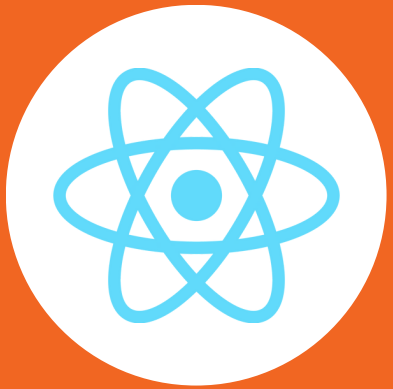
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 7: BOTTOM VÀ TOP TABS NAVIGATOR TRONG REACT NATIVE

PHẦN 1: GIỚI THIỆU VỀ BOTTOM TABS NAVIGATOR

Mục tiêu

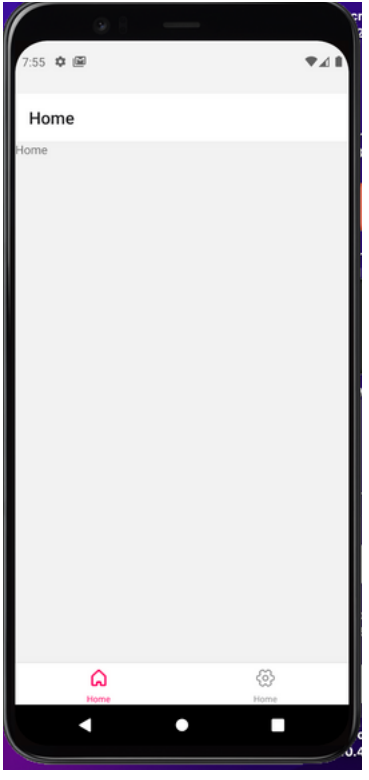
- [ ] Biết cách xây dựng bottom tabs navigator
- [ ] Chỉnh sửa giao diện cho bottom tabs

Giới thiệu bottom tabs navigator

Bottom Tabs Navigator

Một thanh tab đơn giản ở cuối màn hình cho phép bạn chuyển đổi giữa các route khác nhau. Các route được khởi tạo một cách lazily - các thành phần màn hình của chúng không được mount cho đến khi chúng được nhấn vào lần đầu tiên.

Giới thiệu bottom tabs navigator



- Một **bottom tabs navigator** sẽ có giao diện giống như hình này.

Cài đặt thư viện

- Để sử dụng **bottom tabs navigator** chúng ta sẽ phải cài đặt thêm thư viện, dưới đây là hướng dẫn cách cài đặt:

  Để sử dụng bộ điều hướng này, hãy đảm bảo rằng bạn có **@react-navigation/native** và các phụ thuộc của nó, sau đó cài đặt **@react-navigation/bottom-tabs**:

```
npm install @react-navigation/bottom-tabs
```

Bottom tabs navigator

- Để sử dụng trình điều hướng tab này, hãy import nó từ **@react-navigation/bottom-tabs**:

```
const Tab = createBottomTabNavigator();

const BottomTabs = () => {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Article" component={ArticleScreen} />
      <Tab.Screen name="Contacts" component={FeedScreen} />
      <Tab.Screen name="Settings" component={FeedScreen} />
    </Tab.Navigator>
  );
};
```

- Reload lại ứng dụng chúng ta được kết quả như sau:

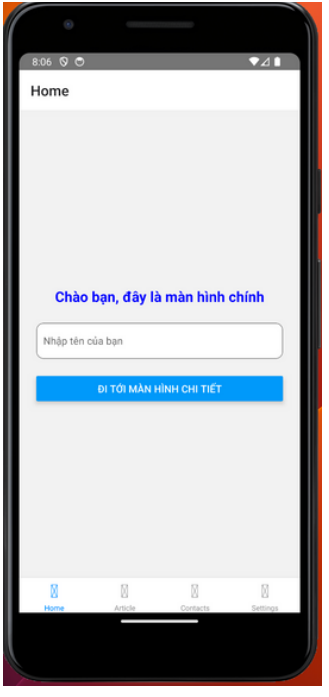
Bottom tabs navigator

- Phần giao diện của bottom tabs của chúng ta có vẻ chưa được đẹp lắm, nhưng các bạn đừng lo. Chúng ta cũng sẽ được hướng dẫn tuỳ chỉnh giao diện cho các bottom tabs này.
- Ở bài trước, các bạn đã được hướng dẫn cách lồng **Drawer Navigator** vào **Stack Navigator**, thì hôm nay chúng ta cũng sẽ được học cách lồng **Bottom Tabs Navigator** vào **Stack Navigator**. Đây cũng là cách phổ biến mà các ứng dụng đang làm.

```
const RootStack = () => {
  return (
    <Stack.Navigator screenOptions={{headerShown: false}}>
      <Stack.Screen name="BottomTabs" component={BottomTabs} />
      <Stack.Screen name="Details" component={DetailsScreen} />
    </Stack.Navigator>
  );
};
```

Bottom tabs navigator

- Reload lại ứng dụng chúng ta có kết quả như sau:



- Từ màn hình **Home** này các bạn có thể navigate tới màn hình **Details** và truyền params như bình thường.
- Khi bạn vào màn hình chi tiết, thì bạn sẽ thấy không còn xuất hiện **Bottom Tabs** nữa. Lý do, bởi vì màn hình chi tiết thuộc **Stack**, chứ không thuộc **Bottom Tabs**.

Prop bottom tabs

Props

Thành phần **Tab.Navigator** chấp nhận các prop sau:

- **id:** ID duy nhất tùy chọn cho người điều hướng. Điều này có thể được sử dụng với **navigation.getParent** để tham chiếu đến điều hướng này trong child navigator.
- **initialRouteName:** Tên của route để hiển thị khi tải đầu tiên của route.
- **screenOptions:** Tùy chọn mặc định để sử dụng cho các màn hình trong bộ điều hướng.

Prop bottom tabs

- **backBehavior:** Điều này kiểm soát những gì xảy ra khi **goBack** được gọi trong trình điều hướng. Điều này bao gồm nhấn nút quay lại hoặc cử chỉ quay lại của thiết bị trên Android.

  Nó hỗ trợ các giá trị sau:

  - **firstRoute** - quay lại màn hình đầu tiên được xác định trong bộ điều hướng (mặc định).
  - **initialRoute** - trở về màn hình ban đầu được truyền trong prop **initialRouteName**, nếu không được thông qua, mặc định là màn hình đầu tiên.
  - **order**- Quay lại màn hình được xác định trước màn hình đang chọn.

Prop bottom tabs

- **history** - quay lại màn hình truy cập lần cuối trong bộ điều hướng; Nếu cùng một màn hình được truy cập nhiều lần, các mục cũ hơn sẽ bị xóa khỏi lịch sử.
- **none** - không xử lý nút quay lại.

- **detachInactiveScreens:** Boolean được sử dụng để chỉ ra liệu màn hình không hoạt động có nên được tách ra khỏi hệ thống phân cấp chế độ xem để tiết kiệm bộ nhớ hay không. Điều này cho phép tích hợp với **react-native-screens**. Mặc định là true.
- **sceneContainerStyle:** Đối tượng style cho thành phần nội dung màn hình.
- **tabBar:** Hàm trả về một phần tử React để hiển thị dưới dạng thanh tab.

Style bottom tabs

- Từ prop **tabBar** chúng ta sẽ đi vào ví dụ sử dụng prop này, bây giờ chúng ta sẽ bắt đầu tìm hiểu cách custom giao diện cho **bottom tabs**.

```
const BottomTabs = () => {
  return (
    <Tab.Navigator tabBar={props => <MyTabBar {...props} />}>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Article" component={ArticleScreen} />
      <Tab.Screen name="Contacts" component={FeedScreen} />
      <Tab.Screen name="Settings" component={FeedScreen} />
    </Tab.Navigator>
  );
};
```

