



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

React Native cơ bản

Bài 3: Chức năng Xem Thông tin

<http://csc.edu.vn/laptrinh/>

Chức năng xem thông tin



- ☐ Component là gì?
- ☐ Sử dụng Style
- ☐ Minh họa

Component là gì?



- ☐ Giới thiệu
- ☐ Vòng đời của Component
- ☐ Props và State
- ☐ Event

- Là thành phần quan trọng nhất và được sử dụng để xây dựng các chức năng màn hình trong ứng dụng. Mỗi chức năng trong màn hình đều được xem là 1 Component.
- Các Components có thể chứa các Components con.
- Các Components cơ sở như:
 - View
 - Image
 - TouchableOpacity
 - Text
 - TextInput
 - ScrollView
 - FlashList
 - StyleSheet

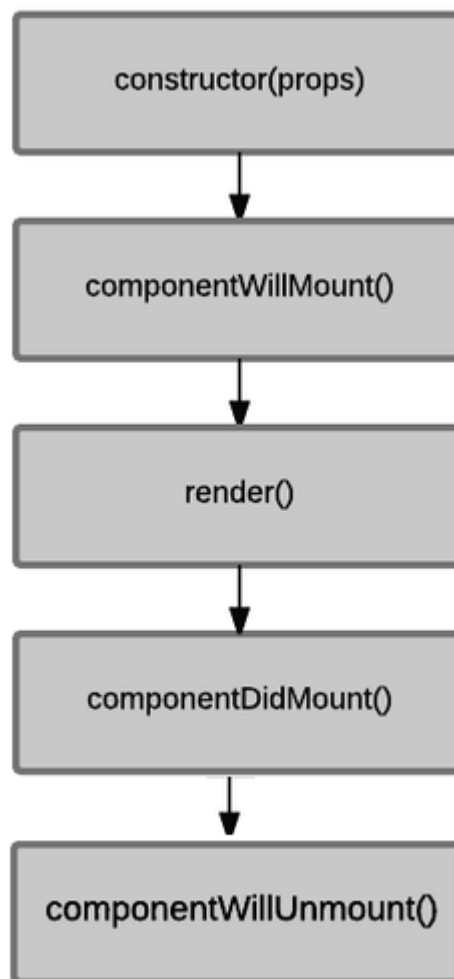
- Ví dụ:

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

export default class App extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Header/>
        <Text style={styles.welcome}>Chào bạn đến với React Native!</Text>
        <View>
          <Text style={{ fontSize: 24 }}>
            Hello
          </Text>
        </View>
      </View>
    );
  }
}
```

- Ngoài ra, còn có các Component được phát triển từ các Developer. Để sử dụng các Component này, chúng ta phải cài đặt vào ứng dụng trước khi dùng.

Vòng đời của Component



Vòng đời của Component



- `constructor (props)`: là phương thức khởi tạo. Phương thức này được thực thi đầu tiên khi một Component được tạo ra. Chúng ta thường khởi tạo các giá trị ban đầu cho các biến, state hay props.
- `componentWillMount()`: phương thức này thực thi sau constructor. Chúng ta ko nên cập nhật giá trị state trong phương thức này.
- `render()`: là phương thức quan trọng phải có trong 1 Component. Phương thức này trả về các React Element (được tạo bằng JSX) dựa trên các state và props của Component.
- `componentDidMount()`: phương thức này được gọi sau khi React Element đã được phát sinh. Thường chúng ta đọc dữ liệu từ các API ở phương thức này.
- `componentWillUnmount()`: phương thức này được gọi trước khi component kết thúc

Vòng đời của Component



- Ngoài ra chúng ta có thể định nghĩa các phương thức xử lý nghiệp vụ trong Component khi cần.

