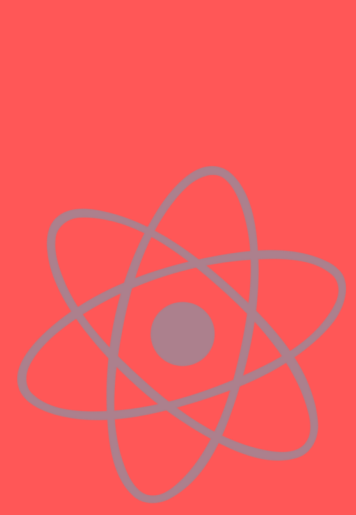
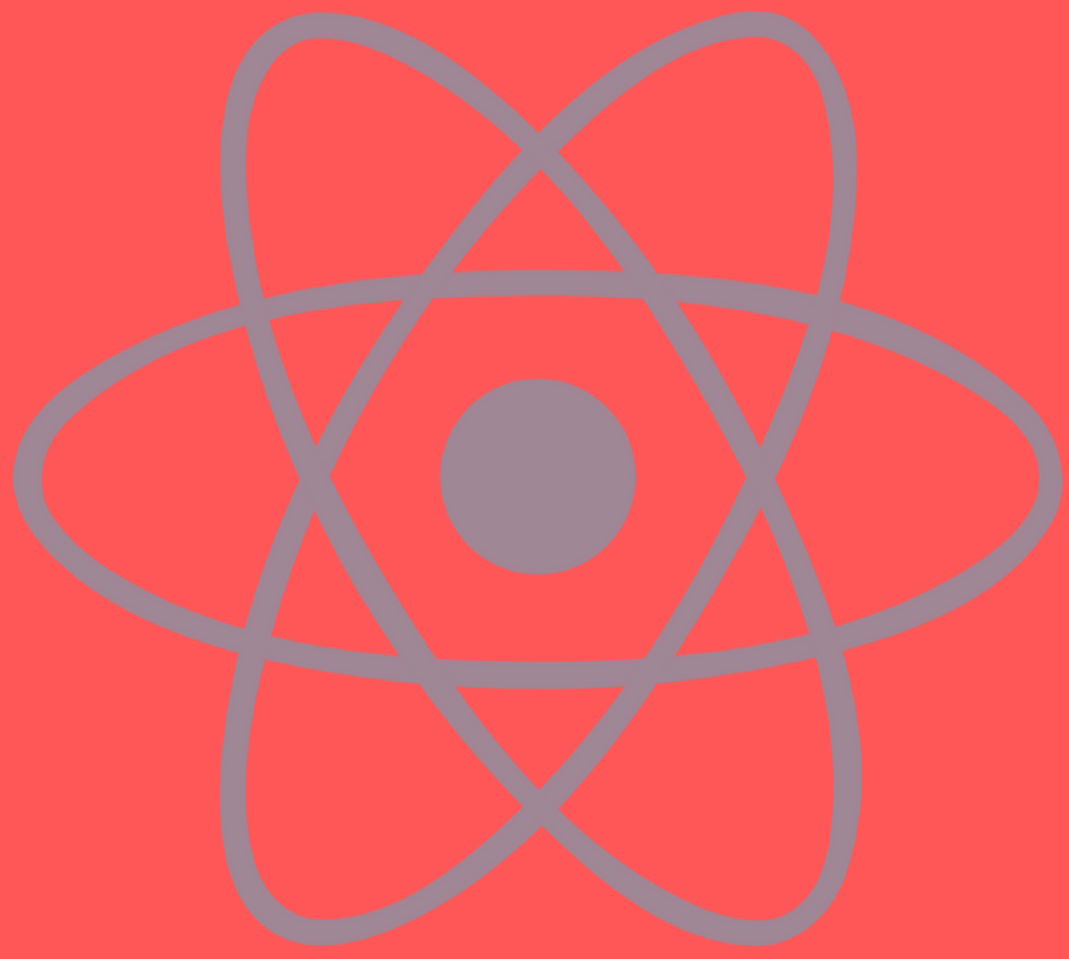
All

React.Js Hooks Explained





@umargafia

- useState Hook
- useEffect Hook
- useRef Hook
- useCallback Hook
- useMemo Hook
- useContext Hook
- useReducer Hook
- useDebugValue
- useDeferredValue
- useId
- useImperativeHandle
- useInsertionEffect
- useLayoutEffect
- useSyncExternalStore
- useTransition

@umargafia

useState Hook

This hook allows you to store and change information within a component. It returns a pair of the current state, a function that lets you update it and the value of the state

useState code example

```
const [stateVariable, setStateFunction] = useState(initialValue);
```

@umargafia

useEffect Hook

This hook lets you perform side effects like data fetching, updating the DOM, or subscribing to events.

useEffect code example

```
useEffect(() => {
  // Side effect code here
  return () => {
    // Cleanup code (optional)
  };
}, [dependencies]);
```

@umargafia

useRef Hook

This hook lets you reference a value in your component. It doesn't cause the component to re-render when it changes and it remembers its value even after the component re-renders.

