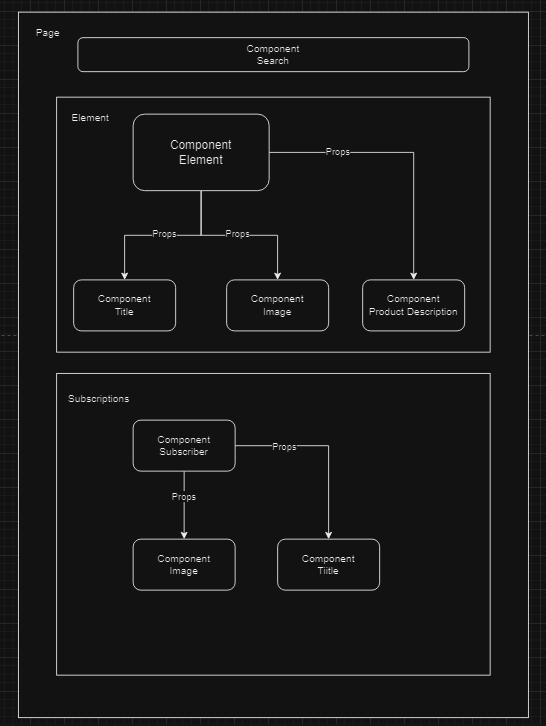
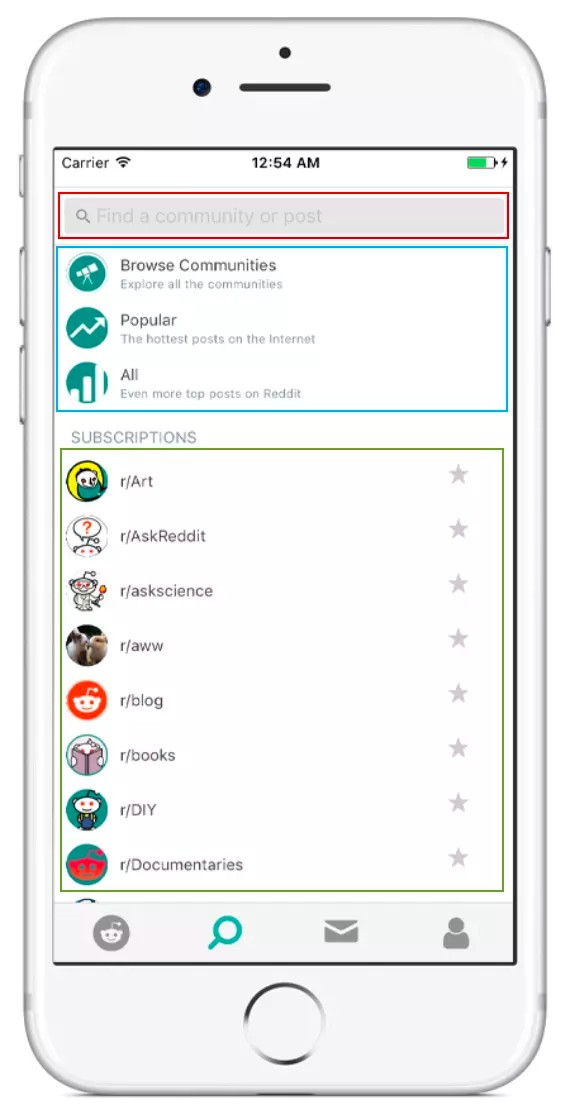
**Day2**