```
export default class App extends Component{
  constructor(props){
    super(props);
    console.log("Phương thức khởi tạo");
  }

  XL_Nhan(){
    console.log("Phương thức tự định nghĩa");
  }
  componentWillMount(){
    console.log("Phương thức componentWillMount");
  }
  render(){
    return(
      <View style={styles.middle}>
        <View style={[styles.box, styles.big_green_box]}>
          <View style={[styles.inner_box, styles.orange_box]}></View>
        </View>
      </View>
    );
  }
  componentDidMount(){
    console.log("Phương thức componentDidMount");
  }
  componentWillUnmount(){
    console.log("Phương thức componentWillUnmount");
  }
}
```


❑ Props

- Là thuộc tính của Components, cho phép chúng ta lưu trữ và truyền dữ liệu khi khởi tạo Components

❑ Props

- Ví dụ: Tạo component: Heading.js có nội dung sau

```
export default class Heading extends Component {
  render() {
    return (
      <View>
        <Text style={style.header}>{this.props.message}</Text>
        <Text style={style.header}>{this.props.message1}</Text>
      </View>
    )
  }
}

Heading.propTypes = {
  message: PropTypes.string
}

Heading.defaultProps = {
  message: 'Heading One'
}

const style = StyleSheet.create({
  header: {
    fontSize: 20,
    textAlign: 'center',
    margin: 2,
    color: 'red',
    fontWeight: "bold"
  }
})
```

❑ Props

- Ví dụ: Tạo component: Footer.js có nội dung sau

```
import React, { Component } from "react";
import { View, Text, StyleSheet } from "react-native";
import PropTypes from 'prop-types'
// Child component
export default class Footer extends Component {
  render() {
    return (
      <View>
        <Text style={styles.tieu_de_footer}>{this.props.tieu_de}</Text>
      </View>
    )
  }
}
Footer.propTypes = {
  tieu_de: PropTypes.string
}
Footer.defaultProps = {
  tieu_de: 'Footer One'
}
const styles = StyleSheet.create({
  tieu_de_footer: {
    alignItems: 'center',
    fontSize: 18
  }
});
```

❑ Props

- Ví dụ: Tạo component: MH_Props.js có nội dung sau

```
import React, { Component } from "react";
import { View, Text, StyleSheet } from "react-native";
import Heading from "../Heading";
import Footer from "../Footer";
export default class MH_Prop extends Component {
  render() {
    return (
      <View style={styles.container}>
        <Heading message={'Trung Tâm Tin Học - ĐHKHTN'} message1={'Ví dụ Props'} />
        <View>
          <Text>React Native - Chức năng Xem Thông tin </Text>
        </View>
        <Footer tieu_de={'Cơ sở chính: 227 Nguyễn Văn Cừ Quận 5 '} />
      </View>
    )
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    //justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#F5FCFF',
  }
});
```

❑ State

- Là trạng thái của Components, cho phép chúng ta lưu trữ thông tin. Khi nội dung của state thay đổi thì phương thức render sẽ được tự động gọi thực hiện

❑ State

- Ví dụ: Tạo MH_State.js

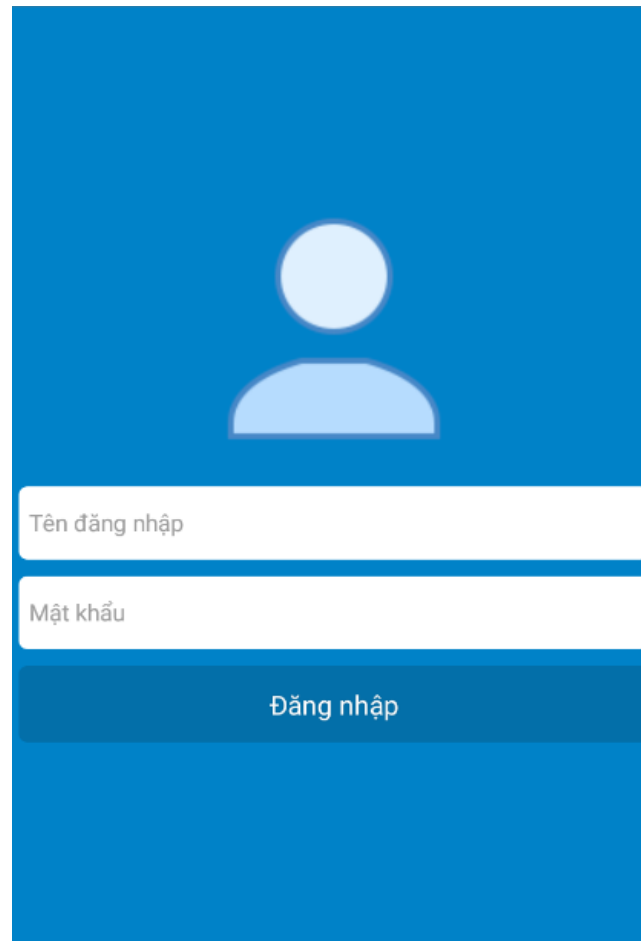
```
import React, { Component } from 'react';
import { View, Text, StyleSheet, TextInput } from "react-native";
import Heading from "../Heading";
import Footer from "../Footer";
export default class MH_State extends Component {
  constructor(props) {
    super(props)
    this.state = {input: ''}
  }
  handleChangeInput = (text) => {
    this.setState({ input: text })
  }
  render() {
    return (
      <View style={style.container}>
        <Heading message={'Trung Tâm Tin Học - ĐHKHTN'} message1={'Ví dụ State'} />
        <View>
          <TextInput style={style.input}
            onChangeText={this.handleChangeInput}
            value={this.state.input} placeholder='Nhập Họ tên của bạn'
          />
          <Text style={style.text}>{this.state.input}</Text>
        </View>
        <Footer tieu_de={'Cơ sở chính: 227 Nguyễn Văn Cừ Quận 5 '} />
      </View>
    );
  }
}
```

❑ Event (sự kiện):

- Khi ứng dụng thực thi, ngoài các sự kiện do ứng dụng phát sinh thì khi người dùng tương tác với ứng dụng sẽ phát sinh sự kiện tương ứng, ví dụ như khi người chọn vào một chức năng “đăng nhập” trong màn hình đăng nhập.
- Để xử lý, chúng ta phải khai báo và cài đặt sự kiện tương ứng với thao tác của người.

❑ Event (sự kiện):

- Ví dụ: xử lý sự kiện khi người dùng chọn vào nút đăng nhập



A login form with a blue background. At the top center is a light blue user icon. Below it are two white input fields with blue borders. The first field is labeled 'Tên đăng nhập' and the second is labeled 'Mật khẩu'. Below these fields is a dark blue button with the text 'Đăng nhập' in white.

❑ Event (sự kiện):

- Ví dụ: Xây dựng MH_Dang_nhap.js