useRef code example

```
const myRef = useRef(initialValue);
```

@umargafia

useCallback Hook

This hook is used to memoize (remember) functions, so you don't have to recreate it every time your component re-renders, unless something it depends on changes.

useCallback code example

```
const memoizedCallback = useCallback(callbackFunction, dependencies);
```

@umargafia

useContext Hook

React Context is a way to manage state globally, It helps you pass data around components without having to manually pass it through props.

useContext code example

```jsx
import React, { createContext, useContext } from 'react';

const MyContext = createContext();

const ParentComponent = () => (
  <MyContext.Provider value="Hello from context!">
    <ChildComponent />
  </MyContext.Provider>
);

const ChildComponent = () => {
  const contextValue = useContext(MyContext);
  return <p>{contextValue}</p>;
};
```

@umargafia

useReducer Hook

This hook is used for managing complex state logic that involves multiple sub-values or when the next state depends on the previous one

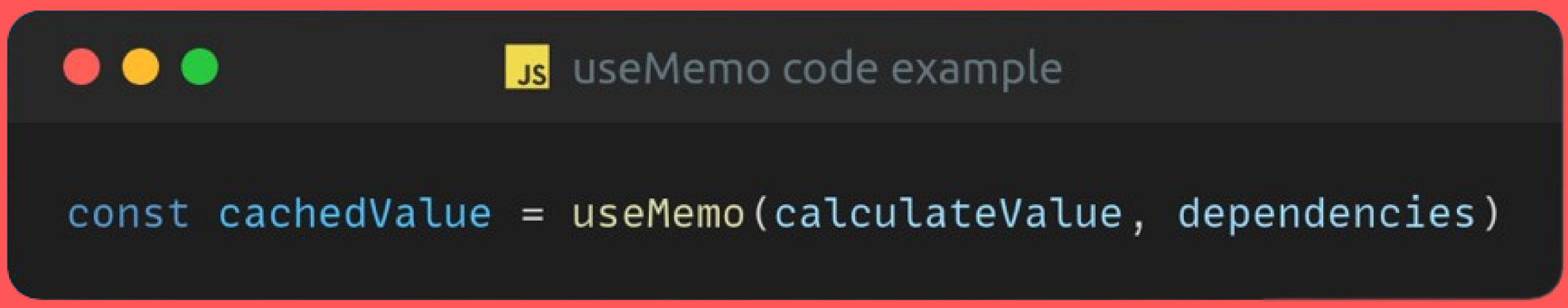
useReducer code example

```
const [state, dispatch] = useReducer(reducerFunction, initialArg, initFunction);
```

@umargafia

useMemo Hook

This hook remembers the result of a function and only re-calculates if the inputs change.



@umargafia

useDebugValue Hook

This hook allows developers to add a label or description for custom hooks, it helps developers understand the purpose and usage of custom hooks when inspecting components in browser developer tools.

JS useDebugValue code example

```
useDebugValue(value, format);
```

@umargafia

useDeferredValue Hook

This hook allows you to postpone a value change that might cause a component to lag during rendering.

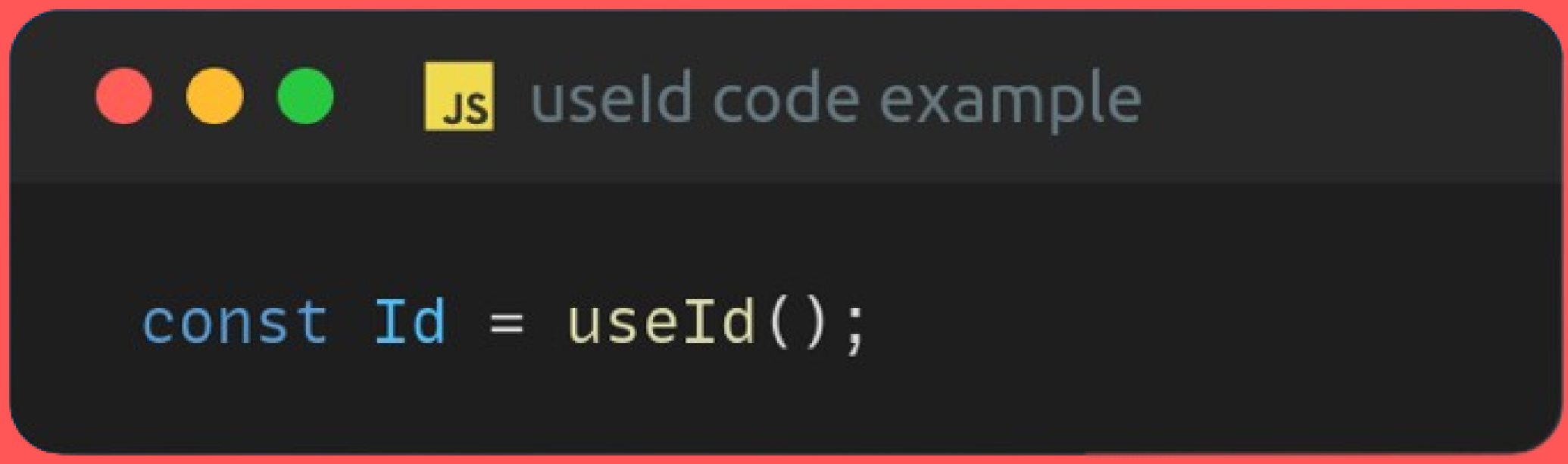
useDeferredValue code example

```
const deferredValue = useDeferredValue(value, options);
```

