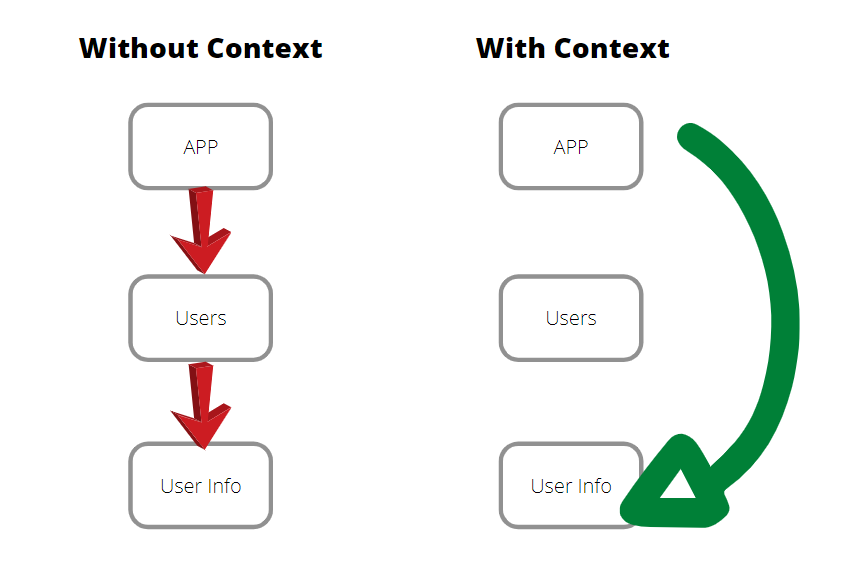
**React**

**Day 55th**

**Context API**

Context API allows data to be passed through a component tree without having to pass props manually at every level. This makes it easier to share data between components. A diagram illustrating how context API works.

**What is props drilling??**

Prop drilling is a term used in react (and other component based frameworks) to describe the process of passing data through multiple layers of nested components, even when some of those intermediate components don’t actually need the data themselves. It’s like passing a message through a series of people where some people where some people only relay the message without needing to understand or use it.

**Context API**

Context API is a kinda new feature added in version 16.3 of React that allows one to share state across the entire app or part of it lightly and with ease.

The react context API is a way for a react app to effectively produce global variables that can be passed around. This is the alternative to prop drilling or moving props from grandparent to child parent and so on. Context is also touted as an easier, lighter approach to state management using redux.

Context API is a new feature added in version 16.3 of React that allows one to share state across the entire app lightly and with ease.

**Component Composition**

Component composition in React is a fundamental pattern for building complex user interfaces by combining smaller, self-contained components. It emphasizes creating a modular, reusable and maintainable code by treating components as building blocks that can be assembled together.

**Key aspects of component composition**

**Nesting Components**

Components can be nested within one another, forming a hierarchical structure. A parent component can render one or more child components, effectively encapsulating their behavior and rendering.

**Children Prop**

A common way to achieve composition is by utilizing the children prop. This implicit prop allows a component to render whatever is passed between its opening and closing tags providing a flexible way to inject content or other components.

**Passing component as prop**

Components can also be passed as explicit props to other components, allowing for more specific control over what is rendered and where.

**Reusability and modularity**

Composition promotes breaking down a UI into smaller, focused components, each responsible for a specific piece of functionality or UI. This enhances reusability across different parts of an application and improves overall code organization and maintainability.

**Flexibility and extensibility**

By composing components you can easily extend or modify the behavior and appearance of a component without altering its internal logic. This leads to more adaptable and scalable applications.

**What is Children props (props.children)**

The concept of children props (written as {prosp.children}) is far less twisted than it sounds when placed within the context of ReactJS code. The {**props.children**} property allows you to create a generic template component that can be modified by the parent when it is invoked. This means that a parent component can pass whatever is needed in the child component, even generated layout features that can then be rendered by the child.

**The props and child a tragic story**

Once upon a time there was a component that wanted to pass information down to its grandchildren, and any subsequent children in the application hierarchy. This worked well until the family started getting really big and the parent component noticed that many of the children were nearly identical and only needed slightly different information to be passed down through props.

The most basic implementation of props is to define props in the parent component and pass it down to the next child component in the hierarchy, and then pass the value again through props in the child component, and then again through the grandchild – which is repeated as needed until the passed value arrives in the target component. The process is tedious, error prone, and also makes the code less flexible.

{**props.children**} **a better way**

Implementation is pretty simple. In the parent component, import the child component, but instead invoking it with a self – closing tag, use an standard open/closing tag. The information that you want to pass through props – in addition to the standard implementation of passing props – is placed between the opening and closing tags of child component.