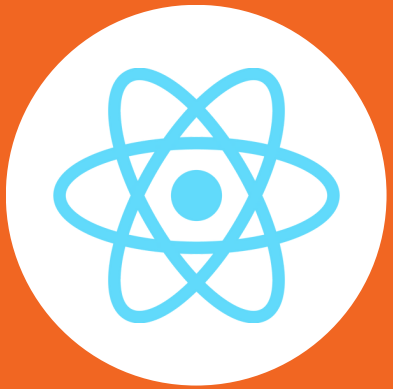
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 3: CORE COMPONENT TRONG REACT NATIVE (P1)

PHẦN 1: CÁC THÀNH PHẦN CƠ BẢN

Mục tiêu

- Giới thiệu chung về component trong React Native
- Xây dựng vùng chứa với **<View/>**
- Hiển thị văn bản với **<Text/>**
- Nhập văn bản với **<TextInput/>**

Giới thiệu component

- Giới thiệu chung về component trong React Native:

  Trong React Native, các thành phần (component) được sử dụng để tạo nên giao diện (UI) cho ứng dụng.

  Component trong React Native được tạo ra bằng cách kế thừa từ class hoặc hàm (function) và định nghĩa các thuộc tính và phương thức để xây dựng giao diện. Mỗi component sẽ chứa một phần của giao diện và có thể bao gồm các thành phần khác. Các thành phần được sử dụng trong React Native được chia thành hai loại: các thành phần cơ bản (**basic components**) và các thành phần nâng cao (**advanced components**). Các thành phần cơ bản là các thành phần đơn giản như **Text, View, Image, TextInput, Button, ScrollView, FlatList**,... Các thành phần nâng cao là các thành phần được xây dựng trên các thành phần cơ bản, như **Modal, Picker, WebView, Slider**,...

Giới thiệu component

Mỗi thành phần có thể có các thuộc tính (**props**) để cấu hình và các phương thức để xử lý sự kiện và thay đổi trạng thái. Khi một thành phần được cập nhật, React Native sẽ tự động cập nhật lại giao diện để phản ánh các thay đổi.

Việc sử dụng các thành phần trong React Native giúp cho việc phát triển ứng dụng di động trở nên dễ dàng hơn và nhanh chóng hơn bằng cách sử dụng các thành phần đã được định nghĩa trước đó.

View component

- Hiển thị vùng chứa với **<View/>**

Thành phần cơ bản nhất để xây dựng giao diện người dùng, **View** là một vùng chứa hỗ trợ bố cục với **flexbox**, **style**, một số xử lý chạm và accessibility . **View** truy cập trực tiếp đến chế độ xem native tương đương trên bất kỳ nền tảng nào React Native đang chạy, cho dù đó là **UIView**, ** **, **android.view**, v.v.

View được thiết kế để lồng vào bên trong các View khác và có thể có từ 0 đến nhiều child thuộc bất kỳ loại nào.

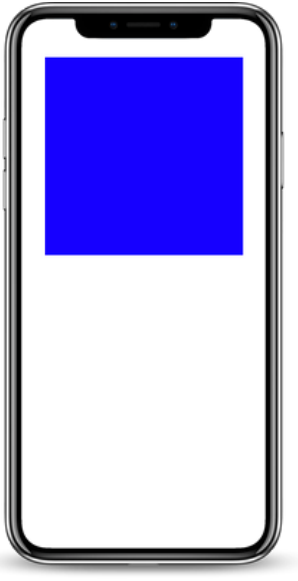
View component

- Props sử dụng phổ biến trong **<View/>**
  - **style:** chỉnh sửa giao diện của **View**
  - **onLayout:** được gọi khi UI khởi tạo lần đầu và layout thay đổi

    Sự kiện này được kích hoạt ngay lập tức sau khi bố cục đã được tính toán, nhưng bố cục mới có thể chưa được phản ánh trên màn hình tại thời điểm nhận được sự kiện, đặc biệt nếu animation đang diễn ra.

