**Introduction to React  
  
What is ReactJS?**

**ReactJS** is an open-source JavaScript library used to create single page

Application

React is a JavaScript library created by Facebook.  
  
**Jordan Walke**, One of facebook’s Software engineers, Created React

**Why we use ReactJS?**

• It uses virtual DOM (JavaScript object), which improves the performance of

the app.

• The JavaScript virtual DOM is faster than the regular DOM.

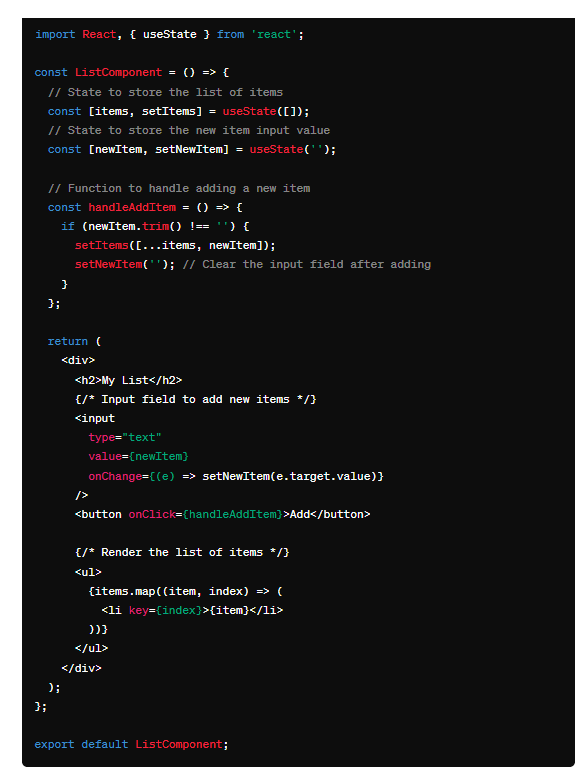
Framework - it is bootstrapped with nearly every single thing that you'll

need to make a complete, large-scale app.

* Component-Based Architecture
* Virtual Dom,
* Declarative Syntax
* Reusability
* Larger Ecosystem
* Strong Community Support
* Developed and Maintained by Facebook

**What is Virtual DOM ?**

Sure, let's consider a simple example of a React component that renders a list of items and allows the user to add new items to the list. We'll see how React utilizes the Virtual DOM to efficiently update the UI.



**In this example:**

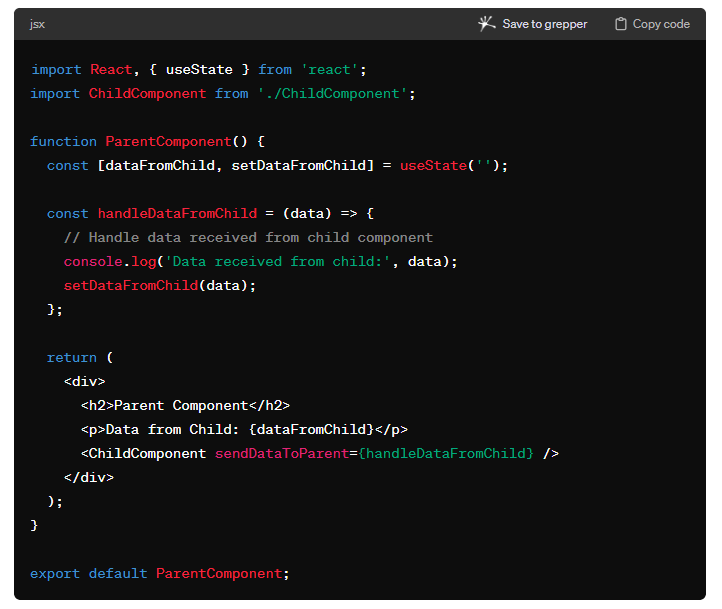
* We have a ListComponent that maintains two state variables: items, which holds the list of items, and newItem, which holds the value of the input field for adding new items.
* When the user types in the input field, the newItem state is updated.
* When the user clicks the "Add" button, a new item is added to the items state array, and the input field is cleared.
* The list of items is rendered using the map function, creating a <li> element for each item in the items array.
* Each list item has a unique key attribute set to its index in the array. This helps React identify which items have changed, been added, or been removed efficiently during updates.

