# How to Build a Custom React Hook

This document outlines the steps and best practices to create your own custom React hook.

## 1. Identify Shared Logic

Look for duplicated logic in multiple components (e.g., window resize, form fields, fetch calls).

## 2. Create a Function Starting with 'use'

function useWindowWidth() {  
 const [width, setWidth] = useState(window.innerWidth);  
 useEffect(() => {  
 const handleResize = () => setWidth(window.innerWidth);  
 window.addEventListener('resize', handleResize);  
 return () => window.removeEventListener('resize', handleResize);  
 }, []);  
 return width;  
}

## 3. Use Other Hooks Inside

You can call `useState`, `useEffect`, `useRef`, or even other custom hooks inside your new hook.

## 4. Return What You Need

Return the data, functions, or state values your component will use.

## 5. Use It Like a Built-in Hook

const width = useWindowWidth();

Use it in the top level of your component function, just like React's built-in hooks.

## 6. Best Practices

• Always start with `use` in the function name.  
• Never call hooks inside loops or conditionals.  
• Keep hooks small, focused, and reusable.  
• Use `useDebugValue` to improve DevTools visibility (optional).