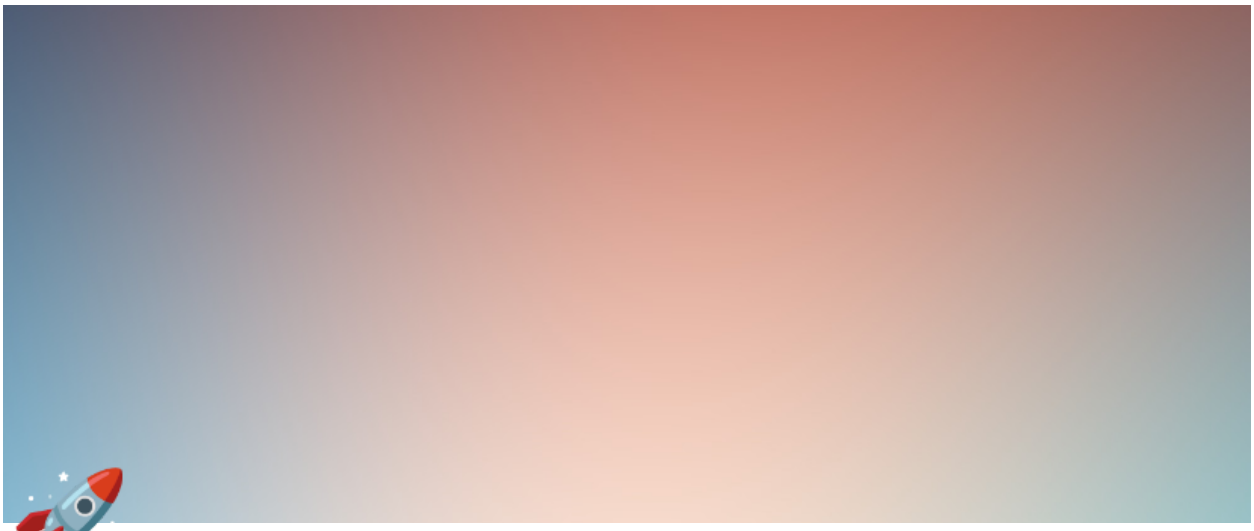

Talk is Cheap, Show me the Code! (Namaste-React)

💡 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 Components and 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

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>
 )
}
```