View component

- Ở ví dụ bên dưới, ta sẽ tạo một vùng chứa như hình dưới đây



View component

- Để dùng **View** ta cần:
  - Import thẻ **<View/>** từ 'react-native'

```
import React from 'react';
import {View, Text} from 'react-native';
```

View component

- Thẻ **<View/>** dùng để tạo một vùng chứa component, cách sử dụng như bên dưới

```
const ViewComponent = () => {
  return (
    <View
      style={{
        backgroundColor: 'blue',
         height: 200,
         width: 200,
         alignSelf: 'center',
         marginTop: 20
        }}>

    </View>
  );
};

export default ViewComponent;
```

View component

Lưu ý: Chúng ta có 2 cách để export **ViewComponent** ở bên trên

```
const ViewComponent = () => {
  return (
    <View
      style={{
        backgroundColor: 'blue',
         height: 200,
         width: 200,
         alignSelf: 'center',
         marginTop: 20
        }}>

    </View>
  );
};

export default ViewComponent;
```

Cách 1

```
export const ViewComponent = () => {
  return (
    <View
      style={{
        backgroundColor: 'blue',
         height: 200,
         width: 200,
         alignSelf: 'center',
         marginTop: 20
        }}>

    </View>
  );
};
```

Cách 2

View component

- Vậy sự khác biệt khi export component theo 2 cách này là gì?

  - Cách 1: Khi bạn export component theo cách này. Ở một file khác, nếu bạn muốn import ViewComponent vào, thì bạn phải sử dụng
    import ViewComponent from '...'

  - Cách 2: Khi bạn export component theo cách này. Ở một file khác, nếu bạn muốn import ViewComponent vào, thì bạn phải sử dụng
    import { ViewComponent } from '...'

Text component

- Hiển thị văn bản với **<Text/>**

  Text hỗ trợ lồng nhau, **styling** và xử lý chạm.

- Props sử dụng phổ biến trong **<Text/>**
  - **style:** chỉnh sửa giao diện của **Text**
  - **onPress:** được gọi khi nhấn người dùng, được kích hoạt sau onPressOut.
  - **onLongPress:** được gọi khi nhấn và giữ

- Chúng ta sẽ đi vào ví dụ, tạo một đoạn văn dưới đây

Text component

- Chúng ta tiếp tục sử dụng bài từ phần trước để tiếp tục
  - Import một số thành phần mới
  - Lưu ý: bài demo hiện tại đang lập trình bằng **TypeScript**, các phiên bản mới nhất của React Native cũng đã tự động tạo project bằng **Typescript** cho chúng ta. Nhận biết đuôi file có **.tsx**

```
import React from 'react';
import {StyleSheet, Text, TextStyle, View} from 'react-native';
```

Text component

- Viết styles cho **ViewComponent,** đây là một trong những cách để truyền giá trị vào style. Bình thường các bạn sẽ style component bằng **StyleSheet**

```
// Truyền prop vào style để custom size tuỳ ý
const sizeText = (size: number) => ({
  fontSize: size,
});

const colorText = (color: string) => ({
  color,
});

// styleText này sẽ nhận tất cả props style mà th
const styleText = (props: TextStyle) => ({
  ...props,
});
```

Text component

- Code giao diện như yêu cầu bên trên

```
const ViewComponent = () => {
  return (
    <View style={styles.container}>
      {/* Line 1 */}
      <Text style={styles.baseText}>
        Em vào đời bằng{' '}
        <Text style={[styles.boldText, colorText('red')]}>vang đỏ</Text> anh vào
        đời bằng{' '}
        <Text style={[styles.boldText, colorText('yellow')]}>nước trà</Text>
      </Text>

      {/* Line 2 */}
      <Text style={styles.baseText}>
        Bằng cơn mưa thơm{' '}
        <Text style={[styles.boldText, sizeText(20), styles.italicText]}>
          mùi đất{' '}
        </Text>{' '}
        và{' '}
        <Text style={[sizeText(10), styles.italicText]}>
          bằng hoa dại mọc trước nhà
        </Text>
      </Text>
```

- Reload app và ta có kết quả như sau

Góc giải đáp

TextStyle là gì?

