1. **What is prop drilling?**

* Prop drilling is basically a situation when the same data is being sent at almost every level due to requirements in the final level.
* Prop drilling is a situation where data is passed from one component through multiple interdependent components until you get to the component where the data is needed.

1. **What is lifting the state up?**

* In React, sharing state is accomplished by moving it up to the closest common ancestor of the components that need it. This is called “lifting state up”.
* This is the state we “lifted up” from the inputs, and it will serve as the “source of truth” for both of them.
* It is the minimal representation of all the data we need to know in order to render both inputs.
* This allows us to more easily share state among all of these components that need rely upon it.

1. **What is Context Provider and Context Consumer?**

* Context provides a way to pass data through the component tree without having to pass props down manually at every level.
* In a typical React application, data is passed top-down (parent to child) via props, but such usage can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application.
* Context provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.
* Context.Provider –
  + Every Context object comes with a Provider React component that allows consuming components to subscribe to context changes.
  + The Provider component accepts a value prop to be passed to consuming components that are descendants of this Provider.
  + One Provider can be connected to many consumers. Providers can be nested to override values deeper within the tree.
* Context.Consumer –
  + A React component that subscribes to context changes. Using this component lets you subscribe to a context within a [function component](https://reactjs.org/docs/components-and-props.html#function-and-class-components).
  + Requires a [function as a child](https://reactjs.org/docs/render-props.html#using-props-other-than-render). The function receives the current context value and returns a React node.
  + The value argument passed to the function will be equal to the value prop of the closest Provider for this context above in the tree.
  + If there is no Provider for this context above, the value argument will be equal to the defaultValue that was passed to createContext().

1. **If you don’t pass a value to the provider does it take the default value?**

* If there is no Provider for this context above, the value argument will be equal to the defaultValue that was passed to createContext().
* When there's no Provider, the defaultValue argument is used for the function createContext.
* This is helpful for testing components in isolation without wrapping them, or testing it with different values from the Provider.