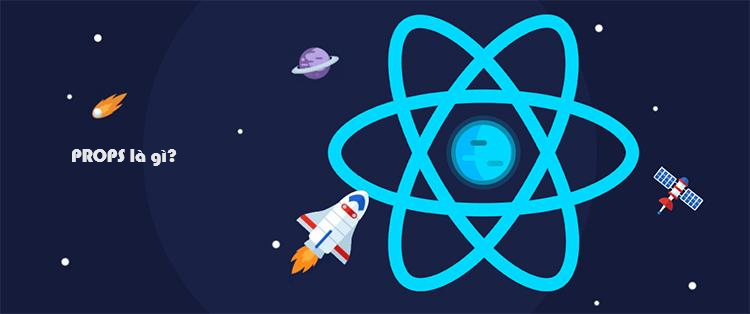
**Props và State trong React Native**

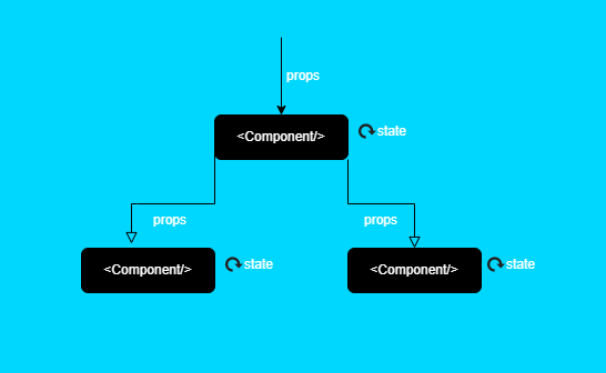


1. **Giới thiệu khái niệm Props và State**

Chào các bạn! Hôm nay, chúng ta sẽ bắt đầu tìm hiểu về hai khái niệm quan trọng trong React Native: "Props" và "State". Đây là những khái niệm không thể thiếu trong việc phát triển ứng dụng React Native. Hãy cùng tìm hiểu cách chúng hoạt động và tại sao chúng lại quan trọng đặc biệt.

1. **Props là gì?**





Props trong React Native có thể giải thích một cách dễ hiểu như việc bạn chia sẻ thông tin khi gặp bạn bè. Khi bạn chia sẻ thông tin, bạn giúp họ biết điều gì đang xảy ra và họ có thể tương tác dựa trên thông tin đó.

Tương tự, Props cho phép bạn truyền dữ liệu từ component cha đến component con. Dữ liệu thông qua Props thường là không thay đổi và được quản lý bởi component cha. Props giúp chúng ta truyền thông tin một cách dễ dàng và tạo sự tương tác giữa các component.

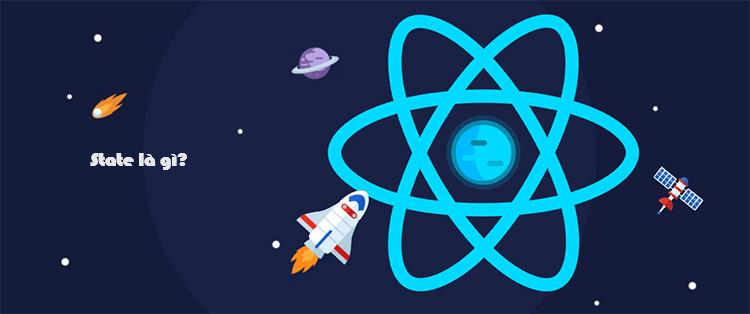
Hãy xem cách chúng ta có thể sử dụng Props trong React Native thông qua ví dụ cụ thể sau đây!



Trong component cha "ParentComponent," chúng ta gọi component con "ChildComponent" và truyền vào ba thông tin: "name," "age," và "className." Các thông tin này được gửi qua Props.

Trong component con "ChildComponent," khi nhận được các dữ liệu truyền vào từ Component cha, tương tự như cách chúng ta truyền dữ liệu vào một hàm. Sau khi nhận được dữ liệu từ Props, bạn có thể sử dụng props.name để hiển thị tên, props.age để hiển thị tuổi, và props.className để hiển thị tên lớp.