Style bottom tabs

- Nếu sử dụng **tabBar** bạn cũng phải tự xử lý thêm sự kiện nhấn trên bottom tabs để chuyển trang.

```
const MyTabBar = ({state, descriptors, navigation}: BottomTabBarProps) => {
  return (
    <View style={styles.container}>
      {state.routes.map((route, index) => {
        const {options} = descriptors[route.key];

        const isFocused = state.index === index;

        const onPress = () => {
          const event = navigation.emit({
            type: 'tabPress',
            target: route.key,
            canPreventDefault: true,
          });

          if (!isFocused && !event.defaultPrevented) {
            navigation.navigate(route.name, route.params);
          }
        };
```

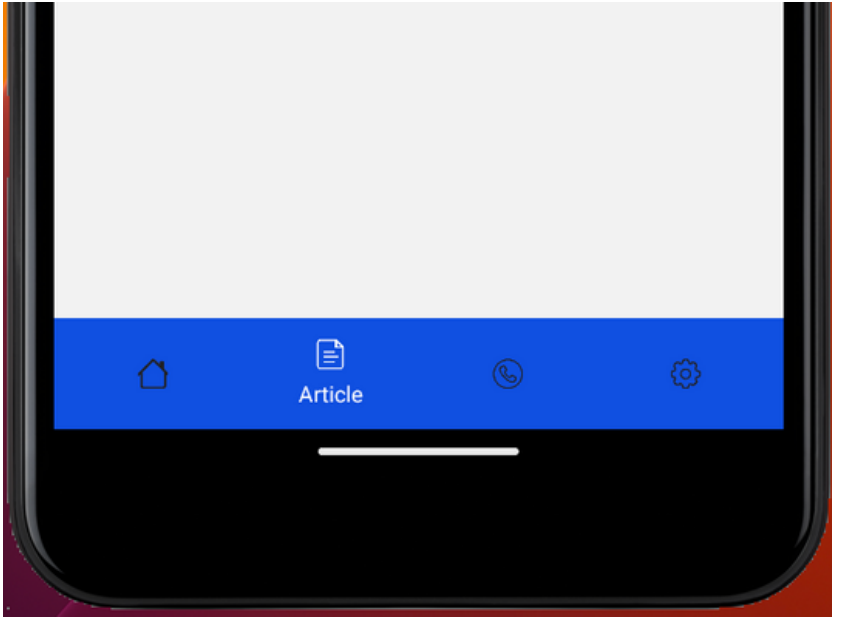
Style bottom tabs

- Bạn có thể custom giao diện từng bottom tab như sau

```
<TouchableOpacity
  key={index}…
  style={styles.containerMenu}>
  <Image
    source={{uri: ICONS_MENU[route.name]}}…
    width={16}
  />
  {!!isFocused && (
    <Text style={styles.textStyle(isFocused)}>
      {label.toString()}
    </Text>
  )}
</TouchableOpacity>
```

Style bottom tabs

- Reload lại ứng dụng, chúng ta có kết quả sau:



Options bottom tabs

Options

Các tùy chọn sau đây có thể được sử dụng để cấu hình màn hình trong navigator. Chúng có thể được chỉ định trong **screenOptions** prop của **Tab.navigator** hoặc tùy chọn prop của **Tab.Screen**.

- **title:** Tiêu đề chung có thể được sử dụng làm dự phòng cho **headerTitle** và **tabBarLabel**.
- **tabBarLabel:** Chuỗi tiêu đề của một tab được hiển thị trong thanh tab hoặc một hàm đã cho **{ focused: boolean, color: string }** trả về một React.Node, để hiển thị trong thanh tab. Khi undefined, **title** màn hình được sử dụng. Để ẩn, sử dụng **tabBarShowLabel**.

Options bottom tabs

- **tabBarShowLabel:** tab có hiển thị hay không. Mặc định là true.
- **tabBarLabelPosition:** Cho dù label được hiển thị bên dưới biểu tượng hay bên cạnh icon.
  - **below-icon:** nhãn được hiển thị bên dưới biểu tượng (điển hình cho iPhone)
  - **beside-icon:** nhãn được hiển thị bên cạnh biểu tượng (điển hình cho iPad)

  Theo mặc định, vị trí được chọn tự động dựa trên chiều rộng thiết bị.
- **tabBarLabelStyle:** Object style cho nhãn tab.

