> React forwardRef and Asynchronous Stability — rewritten in Flavio's voice

So here's what's going on: you've got a React component that's trying to be fancy with `forwardRef`, but somewhere along the way it's tripping over its own feet. When async state updates fire, the reference to your input goes null, and suddenly you can't do the basic stuff you need to — like focus the field or trigger some internal modal logic. That's the kind of bug that feels random until you realize it's all about reference stability.

The goal here isn't just to make the ref "work again." It's to make it *predictable* — stable across renders, immune to async updates, and safely controlled by the parent without exposing more than it should.

Here's the pattern I'd use, the "architect way": treat the internal input ref like a foundation element. Never rebuild it; just keep pointing to the same node. Then expose only the intentional, safe interface through `useImperativeHandle`.

So, something like this:

```
import React, {
  useState,
  useRef,
  useImperativeHandle,
  forwardRef,
  useCallback,
} from "react";

type ModalInputHandle = {
  openModal: () => void;
  focusInput: () => void;
};

const ModalInput = forwardRef<ModalInputHandle, { initialValue?: string }>
(
  ({ initialValue = "" }, ref) => {
    const [value, setValue] = useState(initialValue);
    const inputRef = useRef<HTMLInputElement | null>(null);
    const [isOpen, setIsOpen] = useState(false);

    const openModal = useCallback(() => {
      setIsOpen(true);
      // guarantee focus once the modal animation or async render
completes
      setTimeout(() => inputRef.current?.focus(), 0);
    }, []);

    const focusInput = useCallback(() => {
      inputRef.current?.focus();
    }, []);

    useImperativeHandle(
      ref,
      () => ({
```

```
        openModal,
        focusInput,
      }),
      [openModal, focusInput]
    );

    return (
      <>
        {isOpen && (
          <div className="modal">
            <input
              ref={inputRef}
              value={value}
              onChange={(e) => setValue(e.target.value)}
              placeholder="Type here..."
            />
          </div>
        )}
      </>
    );
  }
);

export default ModalInput;
```