1. **State là gì?**

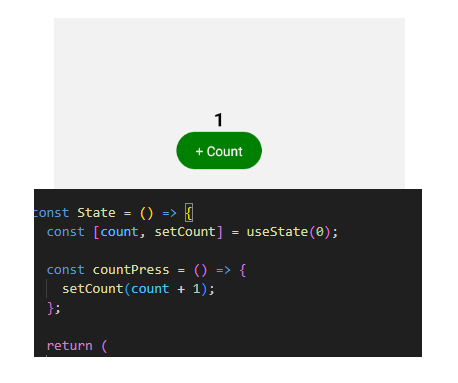


State trong React Native giống như tâm trạng của một người bạn vậy. Khi bạn thay đổi tâm trạng của họ, giao diện xung quanh họ sẽ tự động thay đổi để phản ánh điều đó.

Tương tự như vậy State cho phép lưu trữ thông tin có thể thay đổi trong ứng dụng, khi có sự thay đổi, React Native sẽ tự động cập nhật giao diện.

State thường được sử dụng khi chúng ta muốn theo dõi và cập nhật các giá trị theo thời gian.

Để hiểu hơn về cách State hoạt động trong React Native, hãy xem ví dụ cụ thể sau đây.



Trong component "State," chúng ta sử dụng hook useState để tạo một biến State có tên là "count," với giá trị ban đầu là 0. Biến "count" dùng để lưu số lần người dùng nhấn nút.

Có một hàm "countPress" được kích hoạt khi người dùng nhấn nút. Hàm này sử dụng setCount để cập nhật giá trị của biến State "count," tăng lên một đơn vị.

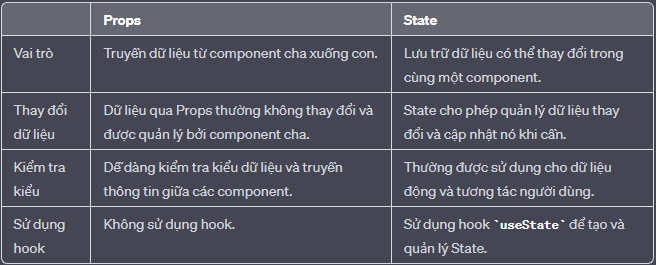
Trên giao diện của component, chúng ta sử dụng thẻ <Text> để hiển thị giá trị của biến State "count."

Một nút <TouchableOpacity> hiển thị với chữ "+ Count." Khi người dùng nhấn nút này, hàm "countPress" được gọi để tăng giá trị của biến State "count."

Bây giờ hãy thử kiểm tra kiến thức của bạn về State qua một bài trắc nghiệm nhỏ!

Vậy Props và State có gì khác nhau? Chúng ta hãy so sánh chúng để cùng hiểu rõ hơn về chúng nhé!

1. **So sánh Props và State**



Về·Vai trò Props truyền dữ liệu từ component cha xuống con còn State sẽ Lưu trữ dữ liệu có thể thay đổi trong cùng một component.

