

REACT THAM KHẢO NHANH

Component, hook, state, effect, pattern

Cơ Bản JSX

Biểu Thức & Thuộc Tính

```
const name = "Alice";
const el = <h1>Hello, {name}</h1>;
const img = <img src={url} alt="photo" />;
```

Quy Tắc JSX

{expression}	Những biểu thức JS bất kỳ
className	Dùng thay cho 'class'
htmlFor	Dùng thay cho 'for'
style={color: 'red'}	Style inline dưới dạng object
<Component />	Thẻ tự đóng bắt buộc
<> ... </>	Fragment (không tạo DOM node thêm)

Component

Function Component

```
function Greeting({ name }) {
  return <h1>Hello, {name}</h1>;
}
```

```
// Arrow function variant
const Greeting = ({ name }) => (
  <h1>Hello, {name}</h1>
);
```

Props

```
function Card({ title, children }) {
  return (
    <div className="card">
      <h2>{title}</h2>
      {children}
    </div>
  );
}
```

```
<Card title="Welcome">
  <p>Content here</p>
</Card>
```

Props Mặc Định

```
function Button({ label = "Click me", onClick }) {
  return <button onClick={onClick}>{label}</button>;
}
```

State (useState)

State Cơ Bản

```
import { useState } from "react";
```

```
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <button onClick={() => setCount(count + 1)}>
      Count: {count}
    </button>
  );
}
```

Cập Nhật Theo Hàm

```
setCount(prev => prev + 1); // use prev state
setItems(prev => [...prev, newItem]); // arrays
setUser(prev => ({...prev, name: "Bob"})); // objects
```

Quy Tắc State

Cập nhật bất biến	Không bao giờ thay đổi state trực tiếp
Batching bất đồng bộ	Các cập nhật có thể được gộp lại
Dạng hàm	Dùng 'prev =>' khi phụ thuộc vào state trước

Effect (useEffect)

Các Pattern Effect

```
import { useEffect } from "react";

// Run on every render
useEffect(() => { /* ... */ });

// Run once on mount
useEffect(() => { /* ... */ }, []);

// Run when deps change
useEffect(() => { /* ... */ }, [count]);

// Cleanup on unmount
useEffect(() => {
  const id = setInterval(tick, 1000);
  return () => clearInterval(id);
}, []);
```

Danh Sách & Key

```
function TodoList({ items }) {
  return (
    <ul>
      {items.map(item => (
        <li key={item.id}>{item.text}</li>
      ))}
    </ul>
  );
}
```

Key phải là ID ổn định, duy nhất — tránh dùng index mảng làm key

Xử Lý Sự Kiện

Sự Kiện

```
<button onClick={handleClick}>Click</button>
<button onClick={() => handleDelete(id)}>Del</button>
<input onChange={(e) => setVal(e.target.value)} />
<form onSubmit={(e) => {
  e.preventDefault();
  handleSubmit();
}}>
```

Sự Kiện Phổ Biến

onClick	Click chuột
onChange	Giá trị input thay đổi
onSubmit	Gửi form
onKeyDown	Nhấn phím

onFocus / onBlur Focus được / mất focus

onMouseEnter Chuột đi vào phần tử

Render Có Điều Kiện

```
// Ternary
{isLoggedIn ? <Dashboard /> : <Login />}

// Logical AND (short-circuit)
{hasError && <ErrorBanner />}

// Early return
function Page({ user }) {
  if (!user) return <Login />;
  return <Dashboard user={user} />;
}
```

Hook

useRef

```
const inputRef = useRef(null);
// Access: inputRef.current.focus();
<input ref={inputRef} />
```

useRef lưu giá trị qua các lần render mà không kích hoạt re-render

useMemo & useCallback

```
// Memoize expensive computation
const sorted = useMemo(
  () => items.sort(compareFn),
  [items]
);

// Memoize callback
const handleClick = useCallback(
  () => setCount(c => c + 1),
  []
);
```

useContext

```
import { useContext } from "react";
const value = useContext(MyContext);
```

Custom Hook

```
function useLocalStorage(key, initial) {
  const [value, setValue] = useState(() => {
    const saved = localStorage.getItem(key);
    return saved ? JSON.parse(saved) : initial;
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);

  return [value, setValue];
}
```

// Usage
const [name, setName] = useLocalStorage("name", "");

Custom hook phải bắt đầu bằng 'use'

Context API

Tạo & Cung Cấp

```
import { createContext, useContext } from "react";
const ThemeCtx = createContext("light");
```

```
function App() {
  return (
    <ThemeCtx.Provider value="dark">
      <Page />
    </ThemeCtx.Provider>
  );
}
```

Tiêu Thu

```
function Page() {
  const theme = useContext(ThemeCtx); // "dark"
  return <div className={theme}>...</div>;
}
```