```
import React, { Component } from 'react';
import { View, Image, TouchableOpacity, Text, TextInput, StyleSheet, Alert } from 'react-native';
import Du_lieu from '../data/Du_lieu';
export default class MH_Dang_nhap extends Component {
  constructor(props) {
    super(props);
    this.state = { Ten_Dang_nhap: '', Mat_khau: '', Thong_bao: '' }
  }
  render() {
    return (
      <View style={style.container}>
        <View style={{ flex: 1, alignItems: "center", justifyContent: "flex-end" }}>
          <Image style={style.image} source={require('../images/user_login.png')}></Image>
        </View>
        <View style={{ flex: 1, justifyContent: "flex-start" }}>
          <TextInput onChangeText={({ Ten_Dang_nhap }) => this.setState({ Ten_Dang_nhap })}
            value={this.state.Ten_Dang_nhap} placeholder='Tên đăng nhập' style={style.input} />
          <TextInput onChangeText={({ Mat_khau }) => this.setState({ Mat_khau })}
            value={this.state.Mat_khau} placeholder='Mật khẩu' style={style.input} secureTextEntry />
          <TouchableOpacity onPress={this.XL_Nhan.bind(this)} activeOpacity={0.5}>
            <View style={style.button}>
              <Text style={style.text}>Đăng nhập</Text>
            </View>
          </TouchableOpacity>
          <Text style={{ textAlign: "center" }}>{this.state.Thong_bao}</Text>
        </View>
      </View>
    );
  }
}
```

❑ Event (sự kiện):

- Cài đặt sự kiện:

```
<TouchableOpacity onPress={this.XL_Nhan.bind(this)} activeOpacity={0.5}>
  <View style={style.button}>
    <Text style={style.text}>Đăng nhập</Text>
  </View>
</TouchableOpacity>
```

- Xử lý sự kiện:

```
XL_Nhan() {
  if (this.state.Ten_Dang_nhap == Du_lieu.Nguoi_dung.Ten_Dang_nhap
    && this.state.Mat_khau == Du_lieu.Nguoi_dung.Mat_khau)
    //Alert.alert("Đăng nhập thành công");
    this.setState({"Thong_bao": "Đăng nhập thành công"})
  else
    //Alert.alert("Đăng nhập thất bại");
    this.setState({"Thong_bao": "Đăng nhập thất bại"})
}
```

❑ Event (sự kiện):

- StyleSheet