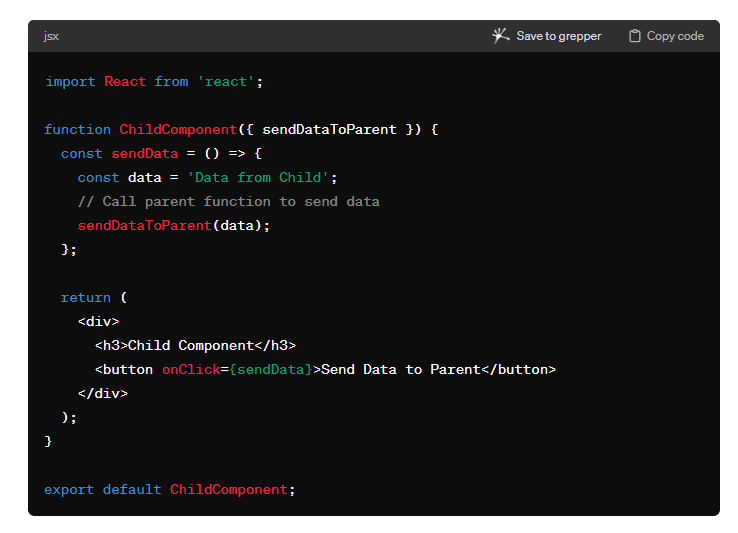
Behind the scenes, React uses the Virtual DOM to perform efficient updates to the UI. When the state changes (e.g., adding a new item), React reconciles the Virtual DOM with the previous state, identifies the differences, and updates only the necessary parts of the real DOM, resulting in a smooth and performant user experience.

**Settings necessary tools for React**

**React with CDN  
  
  
Passing Props from child to parent component**

**Parent Component :**



**Child Component**

• The parent component (ParentComponent) defines a state variable dataFromChild to store the data received from the child component.

• It also defines a callback function handleDataFromChild to update the state with the data received from the child.

• The child component (ChildComponent) receives a prop sendDataToParent, which is the callback function passed from the parent.

• When a button is clicked in the child component, it calls the sendDataToParent function with the data to be sent to the parent.

• Upon receiving the data, the parent component updates its state with the received data and renders it.

**useEffect**

1. It will render first time when components load is called as component mounting (it will run atlast)
2. Whenever state your having is changing, UseEffect will be run as called component updating

3. When u add array [] dependency – It will run only components get mounted.

**4.** When u add array with Condition [condition]); - It will run only when the

condition is changed// Every rerender

useEffect(() => {

console.log("I run everytime this component rerenders");

});

// onMount

useEffect(() => {

console.log("I Only run once (When the component gets mounted)")

}, []);

// Condition based

useEffect(() => {

console.log("I run everytime my condition is changed")

}, [condition]);

// Condition based with "clean up"

useEffect(() => {

console.log("I run everytime my condition is changed")

return () => {

console.log("Use this return as a 'clean up tool' (this runs before the actual code)")

}

}, [condition]);

**useRef**

Exactly! You've provided a concise explanation of what useRef is and how it's used in React functional components.

useRef indeed allows you to create a mutable reference that persists across renders without causing the component to re-render when its value changes. It's commonly used for accessing and interacting with DOM elements directly, like focusing an input element or measuring its dimensions.

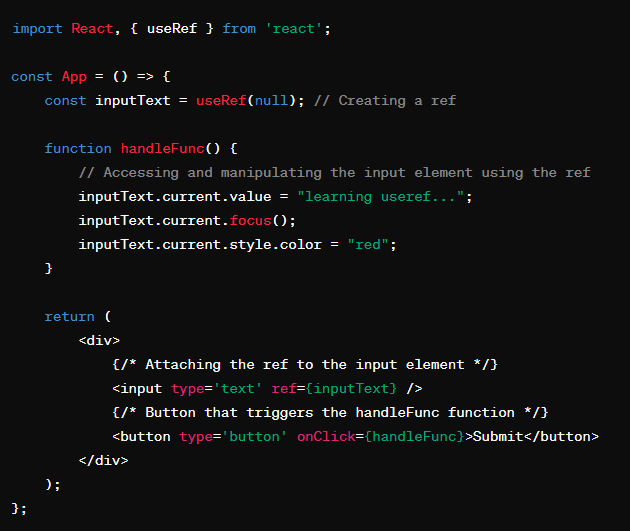
**Here's a breakdown of your explanation:**

useRef is a React Hook: It's part of the Hooks API introduced in React 16.8, which allows functional components to use state and other React features without writing a class.

