

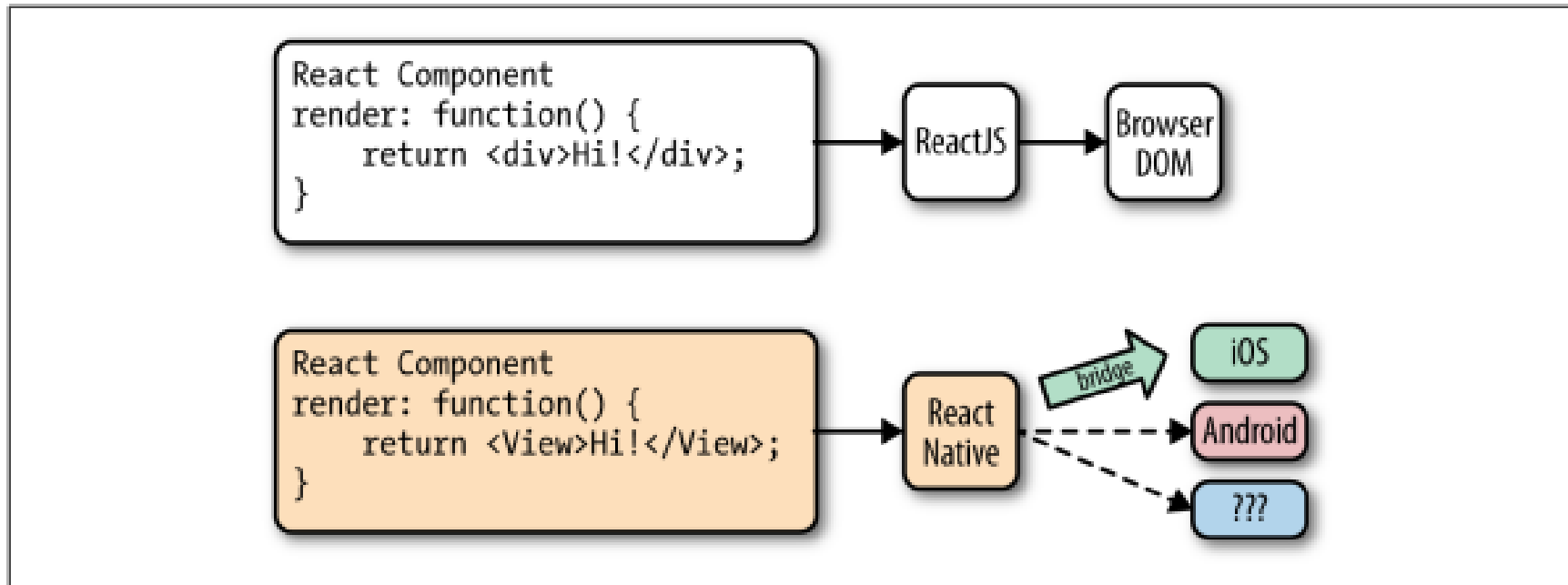
# Chapter 2

---

CORE COMPONENTS AND STYLING

# How Does React Native Work?

- ❖ การทำงานของ React Native จะคล้ายกับ React แต่แทนที่จะเรนเดอร์ DOM ในเว็บเบราว์เซอร์ แต่ React Native จะทำการเรียก Objective-C APIs เพื่อเรนเดอร์คอมโพเนนต์ของ iOS, หรือเรียก Java APIs เพื่อเรนเดอร์คอมโพเนนต์ของ Android



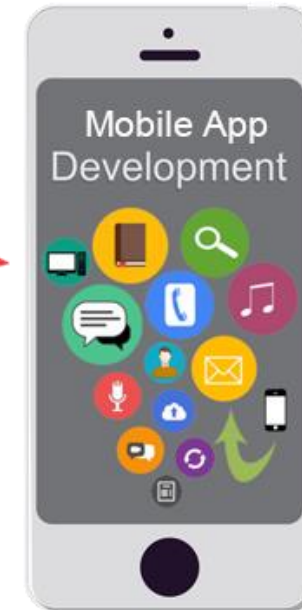
# How Does React Native Work?

React + Real Native Apps

```
Const App = props => {
  Return (
    <View>
      <Text> Hello World!</Text>
    </View>
  );
}
```

Compiled to

Views are  
compiled!