Options bottom tabs

- **tabBarIcon:** Hàm cung cấp**{ focused: boolean, color: string, size: number }** trả về một React.Node, để hiển thị trong thanh tab..
- **tabBarIconStyle:** Object style cho biểu tượng tab.
- **tabBarBadge:** Văn bản để hiển thị trong khung badge trên biểu tượng tab. Chấp nhận một chuỗi hoặc một số.
- **tabBarBadgeStyle:** Kiểu cho khung badge trên icon tab. Bạn có thể chỉ định màu nền hoặc màu văn bản tại đây..
- **tabBarAccessibilityLabel:** Trợ năng label cho nút tab. Trình đọc màn hình đọc được đọc khi người dùng nhấn vào tab. Bạn nên đặt tùy chọn này nếu bạn không có label cho tab.

Options bottom tabs

- **tabBarActiveTintColor:** Màu sắc cho icon và label trong tab đang được chọn.
- **tabBarInactiveTintColor:** Màu sắc cho icon và label trong các tab không được chọn.
- **tabBarActiveBackgroundColor:** Màu nền cho tab đang được chọn.
- **tabBarInactiveBackgroundColor:** Màu nền cho các tab không được chọn.

Options bottom tabs

- **tabBarStyle:**
  Đối tượng style cho thanh tab. Bạn có thể định cấu hình các style như **backgroundColor** tại đây.
  Để hiển thị màn hình của bạn dưới thanh tab, bạn có thể đặt kiểu **position** thành absolute:

```
<Tab.Navigator
  screenOptions={{
    tabBarStyle: { position: 'absolute' },
  }}
>
```

Bạn cũng có thể cần thêm **marginBottom** vào nội dung của mình nếu bạn có thanh tab được style absolute. React Navigation sẽ không tự động làm điều đó.

Options bottom tabs

- **tabBarBackground:** Hàm trả về một React Element để sử dụng làm background cho thanh tab. Bạn có thể hiển thị hình ảnh, gradient, opacity, v.v.:

```
 <Tab.Navigator
   screenOptions={{
   tabBarBackground: () => (
   <BlurView tint="light" intensity={100} style=
{StyleSheet.absoluteFill} />
    ),
  }}
>
```

Events bottom tabs

Events

Bộ điều hướng có thể emit events trên một số hành động nhất định. Các sự kiện được hỗ trợ là:

- **tabPress:** Sự kiện này được kích hoạt khi người dùng nhấn nút tab cho màn hình hiện tại trong thanh tab. Theo mặc định, nhấn tab sẽ thực hiện một số việc:

  - Nếu tab không được chọn, nhấn tab sẽ tập trung tab đó.
  - Nếu tab đã được chọn:

  - Nếu màn hình cho tab hiển thị dạng xem cuộn, bạn có thể sử dụng **useScrollToTop** để cuộn nó lên trên cùng.

  - Nếu màn hình cho tab hiển thị trình điều hướng ngăn xếp, hành động **popToTop** sẽ được thực hiện trên stack.

Events bottom tabs

Để ngăn chặn sự kiện mặc định, bạn có thể gọi **event.preventDefault**:

```
React.useEffect(() => {
  const unsubscribe = navigation.addListener('tabPress', (e)
=> {
    // Ngăn chọn sự kiện nhấn mặc định
    e.preventDefault();

    // Làm gì đó ở đây
    // ...
  });

  return unsubscribe;
}, [navigation]);
```

Events bottom tabs

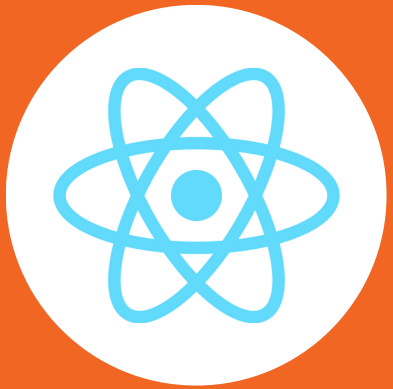
- **tabLongPress:** Sự kiện này được kích hoạt khi người dùng nhấn nút tab cho màn hình hiện tại trong thanh tab trong một khoảng thời gian dài. Nếu bạn có thanh tab tùy chỉnh, hãy đảm bảo emit ra sự kiện này.

```
React.useEffect(() => {
  const unsubscribe = navigation.addListener('tabLongPress',
(e) => {
    // Làm gì đó ở đây
  });

  return unsubscribe;
}, [navigation]);
```