@umargafia

useId Hook

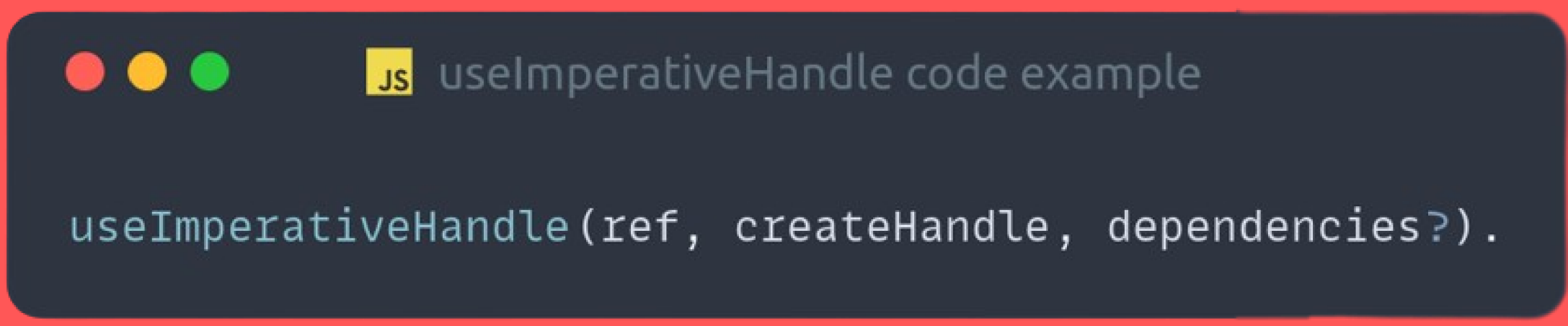
This is a custom hook that generates unique IDs within a React component.



@umargafia

useImperativeHandle

This hook allows you to control what is returned by a ref prop, specifically the ref handle.



@umargafia

useInsertionEffect

useInsertionEffect is a custom hook that allows you to insert the styles before any layout effects fire

useInsertionEffect code example

```
useInsertionEffect(setup, dependencies?)
```

@umargafia

useLayoutEffect Hook

This hook is similar to useEffect, but it runs after all DOM mutations are complete. This makes it useful for reading layout from the DOM and synchronizing state.

useLayoutEffect code example

```
useLayoutEffect(() => {
  //Do something and either return undefined or a cleanup function
  return () => {
    //Do some cleanup here
  };
}, [dependencies]);
```

@umargafia

useSyncExternalStore

useSyncExternalStore is a React Hook that lets you subscribe to an external store.

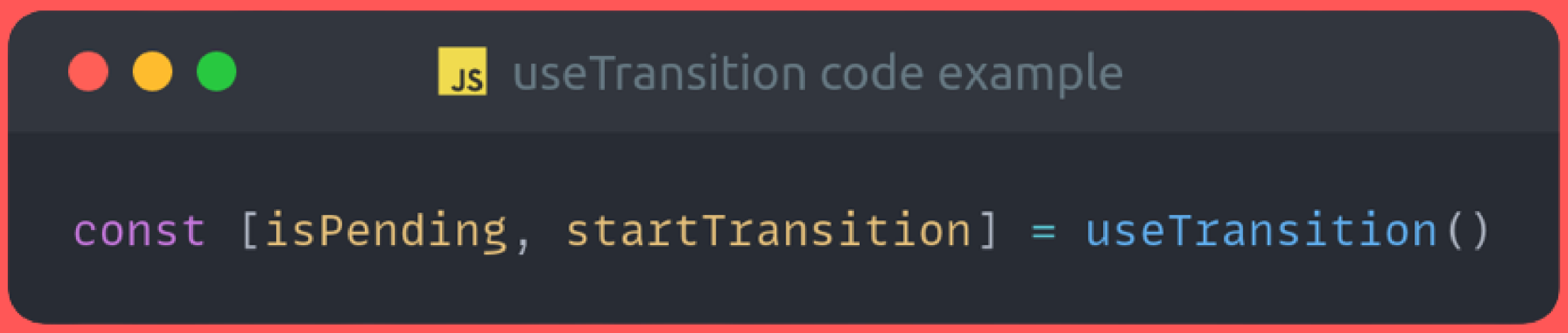
useSyncExternalStore code example

```
const snapshot = useSyncExternalStore(subscribe, getSnapshot, getServerSnapshot?)
```

@umargafia

useTransition Hook

useTransition is a React Hook that lets you update the state without blocking the UI.



@umargafia

Thank You

If you found it valuable, kindly consider liking and sharing it with your friends.

@umargafia