TextStyle là một kiểu dữ liệu được sử dụng trong React Native để mô tả các thuộc tính của văn bản. Nó được sử dụng để xác định các thuộc tính như màu sắc, kích thước, font chữ, độ đậm nhạt, khoảng cách giữa các chữ, v.v. của văn bản trong các thành phần

Tại sao không nên style trực tiếp bên trong thẻ <Text /> hay <View /> ?

- Không thể tái sử dụng mã code
- Làm giảm hiệu suất ứng dụng
- Khó kiểm soát, dễ gây lỗi

TextInput component

- Nhập văn bản với **<TextInput/>**

TextInput dùng để nhập văn bản vào ứng dụng thông qua bàn phím. **TextInput** cung cấp khả năng định cấu hình cho một số tính năng, chẳng hạn như tự động sửa, tự động viết hoa, văn bản chỗ dành sẵn và các loại bàn phím khác nhau, chẳng hạn như bàn phím số.

Cách sử dụng cơ bản nhất của **TextInput** là đăng ký các sự kiện **onChangeText** để đọc đầu vào của người dùng. Ngoài ra còn có các sự kiện khác, chẳng hạn như **onSubmitEditing** và **onFocus** có thể được đăng ký.

TextInput component

- Props sử dụng phổ biến trong **<TextInput/>**

  - **defaultValue:** Cung cấp giá trị ban đầu sẽ thay đổi khi người dùng bắt đầu nhập. Hữu ích cho các trường hợp sử dụng mà bạn không muốn xử lý việc nghe các sự kiện và cập nhật giá trị prop để giữ cho trạng thái được kiểm soát đồng bộ.

  - **inputMode:** Hoạt động giống như thuộc tính **inputmode** trong HTML, nó xác định bàn phím nào sẽ mở, ví dụ: số và được ưu tiên hơn **keyboardType**.

    Hỗ trợ các giá trị sau:
    - **none**
    - **text**
    - **decimal**
    - **numeric**

TextInput component

- **tel**
- **search**
- **email**
- **url**

- **keyboardType:** Xác định bàn phím nào sẽ mở, ví dụ: số.

  Các giá trị sau hoạt động trên cả **ios** và **android:**
  - **default**
  - **number-pad**
  - **decimal-pad**
  - **numeric**
  - **email-address**
  - **phone-pad**
  - **url**

TextInput component

- **multiline:** Nếu đúng, đầu vào văn bản có thể là nhiều dòng. Giá trị mặc định là **false**.
- **onChangeText:** Callback được gọi khi văn bản nhập thay đổi. Văn bản đã thay đổi được chuyển dưới dạng một prop chuỗi cho Callback.
- **placeholder:** Chuỗi sẽ được hiển thị trước khi nhập văn bản.
- **value:** Giá trị để hiển thị cho đầu vào văn bản. **TextInput** là một thành phần được kiểm soát, có nghĩa là giá trị gốc sẽ buộc phải khớp với giá trị này nếu được cung cấp.

TextInput component

- Chúng ta sẽ tạo một vài **<TextInput />** như hình dưới đây

TextInput component

- Chúng ta tiếp tục mở rộng bài từ phần trước
  - Import thêm **useState** và **TextInput**

```
import React, {useState} from 'react';
import {StyleSheet, Text, TextInput, TextStyle, View} from 'react-native';
```

  - Thêm một thẻ **<View />** bọc ngoài cùng **ViewComponent** của chúng ta

```
const ViewComponent = () => {
  return (
    <View>
      <View style={styles.container}>
        {/* Line 1 */}
```