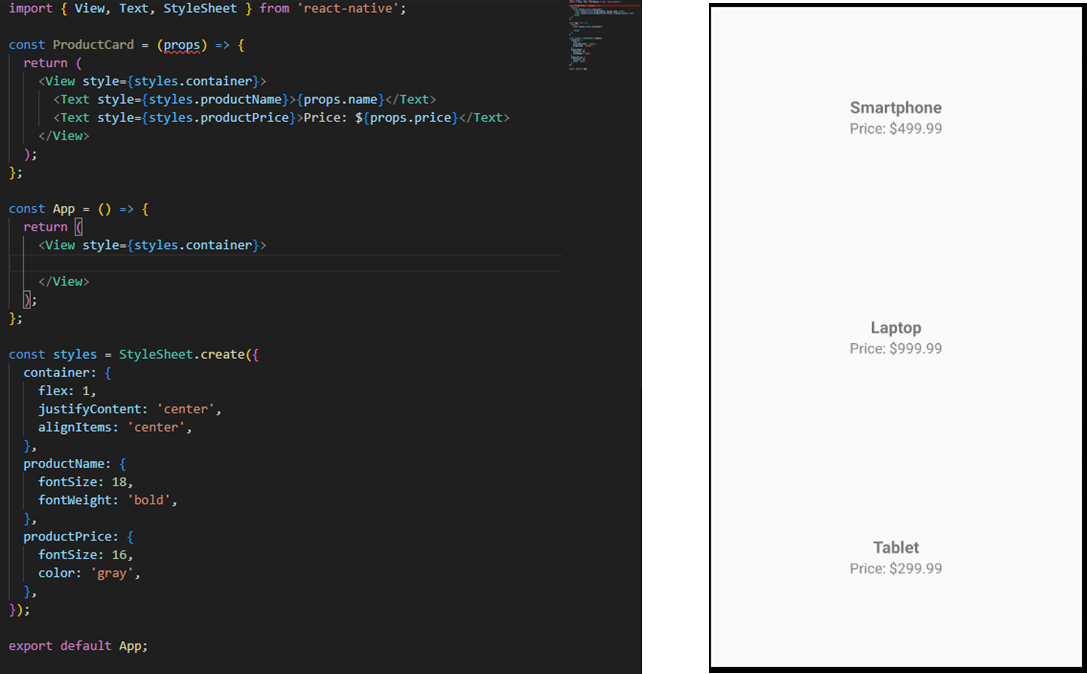
Về Thay đổi dữ liệu Props có dữ liệu qua Props thường không thay đổi và được quản lý bởi component cha còn State cho phép quản lý dữ liệu thay đổi và cập nhật nó khi cần.

Về Kiểm tra kiểu Props Dễ dàng kiểm tra kiểu dữ liệu và truyền thông tin giữa các component còn State Thường được sử dụng cho dữ liệu động và tương tác người dùng.

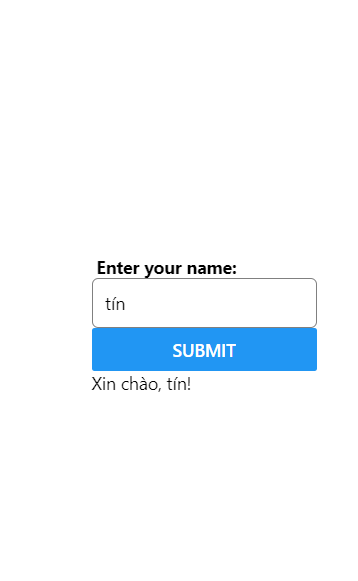
Cuối cùng là về hook Props Không sử dụng hook nhưng State lại sử dụng hook useState để tạo và quản lý State.

Tóm lại, cả Props và State đều đóng vai trò quan trọng trong phát triển ứng dụng React Native. Props giúp truyền thông tin và tương tác giữa các component, trong khi State quản lý dữ liệu có thể thay đổi và phản ánh tương tác người dùng trong thời gian thực.

Thông qua hai khái niệm Props và State, bạn đã nắm được cách chúng hoạt động. Hãy cùng làm một số bài tập nhỏ để áp dụng kiến thức về Props và State.

1. **Bài tập về Props** 

**Luồng chạy:**

1. Bắt đầu bằng việc định nghĩa functional component ProductCard và định nghĩa các props mà nó sẽ nhận (trong trường hợp này là name và price).
2. Khi sử dụng component ProductCard trong component cha App, các giá trị được truyền thông qua props name và price.
3. Component ProductCard sử dụng các giá trị name và price được truyền qua props để hiển thị tên và giá sản phẩm.
4. Giao diện của component ProductCard sẽ hiển thị tên sản phẩm và giá sản phẩm, sử dụng các giá trị từ props.
5. Trong component cha App, một instance của component ProductCard được tạo ra, nhưng không có nội dung hiển thị.
6. **Bài tập về State**
7. 
8. 

