**React Hooks and Their Rules**

React Hooks adalah salah satu fitur paling menarik di React. Mereka **mudah dipelajari pada awalnya**, tetapi untuk menjadi mahir membutuhkan pemahaman yang mendalam tentang cara kerja hooks dan aturan penggunaannya.

**React Hooks**

React Hooks memungkinkan Anda menggunakan state dan fitur React lainnya tanpa menulis komponen berbasis class. Dengan hooks, komponen dapat tetap fungsional namun tetap memiliki kemampuan untuk mengelola state dan efek.

**Overview of All Built-In Hooks**

React memiliki hampir 20 hooks bawaan, tetapi tidak semuanya sering digunakan. Berikut adalah kategorisasi hooks berdasarkan frekuensi penggunaannya:

**Most Used Hooks**

Hooks ini paling sering digunakan dalam pengembangan aplikasi React:

* **useState**: Untuk mengelola state lokal dalam komponen.
* **useEffect**: Untuk menangani efek samping (side effects), seperti fetching data atau berlangganan event.
* **useReducer**: Alternatif untuk useState ketika Anda memiliki logika state yang kompleks.
* **useContext**: Untuk mengakses context React tanpa menggunakan props drilling.

**Less Used Hooks**

Hooks ini lebih jarang digunakan, tetapi penting dalam kasus tertentu:

* **useLayoutEffect**: Mirip dengan useEffect, tetapi dijalankan secara sinkron setelah semua DOM dimutasi.
* **useRef**: Untuk menyimpan referensi yang persisten di antara render.
* **useMemo**: Untuk memcached nilai mahal agar tidak dihitung ulang setiap render.
* **useCallback**: Untuk memcached fungsi agar tidak didefinisikan ulang setiap render.

**Only for Libraries**

Hooks ini dirancang untuk kasus penggunaan khusus, terutama dalam pengembangan library:

* **useSyncExternalStore**: Digunakan untuk berlangganan state luar yang disinkronkan, biasanya dalam library state management.
* **useImperativeHandle**: Untuk memodifikasi instance referensi yang di-ekspos oleh React.forwardRef.

**The Rules of Hooks**

Agar hooks bekerja dengan benar, Anda harus mengikuti dua aturan utama:

**1. Only Call Hooks at the Top Level**

* Hooks tidak boleh dipanggil di dalam loop, kondisi, atau fungsi bersarang.
* Ini karena React bergantung pada urutan panggilan hooks untuk melacak state di antara render.

**2. Only Call Hooks from React Functions**

* Hooks hanya boleh dipanggil di:
  + Komponen fungsional React.
  + Custom hooks (fungsi yang namanya diawali dengan use).

**Hooks Rely on Call Order**

React mengandalkan urutan panggilan hooks untuk mengelola state internalnya. Jika urutan hooks berubah di antara render, React tidak akan tahu bagaimana memetakan state ke hooks yang benar, yang dapat menyebabkan bug yang sulit didiagnosis.

**Contoh Salah**

if (condition) {

useState(); // Tidak boleh dipanggil dalam kondisi

}

**Contoh Benar**

const [state, setState] = useState(); // Dipanggil di level atas

if (condition) {

// Logika lain

}

Dengan mengikuti aturan ini, hooks akan bekerja secara konsisten dan mempermudah pengembangan aplikasi.

**Kesimpulan**

React Hooks memberikan cara yang fleksibel dan modern untuk mengelola state, efek samping, dan banyak lagi. Dengan memahami aturan dan mekanisme hooks, Anda dapat memaksimalkan penggunaannya dalam pengembangan aplikasi React.