TextInput component

- Khai báo 3 biến useState để lưu trữ giá trị của **TextInput**

```
const ViewComponent = () => {
  const [name, setName] = useState('');
  const [phone, setPhone] = useState('');
  const [password, setPassword] = useState('');
```

TextInput component

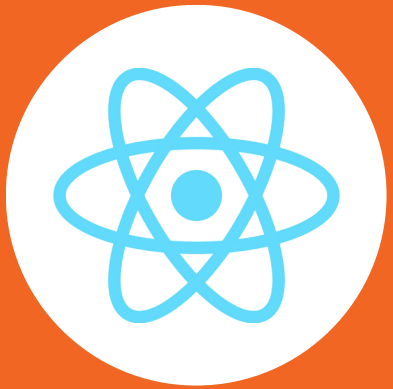
- Sử dụng **TextInput** trong **React Native**
- Dùng **TextInput** để nhập họ tên, số điện thoại và mật khẩu

```
<TextInput
  value={name}
  onChangeText={setName}
  placeholder="Nhập họ tên"
  style={styles.tipStyle}
/>
<TextInput
  value={phone}
  onChangeText={setPhone}
  placeholder="Nhập số điện thoại"
  keyboardType="phone-pad"
  style={styles.tipStyle}
/>
<TextInput
  value={password}
  onChangeText={setPassword}
  placeholder="Nhập mật khẩu"
  secureTextEntry={true}
  style={styles.tipStyle}
/>
```

- Reload ứng dụng và chúng ta có kết quả như sau

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 3: CORE COMPONENT TRONG REACT NATIVE (P1)

PHẦN 2 CÁC THÀNH PHẦN CƠ BẢN (TT)

Mục tiêu

- Hiển thị hình ảnh với **<Image/>**
- Giới thiệu về component chạm **<Pressable/>**
- Giới thiệu về component chạm **<TouchableOpacity/>**
- Giới thiệu về **<Modal/>**

Image component

- Hiển thị hình ảnh với **<Image />**

  Một component trong React để hiển thị các loại hình ảnh khác nhau, bao gồm hình ảnh mạng, tài nguyên tĩnh, hình ảnh cục bộ tạm thời và hình ảnh từ đĩa cục bộ, chẳng hạn như thư viện ảnh.

- Các props sử dụng phổ biến của **<Image />**

  - **defaultSource:** Một hình ảnh tĩnh để hiển thị trong khi tải nguồn hình ảnh.
  - **height:** Chiều cao của thành phần hình ảnh.
  - **width:** Width of the image component.
  - **resizeMode:** Xác định cách thay đổi kích thước hình ảnh khi khung không khớp với kích thước hình ảnh thô. Mặc định để che.

Image component

- **cover:** Chia tỷ lệ hình ảnh đồng đều (duy trì tỷ lệ khung hình của hình ảnh) sao cho cả hai kích thước (chiều rộng và chiều cao) của hình ảnh sẽ bằng hoặc lớn hơn kích thước tương ứng của chế độ xem (trừ phần đệm) ít nhất một chiều của hình ảnh được chia tỷ lệ sẽ bằng với kích thước tương ứng của chế độ xem (trừ phần đệm)
- **contain:** Chia tỷ lệ hình ảnh đồng đều (duy trì tỷ lệ khung hình của hình ảnh) sao cho cả hai kích thước (chiều rộng và chiều cao) của hình ảnh sẽ bằng hoặc nhỏ hơn kích thước tương ứng của chế độ xem (trừ phần đệm).
- **stretch:** Thay đổi chiều cao, rộng của hình ảnh mà không dựa vào tỷ lệ khung hình

Image component

- **repeat:** Lặp lại hình ảnh để che khung của chế độ xem. Hình ảnh sẽ giữ kích thước và tỷ lệ khung hình của nó, trừ khi nó lớn hơn chế độ xem, trong trường hợp đó, nó sẽ được thu nhỏ đồng đều để nó được chứa trong chế độ xem.
- **center:** Căn giữa hình ảnh trong chế độ xem dọc theo cả hai chiều. Nếu hình ảnh lớn hơn chế độ xem, hãy thu nhỏ nó xuống một cách thống nhất để nó được chứa trong chế độ xem.