**Luồng chạy**:

1. Đầu tiên, component App được gọi và render.
2. Trong phần giao diện của component App, một instance của component GreetingForm được gọi và render.
3. Trong component GreetingForm, trong phương thức constructor, state ban đầu được khởi tạo với name và greeting có giá trị rỗng.
4. Giao diện của GreetingForm hiển thị một TextInput để nhập tên và một Button để gọi phương thức handleSubmit.
5. Khi người dùng nhập tên vào TextInput và nhấn nút "Submit", sự kiện onPress của Button được gọi, kích hoạt phương thức handleSubmit.
6. Trong phương thức handleSubmit, state greeting được cập nhật bằng thông báo chào mừng được tạo dựa trên giá trị name trong state.
7. Giao diện của GreetingForm được tái render, hiển thị thông báo chào mừng từ state greeting.
8. Quay lại component App, giao diện của GreetingForm đã được render và hiển thị.
9. Tương tự, một instance khác của GreetingForm cũng được gọi và render để hiển thị thông báo chào mừng cho tên khác.

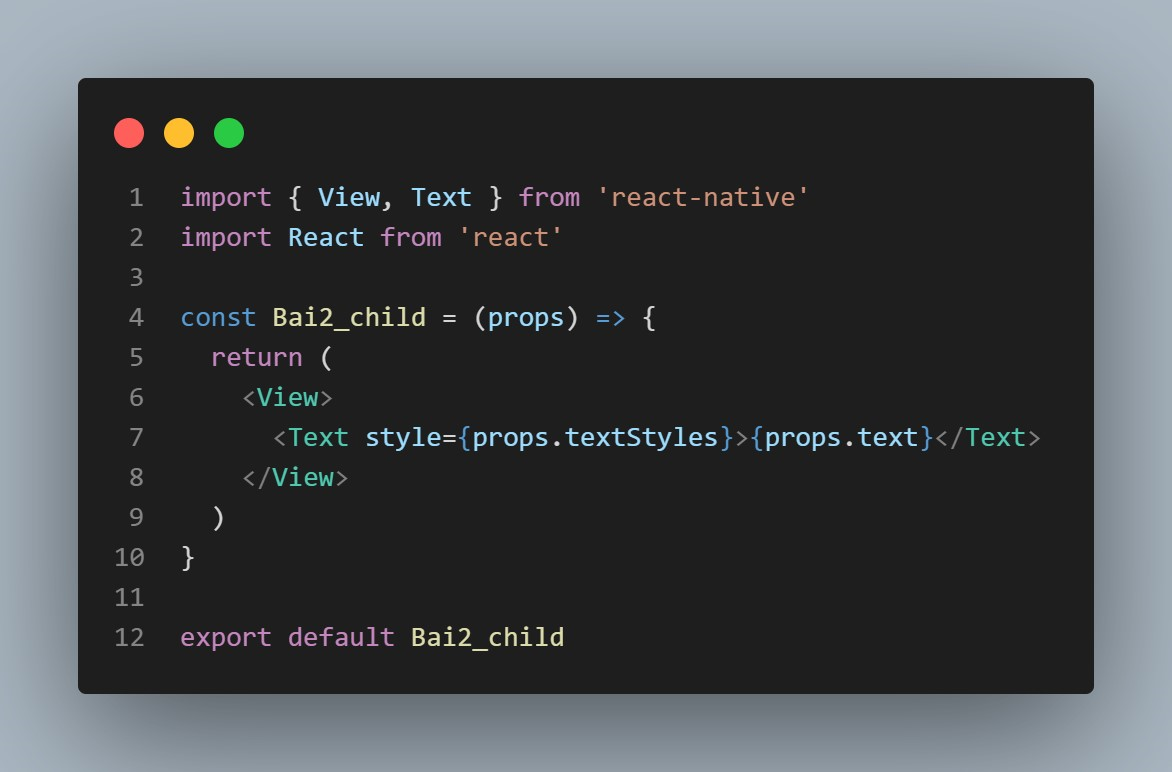
**Bài tập 1**: Tạo một màn hình sử dụng các component trên sử dụng props truyền tên, mssv, lớp và hiển thị lên màn hình

**Bài tập 2**: Tạo một màn hình sử dụng các component trên tạo ra nhiều component con sử dụng props truyền styles để thay đổi giao