It lets you reference a value that’s not needed for rendering: Unlike useState, changes to the value returned by useRef won't trigger a re-render of the component.

useRef allows directly creating a reference to the DOM element: You can create a ref and attach it to a DOM element using the ref attribute. This allows you to interact with the DOM imperatively.

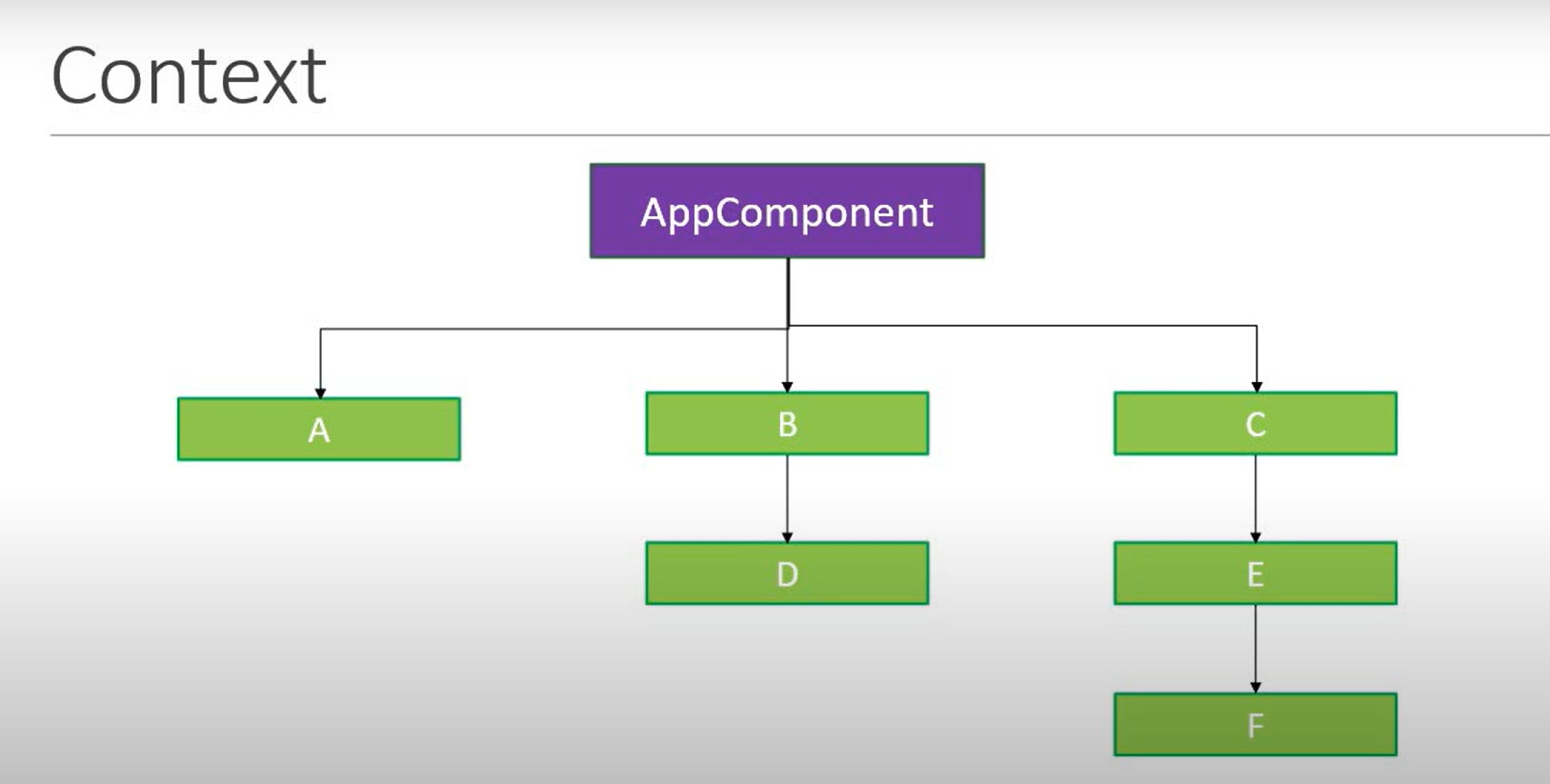
const ref = useRef(initialValue): This is how you typically use useRef. You initialize it with an optional initial value, and the returned ref object persists across re-renders, holding the current value of the reference.