- **source:** Nguồn hình ảnh (URL từ xa hoặc tài nguyên tệp cục bộ).

  Prop này cũng có thể chứa một số URL, được lấy cùng với chiều rộng và chiều cao của chúng và có khả năng sử dụng với URI khác. Sau đó, phía native sẽ chọn uri tốt nhất để hiển thị dựa trên kích thước đo được của vùng chứa hình ảnh.

Image component

Các định dạng hiện được hỗ trợ là png, jpg, jpeg, bmp, gif, webp, psd (chỉ dành cho iOS). Ngoài ra, iOS hỗ trợ một số định dạng ảnh RAW. Tham khảo tài liệu của Apple để biết danh sách các mẫu máy ảnh được hỗ trợ hiện tại (đối với iOS 12, xem https://support.apple.com/en-ca/HT208967).

- **src:** Một chuỗi đại diện cho URL từ xa của hình ảnh. Đạo cụ này được ưu tiên hơn đạo cụ nguồn.
- **style:** mage Style Props, Layout Props, Shadow Props, Transforms

Image component

- Bây giờ chúng ta sẽ đi vào ví dụ, tạo một hình ảnh dưới đây:

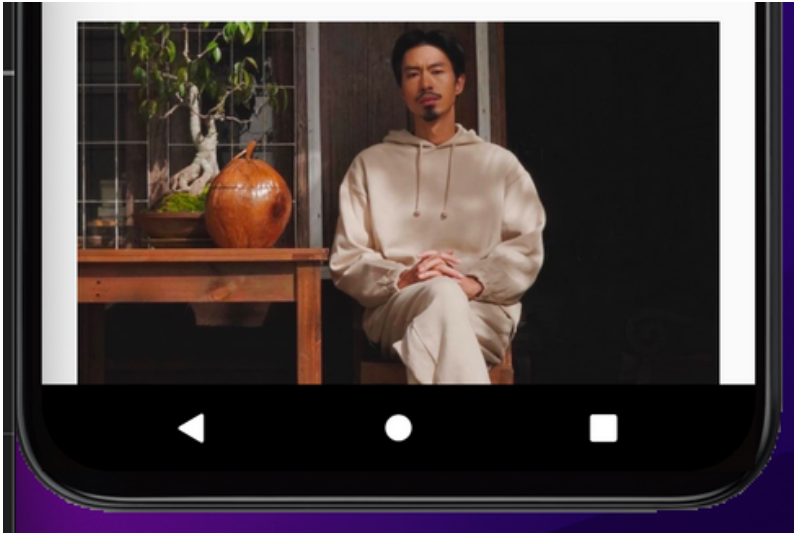


Image component

- Thêm style tên containerImg cho thẻ **<Imgage />**

```
  btnClearTip: {
    marginTop: 10,
    marginHorizontal: 15,
  },
  containerImg: {
    width: '90%',
    height: 200,
    resizeMode: 'cover',
    alignSelf: 'center',
    marginTop: 20,
  },
```

- Import **Image** từ react native

```
import React, {useState} from 'react';
import {Image} from 'react-native';
```

Image component

- Thêm uri image của bạn vào source **<Image />**

```
<Image
  source={{
    uri: 'https://znews-photo.zadn.vn/w660/Uploaded/qfssu/2020_12_17/
    131257110_1569981593189155_2296861462530926702_o_1.jpg',
  }}
  style={styles.containerImg}
/>
</ScrollView>
```

Image component

- Kết quả sẽ như hình bên dưới

- Khi sử dụng dụng thẻ **<Image />**, bạn bắt buộc phải thêm style **width** và **height** thì hình ảnh mới hiện lên