```
const style = StyleSheet.create({
  container: {flex: 1, backgroundColor: '#0082c8', justifyContent: 'center'},
  input: {height: 44, paddingHorizontal: 6, backgroundColor: '#ffffff',
    margin: 5,
    borderRadius: 5
  },
  button: {
    height: 46,
    borderRadius: 5,
    margin: 5,
    backgroundColor: '#036fa9',
    justifyContent: 'center',
    alignItems: 'center'
  },
  text: {color: '#ffffff', fontSize: 16,},
  image: {
    width: 140, height: 140,
    marginBottom: 20,
    borderTopLeftRadius: 10,
    borderTopRightRadius: 10,
    borderBottomLeftRadius: 10,
    borderBottomRightRadius: 10
  }
})
```

- ☐ Giới thiệu
- ☐ Sử dụng FlexBox
- ☐ Sử dụng Position

❑ Style

- Là thành phần dùng để định dạng giao diện của ứng dụng
- Sử dụng ngôn ngữ JavaScript để định dạng kích thước, font chữ , màu sắc cho từng thành phần.
- Để định nghĩa style, chúng ta sử dụng đối tượng StyleSheet, sau đó cài đặt cho các đối tượng được phát sinh thông qua thuộc tính style
- Chúng ta cũng có thể định nghĩa và cài đặt trực tiếp (inline) cho các đối tượng được phát sinh.

❑ Các thuộc tính thường dùng

- color: màu chữ
- backgroundColor: màu nền
- fontSize: kích thước font
- fontColor: màu font
- fontFamily: tên font
- margin: qui định khoảng cách của đối tượng so với đối tượng cha
- padding: qui định khoảng cách của đối tượng so với nội dung bên trong
- height: qui định chiều cao của đối tượng
- width: qui định chiều rộng của đối tượng
- borderRadius: bo tròn khung đối tượng.

❑ Ví dụ 1

- Định nghĩa style:

```
const style = StyleSheet.create({
  container: {flex: 1, backgroundColor: '#0082c8', justifyContent: 'center'},
  input: {height: 44, paddingHorizontal: 6, backgroundColor: 'ffffff',
    margin: 5,
    borderRadius: 5
  },
});
```

- Cài đặt cho các đối tượng được phát sinh

```
<TextInput onChangeText={({Ten_Dang_nhap) => this.setState({ Ten_Dang_nhap })}}
  value={this.state.Ten_Dang_nhap} placeholder='Tên đăng nhập' style={style.input} />
<TextInput onChangeText={({Mat_khau) => this.setState({ Mat_khau })}}
  value={this.state.Mat_khau} placeholder='Mật khẩu' style={style.input} secureTextEntry />
```

❑ Ví dụ 2

- Cài đặt dạng inline: `style={{flex: 1, alignItems: "center", ..." }}`

```
<View style={{ flex: 1, alignItems: "center", justifyContent: "flex-end" }}>  
  <Image style={style.image} source={require('../images/user_login.png')}></Image>  
</View>  
<View style={{ flex: 1, justifyContent: "flex-start" }}>
```


❑ Flex: <https://facebook.github.io/react-native/docs/flexbox>

- Khi sử dụng thuộc tính flex trong style thì thành phần đó sẽ có kích thước như đối tượng cha chứa nó
- Ví dụ 1:

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

export default class FlexDirectionBasics extends Component {
  render() {
    return (
      <View style={{flex:1,backgroundColor:'#0082cB'}}>
      </View>
    );
  }
};
```

❑ Flex

- Kết quả:



Sử dụng FlexBox



❑ Flex: <https://facebook.github.io/react-native/docs/flexbox>

- Ví dụ 2:

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

export default class FlexDirectionBasics extends Component {
  render() {
    return (
      <View style={{flex:1,backgroundColor:'#0082cB'}}>
        <View style={{flex:3,backgroundColor:'powderblue'}}>
        </View>
        <View style={{flex:7,backgroundColor:'steelblue'}}>
        </View>
      </View>
    );
  }
};
```



❑ Flex

- Kết quả:



❑ FlexBox

- Cho phép chúng ta định nghĩa dạng thể hiện các thành phần con bên trong thành phần cha
- `flexDirection`: sắp xếp các đối tượng con bên trong theo column (chiều dọc) hoặc row (chiều ngang) màn hình.
- `justifyContent`: canh các đối tượng con theo trục chính của flex
- `alignItems`: canh các đối tượng con theo trục còn lại của flex

❑ FlexBox

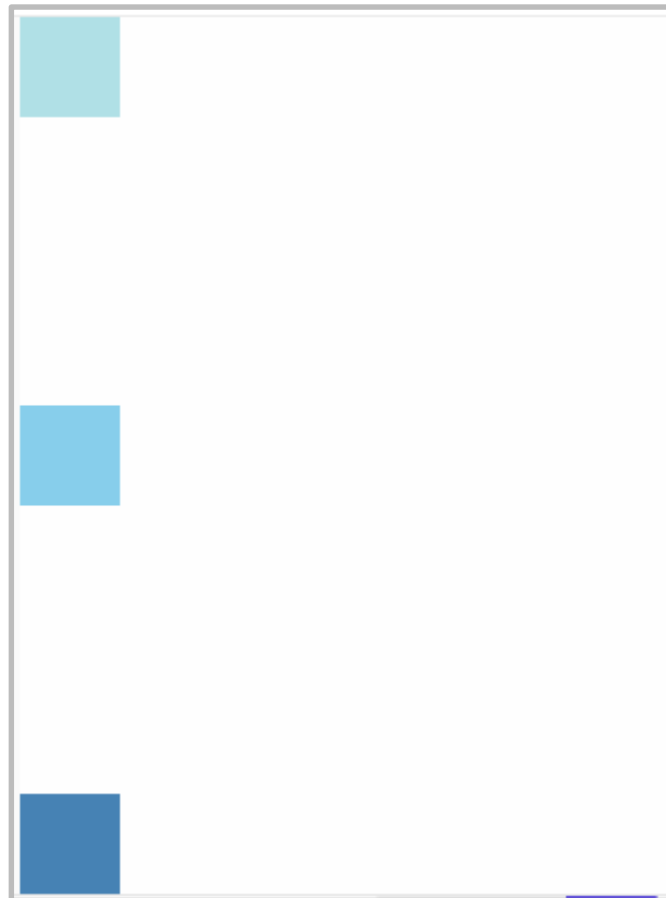
- Ví dụ 1:

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

export default class JustifyContentBasics extends Component {
  render() {
    return (
      // Try setting `justifyContent` to `center`.
      // Try setting `flexDirection` to `row`.
      <View style={{
        flex: 1,
        flexDirection: 'column',
        justifyContent: 'space-between',
      }}>
        <View style={{width: 50, height: 50, backgroundColor:
'powderblue'}} />
        <View style={{width: 50, height: 50, backgroundColor:
'skyblue'}} />
        <View style={{width: 50, height: 50, backgroundColor:
'steelblue'}} />
      </View>
    );
  }
}
```

❑ FlexBox

- Kết quả:



❑ FlexBox

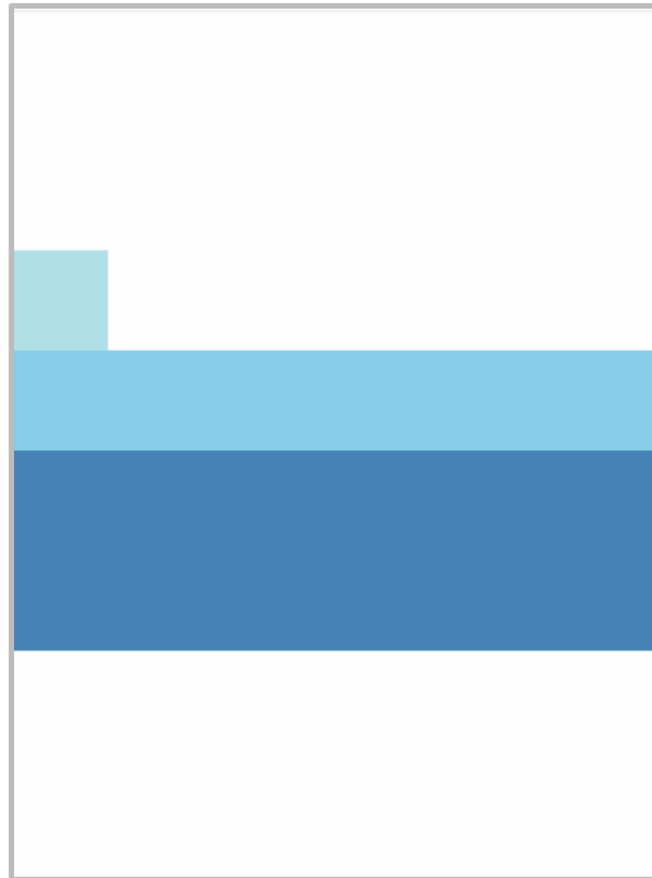
- Ví dụ 2:

