

React Custom Hooks Cheat Sheet

TypeScript-first - React 19.2+ - 9 copy-paste hooks, the race guard, and the 7 anti-patterns. Full post + verified repo: zainvibes.com/blog/react-typescript-custom-hooks

The 9 hooks B1-B9

B1	<code>useToggle(initial = false) → [on, toggle, setOn] as const</code>	Boolean state, tuple API
B2	<code>useDisclosure(initial = false) → { isOpen, open, close, toggle }</code>	Menus, modals, drawers
B3	<code>useDebounceValue<T>(value: T, delayMs = 300) → T</code>	Trailing value; cleanup is the debounce
B4	<code>useFetchJson<T>(url: string null, parse: (json: unknown) → T) → FetchState<T></code>	Fetch state, derived loading, abort-safe
B5	<code>useWindowEvent<K extends keyof WindowEventMap>(type: K, handler, opts?) → void</code>	Typed listeners via <code>useEffectEvent</code> , attach once
B6	<code>useOnClickOutside<E extends HTMLElement>(ref: RefObject<E null>, handler) → void</code>	Close on outside pointerdown
B7	<code>useMediaQuery(query: string, serverFallback = false) → boolean</code>	<code>useSyncExternalStore</code> for browser state
B8	<code>useLocalStorage<T>(key: string, initialValue: T) → [value, setValue, remove] as const</code>	localStorage synced across tabs
B9	<code>useProductSearch(query: string) → FetchState<Product[]></code>	Composition: B3 + B4 as a domain hook

```
type FetchState<T> = { status: 'idle' } | { status: 'loading' } | { status: 'success'; data: T } | { status: 'error'; error: Error }
```

THE RACE GUARD (B4)

```
if (controller.signal.aborted) return;
```

Put it at the top of the SUCCESS handler too, not just `.catch`. A response that lands while `res.json()` is still parsing during a URL change is already aborted - unguarded, its stale `setState` overwrites the fresh result and derived loading sticks on forever. Found by a stress test, not by reading the code.

7 anti-patterns

1. No `useMount` / `useEffectOnce` - name hooks after concrete use cases.
2. No use prefix on a function that calls no hooks.
3. Hooks share logic, not state - shared state lives outside React.
4. Never `setState` in an Effect for a derivable value - compute it in render.
5. Never read or write `ref.current` during render.
6. Never call a hook conditionally - pass null to switch it off.
7. No god hooks - split by concern, then compose (B9).

Rules of Hooks: top level only - components and hooks only - every timer, listener and request gets a cleanup.

Testing (B10): `renderHook` + `act()` from `@testing-library/react` - fake timers for debounce, real timers wherever `waitFor` polls.