**Context API**

* Context provides a way to pass data through component tree without props drilling process
* It's too important for large scale apps
* First Create a context   
  const UserContext = createContext();
* Then wrap whole app through context provider   
  <UserContext.Provider></UserContext.Provider>
* Now set context value   
  <UserContext.Provider value={{}}></UserContext.Provider>
* Now get context value   
  const { } = useContext(ContextName);

**Provider Context**

* Context provider wrap a context with props
* It will helps to use context with professional way
* Provide ris a simple function
* Create a provider   
  const ContextProvider = ({ children }) => {

// state, reducers, functions, values goes here   
 return (  
 <Context.Provider value={}>  
 {children}  
 </Context.Provider >  
 );  
   
}  
export default ContextProvider;

**Reducer State Management**

* State management is a cool things for SPA Apps
* It is the alternative of useState hook
* useState is built by useReducer
* Must we use a Reducer or redux or recoil for large scale application
* We must know reducers to know good use of redux / RTK
* Just be cool and be patience and dont panic

**Structure of Reducer**

* To create a reducer   
  const [ state, dispatch ] = useReducer(reducerFunction, initialState);
* Reducer function   
  const reducerFunction = (state, action / { type, payload }) => {  
   switch( type ){  
   case ‘type01’ :   
   return {}  
   case ‘type 02’ :   
   return {},   
   default :   
   return state;  
   }  
  }
* Dispatch is used to reach the reducer action and pass action type and payload

**Reducer with Context Provider**

* We must use reducer with context provider  
  const ContextProvider = ({ children }) => {

// state, reducers, functions, values goes here   
 return (  
 <Context.Provider value={}>  
 {children}  
 </Context.Provider >  
 );  
   
}  
export default ContextProvider;

[][