Pressable component

- Tạo nút với **<Pressable />**

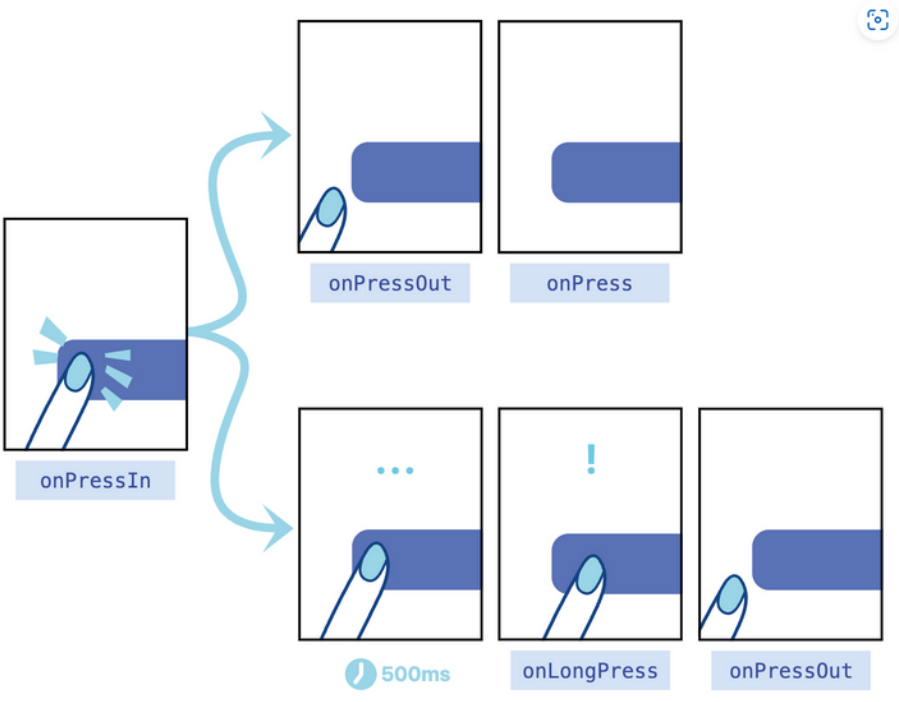
Pressable là một core component có thể phát hiện các giai đoạn tương tác nhấn khác nhau trên bất kỳ children component nào được xác định của nó.

Ngoài **Pressable**, React Native cung cấp cho chúng ta nhiều loại nút nhấn khác nhau như **TouchableOpacity**, **TouchableWithoutFeedback, TouchableHighlight..., .** Các nút này có cách sử dụng tương tự nhau, và có một vài điểm khác biệt nhỏ. Do nội dung bài học này đã quá dài nên chúng ta chỉ đi tìm hiểu nút <**Pressable />**

Pressable component

- Cách hoạt động của **<Pressable />**
  - Trên một phần tử được bao bọc bởi **Pressable**:
    - **onPressIn** được gọi khi nhấn vào
    - **onPressOut** được gọi thả nút ra
  - Sau khi nhấn **onPressIn**, một trong hai điều sẽ xảy ra:
    1. Người đó sẽ thả ngón tay của họ, kích hoạt **onPressOut** theo sau là onPress.
    2. Nếu người đó giữ ngón tay của họ lâu hơn 500 mili giây trước khi thả tay ra, **onLongPress** sẽ được kích hoạt. (**onPressOut** sẽ vẫn kích hoạt khi họ bỏ ngón tay ra.)

