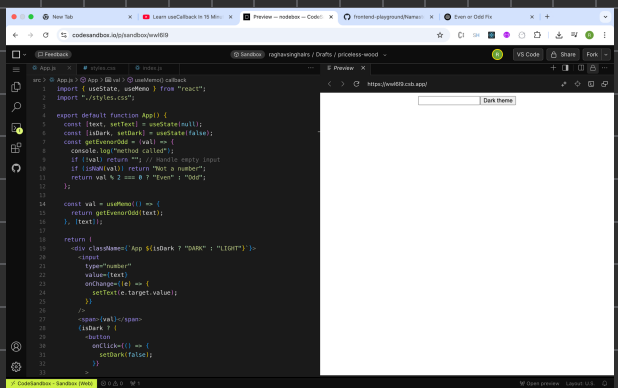
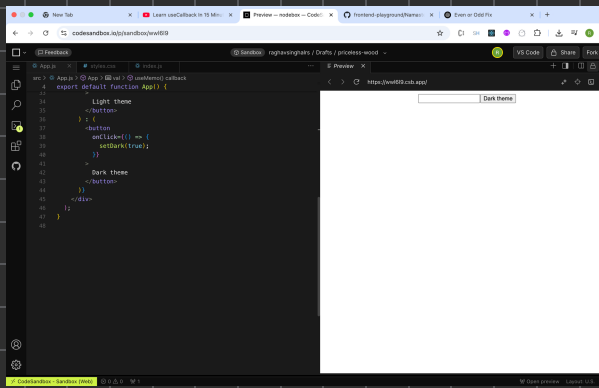
1. Use memo

As how many state variables we are using whenever it updates, whole component rerender so using we memo we can establish a situation where whenever input changes then the calculation rehappens otherwise uses the old cache.





2. Use callback

It is used to memoize a function so that its reference stays the same between renders unless dependencies change.

Some usecases:

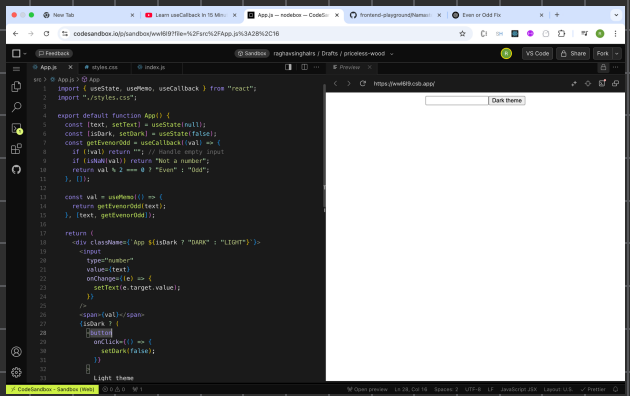
When you pass a function as a prop to child components and want to avoid recreating it on every render.

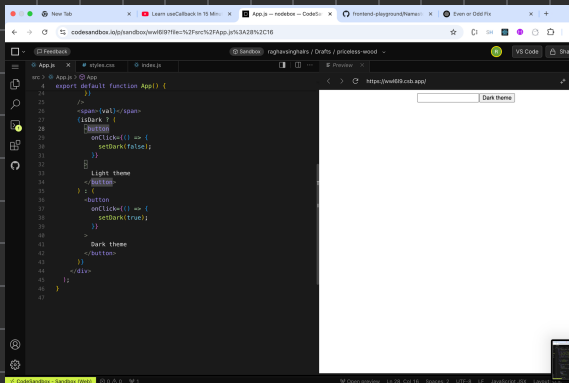
Summary =>

use Callback — getEvenOrOdd

use Memo — getEvenOrOdd(text)

Keeps function and stable. Avoids recalculating unless needed.





3. UseReducer

- It is used for state management.
- Alternative of useState hook.
- Preferable for complex state management logic, when state transitions depend on actions.

Syntax

```
const [state, dispatch] = useReducer(reducer, initialState);
```

reducer(currentState, action)

When use useReducer →

* When you have multiple related state updates.
* When the next state depends on the previous state.

Basic terminology :-

Reducer function → a function that decides how to update the state based on an action.

Dispatch function => used to send ("dispatch") an action to the reducer.