info.id} resName={restaurant}/>)
                    // it will loop all of the restaurant, for earch restaurant we have to return the <ResturantCard/>
                    // and agter that i will pass resturant data, what was the key over there was rest datav key
                }
            </div>
        </div>
    )
}
```

Why we use key- suppose new restaurant card added to the container, react does not know where we need put that restaurant card it has come at first place, it come has second place, without key it will re-render all the cards coz react does not know which is the new card is added it will treat all the restaurant card same. But if you give each of them a unique id then react will render one card or restaurant. It will not all the cards. This is the huge optimization.

React does not uniquely identify that element that's why its re-render everything.

We can use index as a key also.

```
const Body = () => {
    return (
        <div id="search">
            <div className='search'>Search</div>
            <div className='res-container'>
                {/* No need to write again and again  <ResturantCard /> */}
                {/* <ResturantCard resName={restList[0]} />
                <ResturantCard resName={restList[1]} /> */}
                {
                    restList.map((restaurant,index) => <ResturantCard key={index} resName={restaurant}/>)
                    // it will loop all of the restaurant, for earch restaurant we have to return the <ResturantCard/>
                    // and agter that i will pass resturant data, what was the key over there was rest datav key
                }
            </div>
        </div>
    )
}
```

But react says that never index as the key.

Suppose we have same level components and these components that are loops

Suppose new restaurant that came in at the first place what will happen is DOM will have to insert new restaurant on the first place. If you will not give id react will re-render all these restaurant cards coz react does not know which

restaurant card is new, react will not uniquely identify these restaurant cards. Suppose one more restaurant card added so react is basically removes all the cards from container and re-render all the cards coz react does not know which card is added, it will treat all the cards are same. But if you give each of them a unique id and suppose new card came with new id then react exactly knows that id1 , id 2 were already there the new element has come up first place . so, it will just render one restaurant.

Assignment

- Is JSX mandatory for React?
- Is ES6 mandatory for React?
- `{TitleComponent}` vs `{<TitleComponent/>}` vs `{<TitleComponent></TitleComponent>}` in JSX
- How can I write comments in JSX?
- What is <React.Fragment></React.Fragment> and <></> ?
- What is Virtual DOM?
- What is Reconciliation in React?
- What is React Fiber?
- Why we need keys in React? When do we need keys in React?
- Can we use index as keys in React?
- What is props in React? Ways to
- What is a Config Driven UI ?

Coding Assignment:

- Build a Food Ordering App
  - Think of a cool name for your app
  - Build a AppLayout
  - Build a Header Component with Logo & Nav Items & Cart
  - Build a Body Component
  - Build RestaurantList Component
  - Build RestaurantCard Component
  - Use static data initially
  - Make your card dynamic(pass in props)
  - Props - passing arguments to a function - Use Destructuring & Spread operator
  - Render your cards with dynamic data of restaurants

- Use Array.map to render all the restaurants

PS. Basically do everything that I did in the class, in the same sequence. Don't skip small things.

References

- Code Link: https://bitbucket.org/namastedev/namaste-react-live/src/master/
- React without JSX: https://reactjs.org/docs/react-without-jsx.html
- Virtual DOM: https://reactjs.org/docs/faq-internals.html
- Reconciliation: https://reactjs.org/docs/reconciliation.html

- React Fiber Architecture: https://github.com/acdlite/react-fiber-architecture
- React Without ES6: https://reactjs.org/docs/react-without-es6.html
- Index Keys as Anti-Pattern: https://robinpokorny.com/blog/index-as-a-key-is-an-anti-pattern/