FPT Education

FPT POLYTECHNIC



LẬP TRÌNH ĐA NỀN TẢNG VỚI REACT NATIVE

BÀI 7: BOTTOM VÀ TOP TABS NAVIGATOR TRONG REACT NATIVE

PHẦN 2: GIỚI THIỆU VỀ TAB VIEW

https://caodang.fpt.edu.vn/

Mục tiêu

- [ ] Xây dựng được top tab cho ứng dụng
- [ ] Tuỳ chỉnh giao diện cho top tab

React Native Tab View

React Native Tab View là một thành phần Tab View đa nền tảng cho React Native, được triển khai bằng **react-native-pager-view** trên **Android** &; **iOS** và **PanResponder** trên Web, macOS và Windows.

Nó tuân theo các nguyên tắc thiết kế material theo mặc định, nhưng bạn cũng có thể sử dụng thanh tab tùy chỉnh của riêng mình hoặc đặt thanh tab ở dưới cùng.

Top tabs

- Một Top Tab có giao diện như sau:

Top tabs

Cài đặt thư viện

Để sử dụng gói này, hãy mở một Terminal trong thư mục gốc của dự án và chạy:

Hướng dẫn

- Bước 1: Cài đặt **@react-navigation/material-top-tabs react-native-tab-view**

```
npm install react-native-tab-view
```

- Bước 2: Cài đặt **react-native-pager-view**

```
npm install react-native-pager-view
```

Top tabs

API liên quan

Package xuất thành phần **TabView** là thành phần bạn sử dụng để hiển thị chế độ xem tab và thành phần **TabBar** là triển khai thanh tab mặc định.

TabView

Thành phần container chịu trách nhiệm hiển thị và quản lý các tab. Thực hiện theo phong cách thiết kế material theo mặc định.

Nó sẽ được code như thế này:

Top tabs

```
<TabView
  navigationState={{ index, routes }}
  onIndexChange={setIndex}
  renderScene={SceneMap({
    first: FirstRoute,
    second: SecondRoute,
  })}
 />
```

- **TabView Props**
  - **navigationState (yêu cầu)**

Top tabs

State cho dạng xem tab. Trạng thái nên chứa các thuộc tính sau:

- **index**: một số đại diện cho index của route đang hoạt động trong routes array.
- **routes**: an array containing a list of route objects used for rendering the tabs

Mỗi đối tượng route phải chứa các thuộc tính sau:

- **key:** một khóa duy nhất để xác định tuyến đường (bắt buộc)
- **title:** tiêu đề cho tuyến đường hiển thị trong thanh tab
- **icon:** icon cho route hiển thị trong thanh tab
- **accessibility:** label cho nút tab
- **testID:** id kiểm tra cho nút tab

Top tabs

- Ví dụ

```
{
  index: 1,
  routes: [
    { key: 'music', title: 'Music' },
    { key: 'albums', title: 'Albums' },
    { key: 'recents', title: 'Recents' },
    { key: 'purchased', title: 'Purchased' },
  ]
}
```

TabView là một thành phần được kiểm soát, có nghĩa là index cần được cập nhật thông qua callback **onIndexChange**.

Top tabs

- **onIndexChange (yêu cầu):**

  Callback được gọi khi tab thay đổi, nhận chỉ mục của tab mới làm đối số. Trạng thái navigation cần được cập nhật khi nó được gọi, nếu không thay đổi sẽ bị loại bỏ.