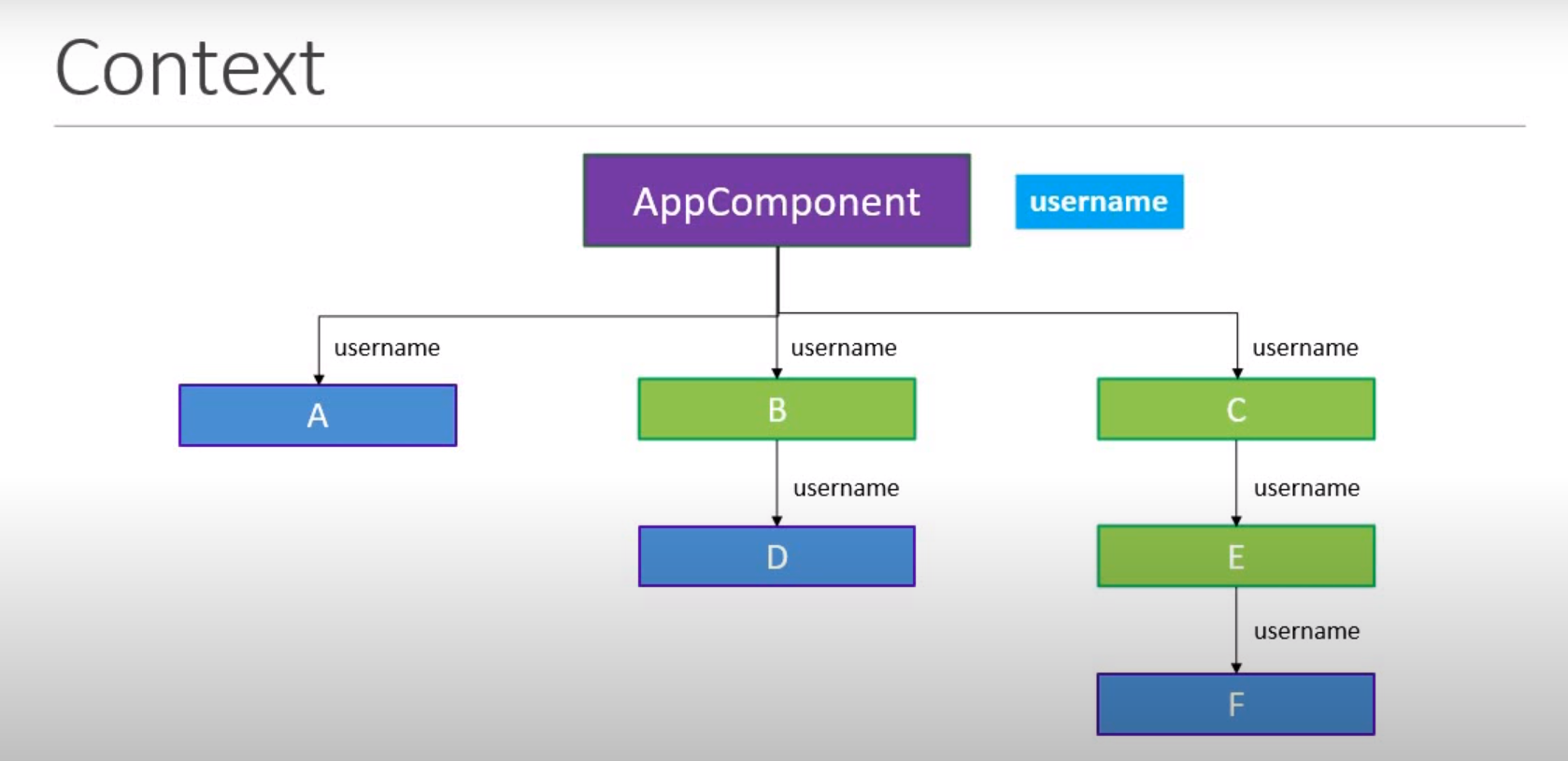
Overall, useRef is a powerful tool for working with React components, especially when you need to interact with the DOM directly or maintain values across renders without causing re-renders.

**UseContext**

1. **What is Context Api ?**
2. **How the react context is used**
3. **How useContext works**

Context provides a way to pass data through the component tree without having to pass props down manually at every level.





**Right most part of component tree**

