Ep-11 Data is the new Oil

Higher Order Components: is a fn that takes a Component & Returns a Component.

Input :- takes a Component, enhances that Component & Returns it back a new Component.

* Every React Application Two layers
  1) UI layer (UI layer is bound by Data layer)
  2) Data layer
     (consist of states, props, local vamables)

React DevTools => Chrome Extension for Debugging.

Controlled & UnControlled Component
→ in which your parent Component is controlling Children Compon
→ when children have it's own state & parent component do have Control our Children Component, then it is Uncontrolled Component.

Props Drilling : React has One-way data flow & passing data from One Component to another Component in huge React Appln is Very Complex process.

React Context : We can avoid Props drilling using React Context
→ Helps to make our data layer more powerful.
- It allows to keep data in Central place. & place data Only to Required Component or all Components
- Context can be used anywhere in the App.
- You have as many as Context in your App.

- How to create a context ⇒ using **createContext()** library

```
import { createContext } from "react";
```

Eg:-
```
const UserContext = createContext({
    isLoggedUser : "default",
});
export default UserContext;
```

- How to Use your Created Context in Component ⇒ using React Hook **useContext()**

```
import {useContext} from "react";
import UserContext from "../path/to/context";

const data = useContext(UserContext);
```

- Only the data which you are using at Multiple places, Only keep that data in Context. Don't put all data in Context.
- Context data can be accessed anywhere in the App.
- You can Use Context in another pages also

*** In class-based Component you can't use Hooks ***
So, you cannot use **useContext()** Hook also.
But, there exists a way to use these Hook class-based Component i.e,

```
import your Context file;
<UserContext.Consumer>
    {(data) => {cfg}}
</UserContext.Consumer>
```

(Context name; callback fn; JSX code)

- Context is the global space, you can provide to whole app or just a small portion of a app
- You can create multiple Context
- You can overwrite anywhere you want

To pass new information to our App ⇒ use

```
<UserContext.provider value={{}}>
    <whole App/>
</UserContext.provider>
```

it helps to pass new information to whole App

<Context.provider/> helps to send the data & you can replace the Context data with new data using (provider)

- The Component which is wrapped in <ContextName.provider/> the data can be accessed only by the wrapped Component.
- If you use provider from a Specific portion you can do that too.