

# Generic Button Component TS



Yes! You can rewrite that line in different ways while keeping the same functionality. Here are a few alternatives:

## 1. Using Arrow Function with Explicit Return Type

```
tsx

const Input = (props: InputProps): JSX.Element => {
  const { label, type = "text", value, onChange, placeholder, ...rest } = props;
  return (
    <div className="flex flex-col">
      {label} && <label className="mb-1 text-gray-700">{label}</label>
      <input
        type={type}
        value={value}
        onChange={onChange}
        placeholder={placeholder}
        className="px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
        {...rest}
      />
    </div>
  );
};
```

## 2. Using a Function Declaration Instead of an Arrow Function

```
tsx

function Input({ label, type = "text", value, onChange, placeholder, ...props }: InputProps): JSX.Element {
  return (
    <div className="flex flex-col">
      {label} && <label className="mb-1 text-gray-700">{label}</label>
      <input
        type={type}
        value={value}
        onChange={onChange}
        placeholder={placeholder}
        className="px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
        {...props}
      />
    </div>
  );
}
```

## 3. Using `React.forwardRef` for Ref Support

If you want to forward a ref for handling focus or other DOM manipulations:

```
tsx

const Input = React.forwardRef<HTMLInputElement, InputProps>(
  ({ label, type = "text", value, onChange, placeholder, ...props }, ref) => {
    return (
```




```

<div className="flex flex-col">
  {label && <label className="mb-1 text-gray-700">{label}</label>}
  <input
    ref={ref}
    type={type}
    value={value}
    onChange={onChange}
    placeholder={placeholder}
    className="px-3 py-2 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-500"
    {...props}
  />
</div>
);
}
);

export default Input;

```

## Which One to Use?

-  **Use the first approach** (arrow function with explicit return) if you want to keep it concise.
-  **Use the second approach** (function declaration) if you prefer traditional functions.
-  **Use `forwardRef`** if you need to work with input refs.

Would you like any other optimizations? 