React App to connect to Phantom Wallet

① Import functionalities

② create type

type refers to the type of a component or prop. A component type defines what the component returns when it's rendered & how it behaves - for ex, a component could be a function or a class, and this would be it's "type".

In React, a component is a self-contained piece of code that manages it's own content, presentation & behaviour.

Components are like JS functions, that accept arbitrary inputs, called "props", and return REACT elements describing what should appear on the screen.

③ Create a provider interface to store the Phantom provider

4. Create state variable for the provider

5. Create state variable for the wallet key

6. prompt user to connect to wallet if it exists.

In typescript, an interface is a way to define a contract on a function, object or class regarding the structure & shape it should have.

The 'window' object is a global object in client-side JS that represents the browser's window.

Node providers are organizations that offer a way to access the information on the blockchain without having to run your own node.

Phantom acts as a Node Provider.

Node - connects with rest of blockchain

Phantom Wallet provides with the connection to connect to the Solana Blockchain.

In this lesson, we'll be making a template for the Phantom provider that will help us connect to a phantom wallet.

Phantom Provider is the API that dApps use to interact with the user's wallet

An interface is an abstract type that details what properties a given

object it can have.

getProvider checks if a Phantom provider exists & then retrieves it.

First check for Solana object in user's browser. When someone downloads a Phantom browser, Phantom injects a Solana browser in the window.

So by checking if Solana is in window, we're checking if Solana has possibly injected a Solana object into the window of the web application.

If that is the case we can set window.solana as the provider.

There are many providers that use Solana, So we also need to check if it is a Phantom provider.

Flow of the app:

Once you connect to the app, you'll need to keep track of a wallet key

useState initializes the Phantom provider variable.

useEffect updates the variable.

We're also setting up a state variable

to keep track of the wallet key:

UI - 3 possible cases:

1. provider exists
   wallet key doesn't
2. Provider exists
   wallet key exists
3. Provider doesn't exist

Connect wallet Function

Get solana item from the window.
If solana item exists, that means that there's a Phantom provider.
We can continue to connect to the wallet using solana.connect.
Extract public key, assign it to walletKey state variable.