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

export default class AlignItemsBasics extends Component {
  render() {
    return (
      // Try setting `alignItems` to 'flex-start'
      // Try setting `justifyContent` to 'flex-end'.
      // Try setting `flexDirection` to 'row'.
      <View style={{
        flex: 1,
        flexDirection: 'column',
        justifyContent: 'center',
        alignItems: 'stretch',
      }}>
        <View style={{width: 50, height: 50, backgroundColor:
'powderblue'}} />
        <View style={{height: 50, backgroundColor: 'skyblue'}}
/>
        <View style={{height: 100, backgroundColor: 'steelblue'}}
/>
      </View>
    );
  }
};
```


❑ FlexBox

- Kết quả:



□ Position :

- Cho phép chúng ta định vị trí của đối tượng:
 - absolute: định vị trí so với đối tượng cha.
 - relative: định vị trí so với chính nó
- Để định vị chúng ta kết hợp sử dụng các thuộc tính: top, left right, bottom
- Ví dụ:

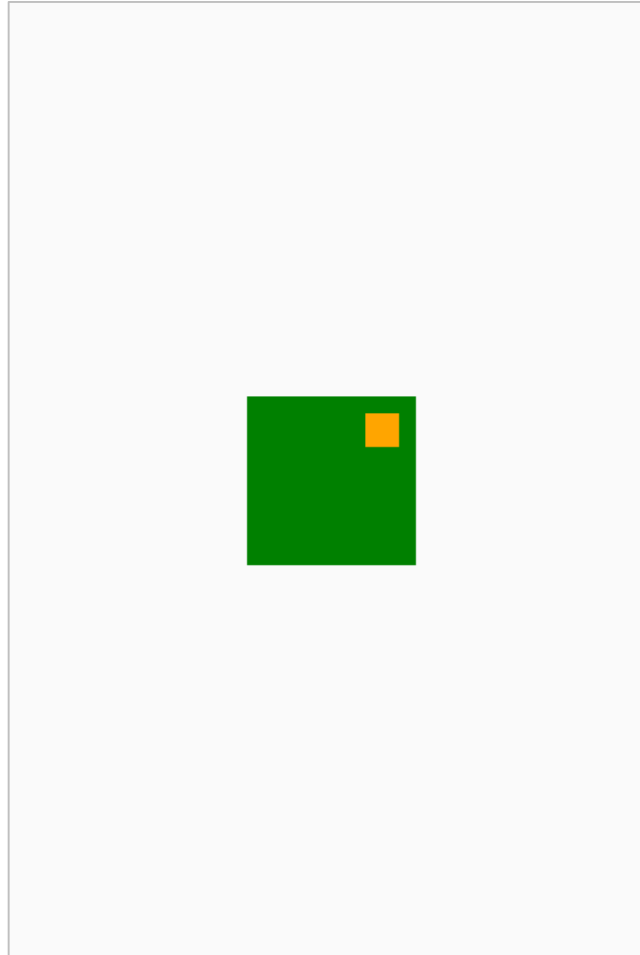
```
render(){
  return(
    <View style={styles.middle}>
      <View style={[styles.box, styles.big_green_box]}>
        <View style={[styles.inner_box, styles.orange_box]}></View>
      </View>
    </View>
  );
}
```

❑ Position:

```
const styles= StyleSheet.create({
  middle: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center'
  },
  box: {
    width: 100,
    height: 100,
    backgroundColor: '#333'
  },
  big_green_box: {
    backgroundColor: 'green'
  },
  inner_box: {
    width: 20,
    height: 20,
  },
  orange_box: {
    position: 'absolute',
    backgroundColor: 'orange',
    top: 10,
    right: 10
  }
})
```

❑ Position:

- Kết quả:



- ❑ Giáo viên minh họa