React Context API

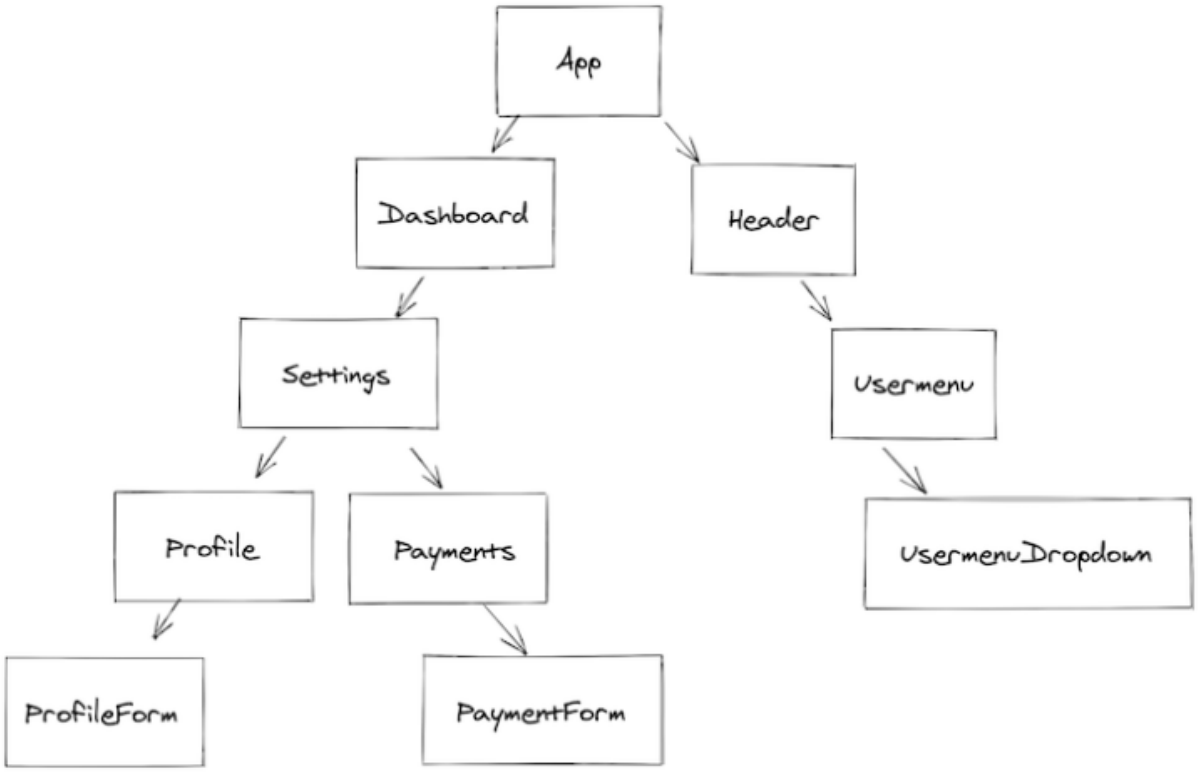
What is context API in ReactJS?

In a traditional React application, data is often shared between components using props. Sharing this data can be hectic, especially when shared between multiple nested members. Also, sharing data between two child components can take time and effort. Hence the need for global state management.

Here comes into play React context API.

The context API is a method in React that helps solve modern application problems such as Prop Drilling.

The props drilling problem occurs when you pass a prop somewhere down the tree. When a project grows, data passing through props becomes chaotic, making the code more vulnerable and complex. To tackle this problem, we use Context API.



React context is a built-in API that uses the useContext hook to share data across components.

Functioning of ContextAPI

ContextAPI provides us with the functionality of global state management.

In global state management components do not have to be dependent on the parent component for getting the data. In fact, it reduces the cumbersome between the functional components.

Context API has two parts -

- Provider
- Consumer

Provider
The provider is the method that provides the states globally to the whole application.

Consumer
The consumer here is the individual component that intends to access the state.

Context API consists of two hooks createContext and useContext

What is createContext?
createContext is React Api which lets you create a context that components can provide or read.

What is useContext?
useContext is **a React Hook that lets you read and subscribe to context from your component.**

How to Use the React Context API?

First, we will create the context from where all the states will be managed.

How to create context?
useContext is a built-in hook in React. You can start using the context API by importing the createContext function from React like this:

```
import React, { createContext } from "react";
const AppContext = createContext();
```

Here, we initialized our context and named it AuthContext. The next step is to provide the context

How to provide the context to the components that need it?

The context API uses a provider to pass data to its child components. You will have to wrap all components with a provider component.

```
<AppContext.Provider value={data}>{children}</AppContext.Provider>
```

The Provider component has a value prop as seen above. The value of the context can either be updated or set using the value prop.

```
import React, { createContext } from "react";
const AppContext = createContext();

const AppProvider = ({ children }) => {
  const data = {
    name: "rajat",
    age: 18,
  };
  return <AppContext.Provider value={data}>{children}</AppContext.Provider>;
};

export { AppContext, AppProvider };
```

Now we will wrap our main App component inside this AppProvider component.

```
<AppProvider>
  <App />
</AppProvider>
```

How to consume the context?

We can consume the context by using the useContext hook. Without passing data through nested components, you can access your context in any component you want. Here's how.

```
import React, { useContext } from "react";
import { AppContext } from "./AppProvider";

const Child = () => {

    const newdata = useContext(AppContext)
  return (
    <>
      <h1>this is child</h1>
      <h2>name : {newdata.name} and age : {newdata.age}</h2>
    </>
  );
};

export default Child;
```