- **renderScene (yêu cầu):**

  Callback trả về một phần tử react để render dưới dạng page cho tab. Nhận một đối tượng chứa route làm đối số:

Top tabs

```
const renderScene = ({ route, jumpTo }) => {
  switch (route.key) {
    case 'music':
      return <MusicRoute jumpTo={jumpTo} />;
    case 'albums':
      return <AlbumsRoute jumpTo={jumpTo} />;
  }
};
```

Bạn cần đảm bảo rằng các route riêng lẻ của bạn triển khai **shouldComponentUpdate** để cải thiện hiệu suất. Để dễ dàng chỉ định các thành phần hơn, bạn có thể sử dụng trình trợ giúp **SceneMap**.

Top tabs

SceneMap lấy một đối tượng với ánh xạ của **route.key** đến các thành phần React và trả về một hàm để sử dụng với **renderScene** prop.

```
import { SceneMap } from 'react-native-tab-view';
...
const renderScene = SceneMap({
  music: MusicRoute,
  albums: AlbumsRoute,
});
```

Chỉ định các thành phần theo cách này dễ dàng hơn và đảm nhận việc triển khai phương thức **shouldComponentUpdate**.

Top tabs

Mỗi cảnh nhận được các đạo cụ sau:

- **route**: tuyến đường hiện tại được thành phần hiển thị.
- **route:** tuyến đường hiện tại được thành phần hiển thị.
- **position**: Nút hoạt hình đại diện cho vị trí hiện tại.

Phương thức **jumpTo** có thể được sử dụng để điều hướng đến các tab khác theo chương trình:

```
props.jumpTo('albums');
```

Top tabs

Tất cả scenes được hiển thị bằng **SceneMap** được tối ưu hóa bằng **React.PureComponent** và không hiển thị lại khi đạo cụ hoặc trạng thái của parent thay đổi. Nếu bạn cần kiểm soát nhiều hơn cách cập nhật scenes của mình (ví dụ: kích hoạt re-render ngay cả khi trạng thái điều hướng không thay đổi), hãy sử dụng **renderScene** trực tiếp thay vì sử dụng **SceneMap**.

Top tabs

Nếu bạn cần chuyển thêm prop, hãy sử dụng hàm **renderScene** tùy chỉnh:

```
const renderScene = ({ route }) => {
  switch (route.key) {
    case 'first':
      return <FirstRoute foo={this.props.foo} />;
    case 'second':
      return <SecondRoute />;
    default:
      return null;
  }
};
```

Top tabs

- **renderTabBar:**

  Callback trả về một React Element tùy chỉnh để sử dụng làm thanh tab:

```
import { TabBar } from 'react-native-tab-view';
...
<TabView
  renderTabBar={props => <TabBar {...props} />}
  ...
/>
```

Top tabs

- **swipeEnabled:**

  Boolean cho biết có bật cử chỉ vuốt hay không. Cử chỉ vuốt được bật theo mặc định. Chuyển false sẽ vô hiệu hóa cử chỉ vuốt, nhưng người dùng vẫn có thể chuyển đổi tab bằng cách nhấn thanh tab.

- **overScrollMode:**

  Được sử dụng để ghi đè giá trị mặc định của chế độ **overScroll** của máy nhắn tin. Có thể là tự động, luôn luôn hoặc không bao giờ (chỉ dành cho Android).

Top tabs

TabBar

Thiết kế material thanh tab theo chủ đề. Để tùy chỉnh thanh tab, bạn cần sử dụng prop **renderTabBar** của TabView để hiển thị **TabBar** và chuyển các prop bổ sung.

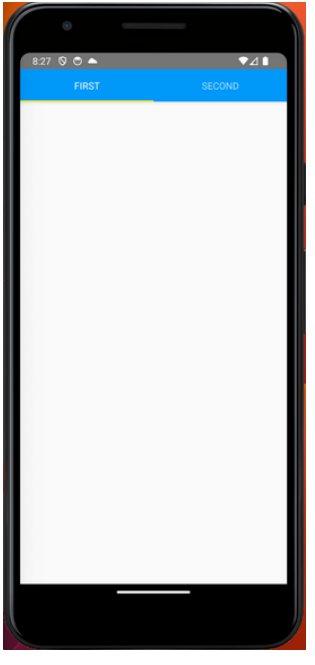
Ví dụ: để tùy chỉnh màu chỉ báo và màu nền của thanh tab, bạn có thể thêm **indicatorStyle** và style props vào **TabBar** tương ứng:

