**📘 React useContext Hook Notes**

**🔹 What is useContext?**

* useContext is a React Hook that allows components to access values from a **React Context**.
* It helps avoid **prop drilling** by allowing data to be passed directly to any component that needs it.

**✅ Step 1: Creating a Context**

js

CopyEdit

import React, { createContext } from 'react';

// Create a context object

export const MyContext = createContext();

**✅ Step 2: Providing Context (Context Provider)**

js

CopyEdit

import React from 'react';

import { MyContext } from './MyContextFile';

const MyProvider = ({ children }) => {

const value = "Hello from Context!";

return (

<MyContext.Provider value={value}>

{children}

</MyContext.Provider>

);

};

export default MyProvider;

**📌 Explanation:**

* MyContext.Provider wraps around children and provides the context value to all nested components.
* The value can be a string, object, function, array, etc.

**✅ Step 3: Using Context in a Component (useContext)**

js

CopyEdit

import React, { useContext } from 'react';

import { MyContext } from './MyContextFile';

const ChildComponent = () => {

const contextValue = useContext(MyContext);

return <p>{contextValue}</p>;

};

**📌 Explanation:**

* useContext(MyContext) accesses the value from the nearest <MyContext.Provider>.
* You can now use contextValue like any other variable.

**✅ Step 4: Full Example**

**App.js**

js

CopyEdit

import React from 'react';

import MyProvider from './MyProvider';

import ChildComponent from './ChildComponent';

const App = () => {

return (

<MyProvider>

<ChildComponent />

</MyProvider>

);

};

export default App;

**🧠 useContext with Complex Objects**

js

CopyEdit

const user = {

name: "Nikhil",

isLoggedIn: true,

};

<MyContext.Provider value={user} />

Access:

js

CopyEdit

const { name, isLoggedIn } = useContext(MyContext);

**⚠️ Important Points**

* Only works inside **function components**.
* Must be used **within a Context Provider**; otherwise, it returns undefined.
* Best for **global state**, theme toggles, language settings, authentication, etc.