



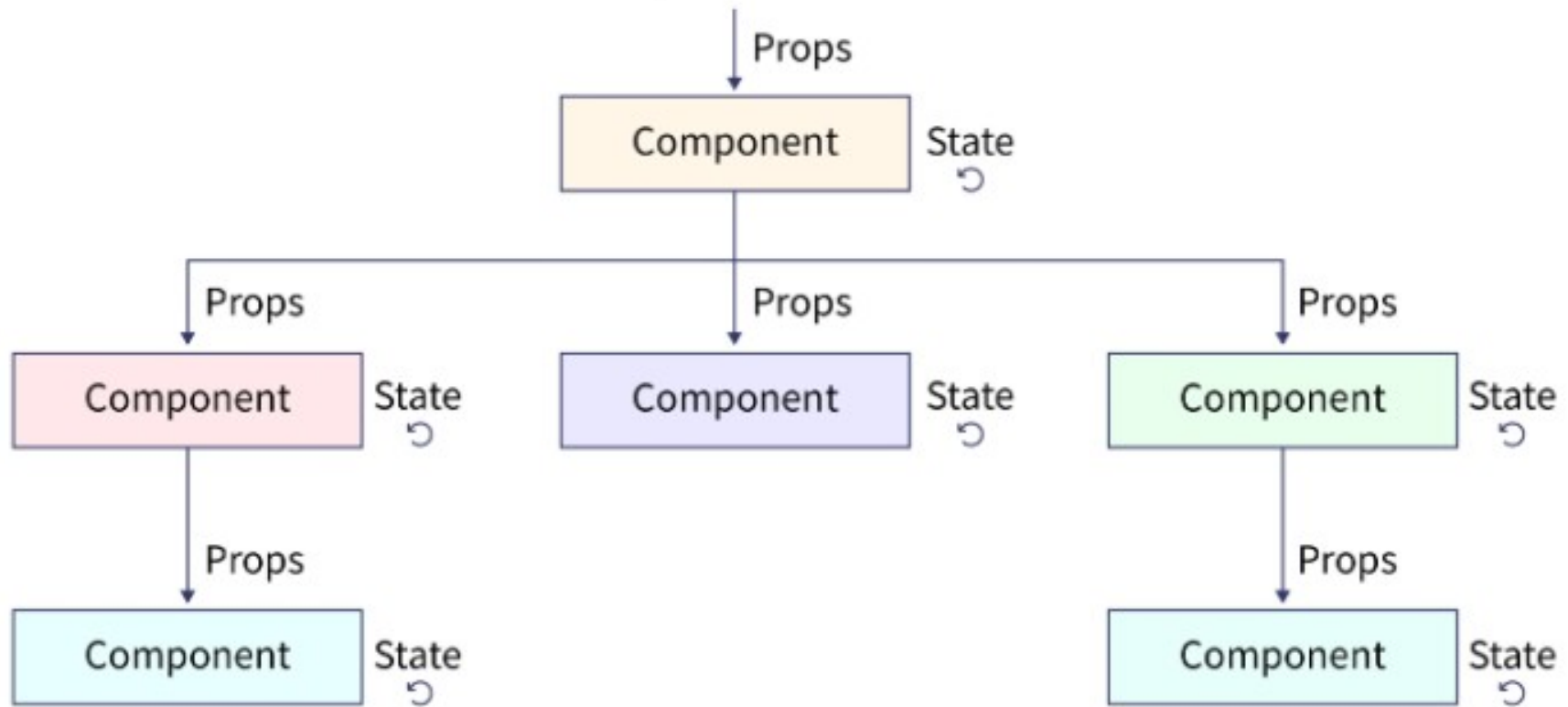
# GIỚI THIỆU REACT STATE



# React State

- State trong React là dữ liệu động mà một component có thể thay đổi trong quá trình chạy của ứng dụng.
- Là trạng thái của một component tại một thời điểm cụ thể và có thể ảnh hưởng đến giao diện và hiển thị của component.

## ReactJs Component State



SCALER  
Topics



# Tại sao cần sử dụng State?

- Cho phép các component React thay đổi dữ liệu và trạng thái của chúng.
- Giúp render lại giao diện khi dữ liệu thay đổi, tạo sự tương tác linh hoạt.
- Dễ dàng quản lý dữ liệu cục bộ của mỗi component mà không ảnh hưởng tới các component khác.



# Sử dụng State trong React

## Khởi tạo State

- Trong React, chúng ta khởi tạo state bằng cách sử dụng hook `useState`. Cú pháp:

```
import React, { useState } from 'react';

function MyComponent() {
  const [stateVariable, setStateVariable] = useState(initialValue);
  // ...
}
```



# Sử dụng State trong React

## Cập nhật State

- Chúng ta sử dụng hàm `useState` để cập nhật giá trị của state. React sẽ tự động gọi lại component và cập nhật giao diện khi state thay đổi.

```
function MyComponent() {  
  const [count, setCount] = useState(0);  
  
  const increment = () => {  
    setCount(count + 1);  
  };  
  
  // ...  
}
```



# Sử dụng State trong R

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  const decrement = () => {
    setCount(count - 1);
  };

  return (
    <div>
      <h1>Counter</h1>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}


export default Counter;
```



# Arrow function

- Arrow function là một cú pháp viết ngắn gọn để định nghĩa hàm trong JavaScript. Chúng được giới thiệu từ ES6 (ECMAScript 2015) và đã trở thành một phần quan trọng của cú pháp lập trình hiện đại.
- Arrow function giúp viết mã ngắn gọn hơn, đặc biệt là trong các trường hợp đơn giản, và cũng giúp làm rõ ngữ cảnh this trong hàm.





```
const functionName = (parameter1, parameter2) => {  
  // Thân hàm  
  return result;  
};
```

```
// Sử dụng function thông thường  
const add = function(a, b) {  
  return a + b;  
};
```

```
// Sử dụng arrow function  
const add = (a, b) => a + b;
```

```
const numbers = [1, 2, 3, 4, 5];  
  
// Sử dụng arrow function trong map  
const doubled = numbers.map(num => num * 2);  
  
// Sử dụng arrow function trong filter  
const evenNumbers = numbers.filter(num => num % 2 === 0);  
  
function Counter() {  
  this.count = 0;  
  
  // Sử dụng arrow function để không bị thay đổi ngữ cảnh this  
  this.increment = () => {  
    this.count++;  
  };  
}
```



# Lý do dùng Arrow function

- Viết mã ngắn gọn: Arrow function giúp viết mã ngắn gọn hơn, đặc biệt là trong các trường hợp hàm đơn giản.
- Không thay đổi ngữ cảnh this: Arrow function giúp tránh sự nhầm lẫn về giá trị của this, đặc biệt trong các tình huống sử dụng callback hoặc nested function.
- Phù hợp cho các hàm không có ngữ cảnh độc lập: Arrow function phù hợp cho các hàm đơn giản, không cần ngữ cảnh this riêng.