Top tabs

```
const renderTabBar = props => (
  <TabBar
    {...props}
    indicatorStyle={{ backgroundColor: 'white' }}
    style={{ backgroundColor: 'pink' }}
  />
);
//...
return (
  <TabView
    renderTabBar={renderTabBar}
    ...
  />
);
```

Top tabs

- Bây giờ chúng ta sẽ bắt đầu xây dựng một Top tabs hoàn chỉnh, có giao diện như bên dưới:



Top tabs

- Chúng ta sẽ cần 2 screen cho mỗi tab, **FirstRoute** và **SecondRoute.** Sau đó thêm 2 screen vào **SceneMap.**

```
const FirstRoute = () => <View style={{flex: 1}} />;

const SecondRoute = () => <View style={{flex: 1}} />;

const renderScene = SceneMap({
  first: FirstRoute,
  second: SecondRoute,
});
```

Top tabs

- Gọi hook **useWindowDemensions** để lấy kích thước màn hình.
- State **index** để định vị vị trí hiện tại của tab.
- State **routes** để định vị screen của **key** từ **SceneMap.** Đặt title cho tab

```
const layout = useWindowDimensions();

const [index, setIndex] = React.useState(0)
const [routes] = React.useState([
  {key: 'first', title: 'First'},
  {key: 'second', title: 'Second'},
]);
```

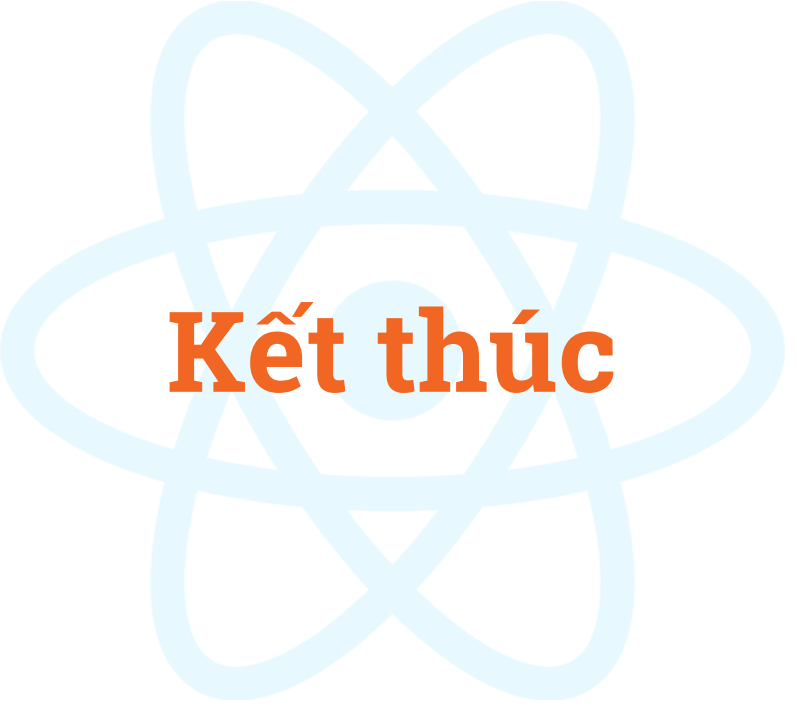
Top tabs

- Gọi **TabView** sau đó thêm những prop chúng ta đã chuẩn bị sẵn ở phần trên:

```
<TabView
  navigationState={{index, routes}}
  renderScene={renderScene}
  onIndexChange={setIndex}
  initialLayout={{width: layout.width}}
/>
```

Tổng kết

- [ ] Biết cách xây dựng bottom tabs navigator
- [ ] Chỉnh sửa giao diện cho bottom tabs
- [ ] Xây dựng được top tab cho ứng dụng
- [ ] Tuỳ chỉnh giao diện cho top tab



Kết thúc