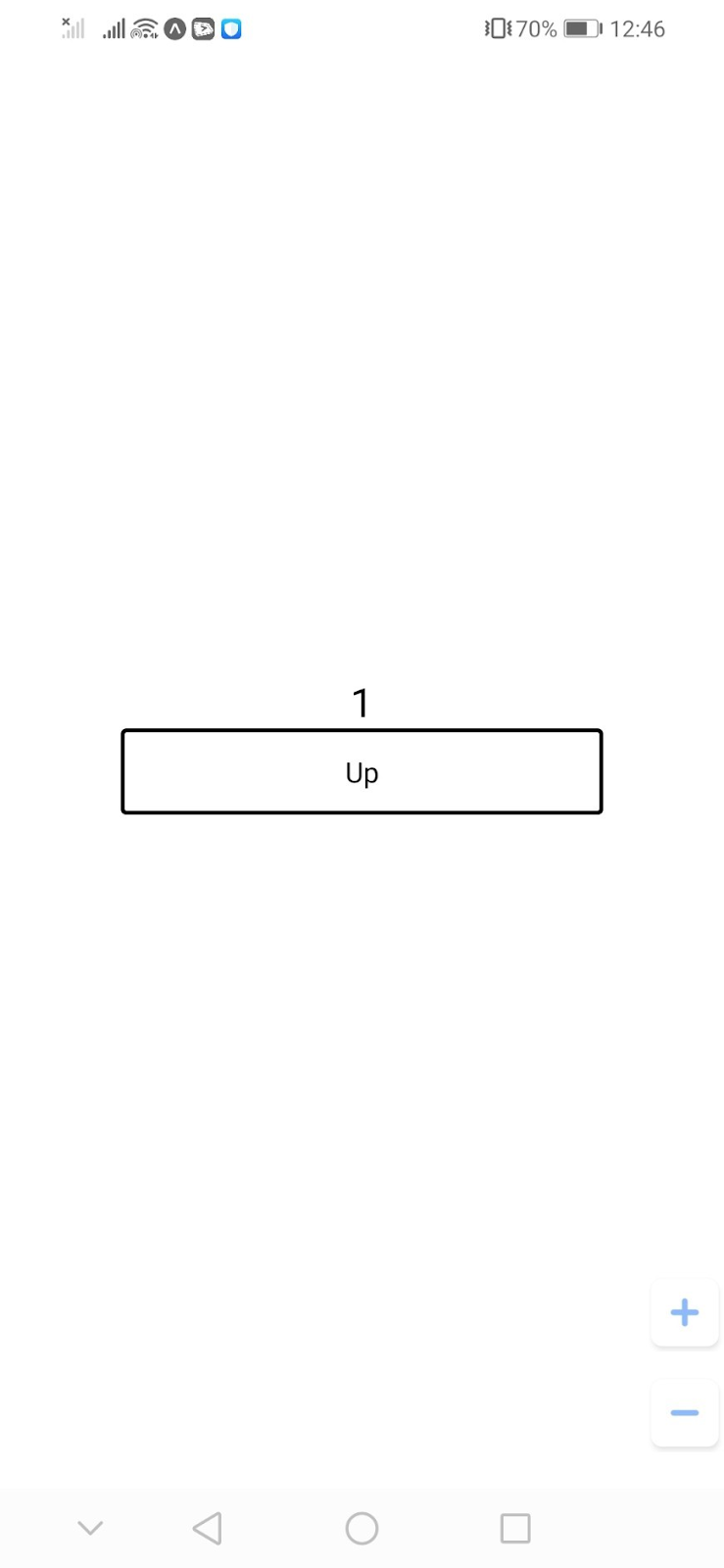
diện màn hình







**Bài tập 3**: Tạo một màn hình sử dụng các component trên sử dụng props truyền function để thay đổi thông số trên component cha







**Bài tập 4**: Tạo một màn hình sử dụng các component trên sử dụng props truyền state để tính tổng giá trị ở component con