Pressable component



Pressable component

- Các prop sử dụng phổ biến trong **<Pressable />**
  - **disabled:** Vô hiệu hoá nút.
  - **onPress:** Được gọi sau **onPressOut.**
  - **style:** tuỳ chỉnh giao diện cho nút

Pressable component

- Chúng ta sẽ đi vào ví dụ, tạo một nút bằng **<Pressable />**

```
<Pressable onPress={onPressFunction}>
  <Text>I'm pressable!</Text>
</Pressable>
```

Modal component

- **Modal** là một cách cơ bản để trình bày nội dung trên chế độ xem kèm theo. Bạn có thể tưởng tượng nó như là **Dialog** bạn đã học ở bên môn Android

- Các Props sử dụng phổ biến của **Modal:**

  - **animationType:** là animation khi Modal xuất hiện

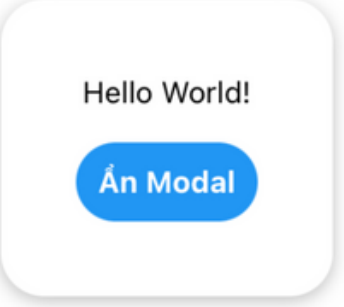
    - **slide** modal xuất hiện từ dưới cùng.
    - **fade** mờ dần vào tầm nhìn,
    - **none** xuất hiện mà không có animation.

  - **transparent:** prop transparent xác định liệu phương thức của bạn có lấp đầy toàn bộ chế độ xem hay không. Đặt điều prop này thành true sẽ hiển thị phương thức trên nền trong suốt.

Modal component

- **visible:** dùng để bật tắt modal

Bây giờ chúng ta sẽ đi vào ví dụ tạo một Modal:



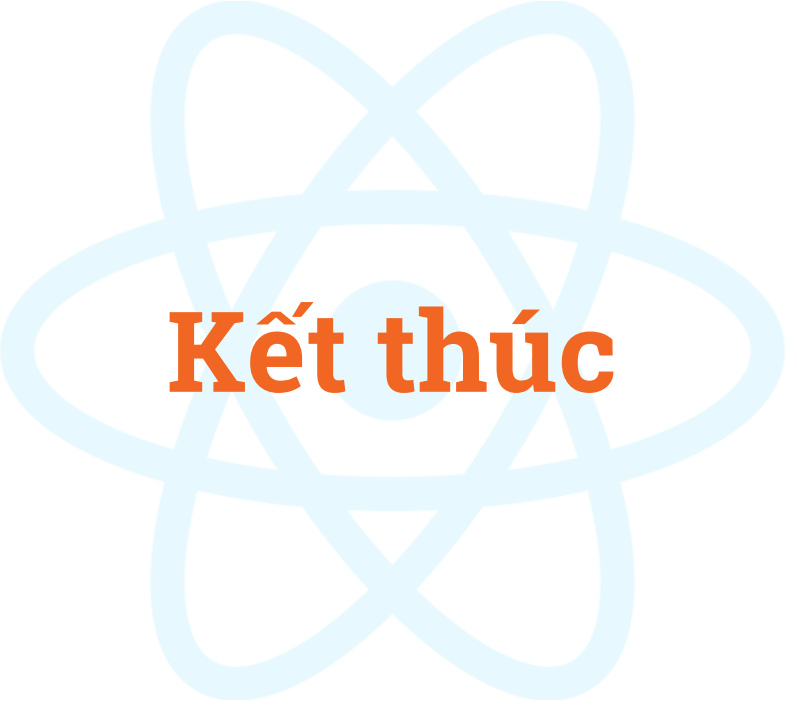
Modal component

- **visible:** đoạn code mẫu sẽ giống như bên dưới

```
<Modal
  animationType="slide"
  transparent={true}
  visible={modalVisible}
  onRequestClose={() => {…
  }}>
  <View style={styles.centeredView}>…
  </View>
</Modal>
```

Tổng kết

- Giới thiệu chung về component trong React Native
- Xây dựng vùng chứa với **<View/>**
- Hiển thị văn bản với **<Text/>**
- Nhận văn bản với **<TextInput/>**
- Hiển thị hình ảnh với **<Image/>**
- Giới thiệu về component chạm **<Pressable/>**
- Giới thiệu về component chạm **<TouchableOpacity/>**
- Giới thiệu về **<Modal/>**



Kết thúc