React for the Web

Native Component



Native Component



React Native



`<div>`

`Android.view`

`UIView`

`<View>`

`<input>`

`EditText`

`UITextField`

`<TextInput>`

...

...

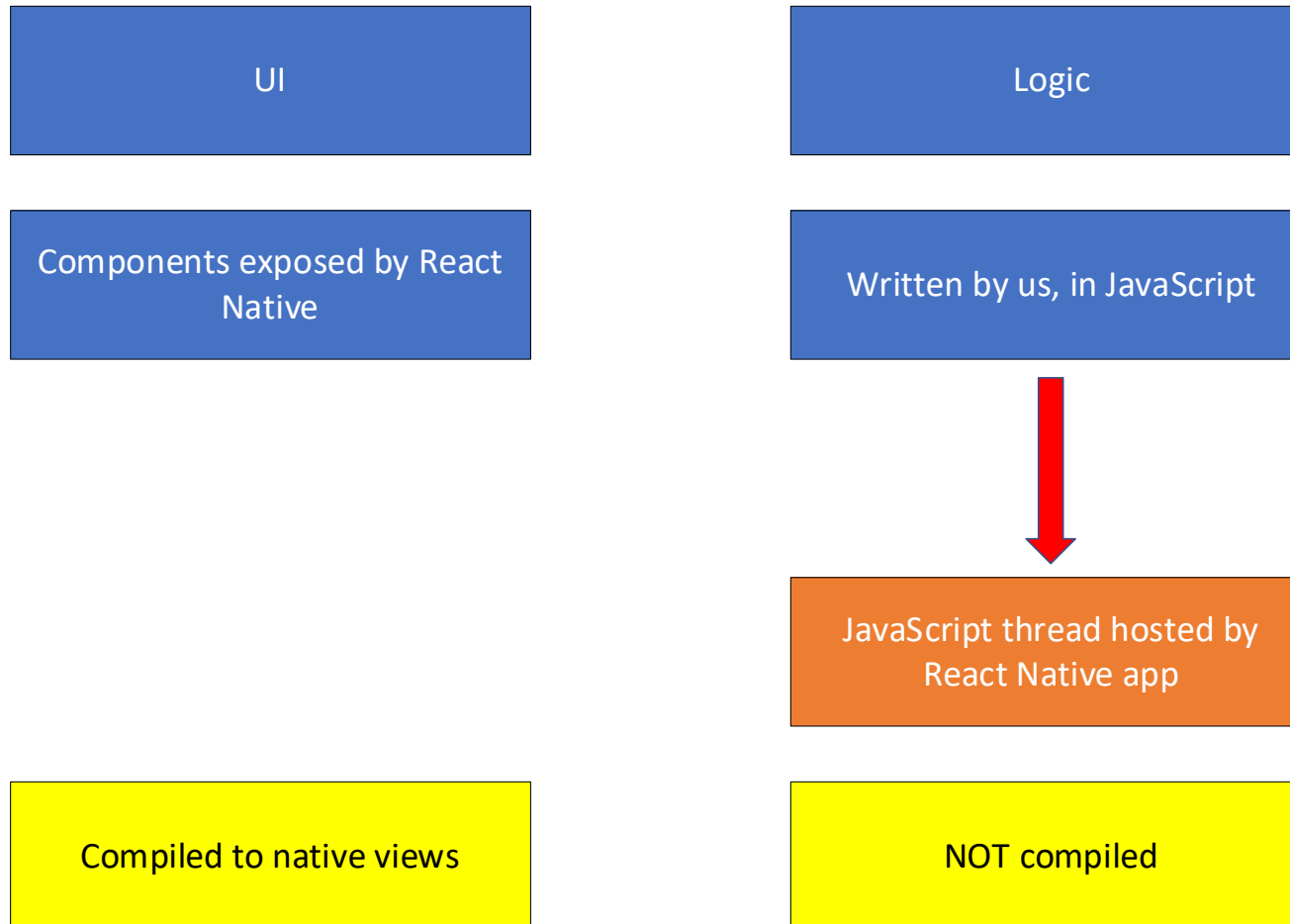
...

...

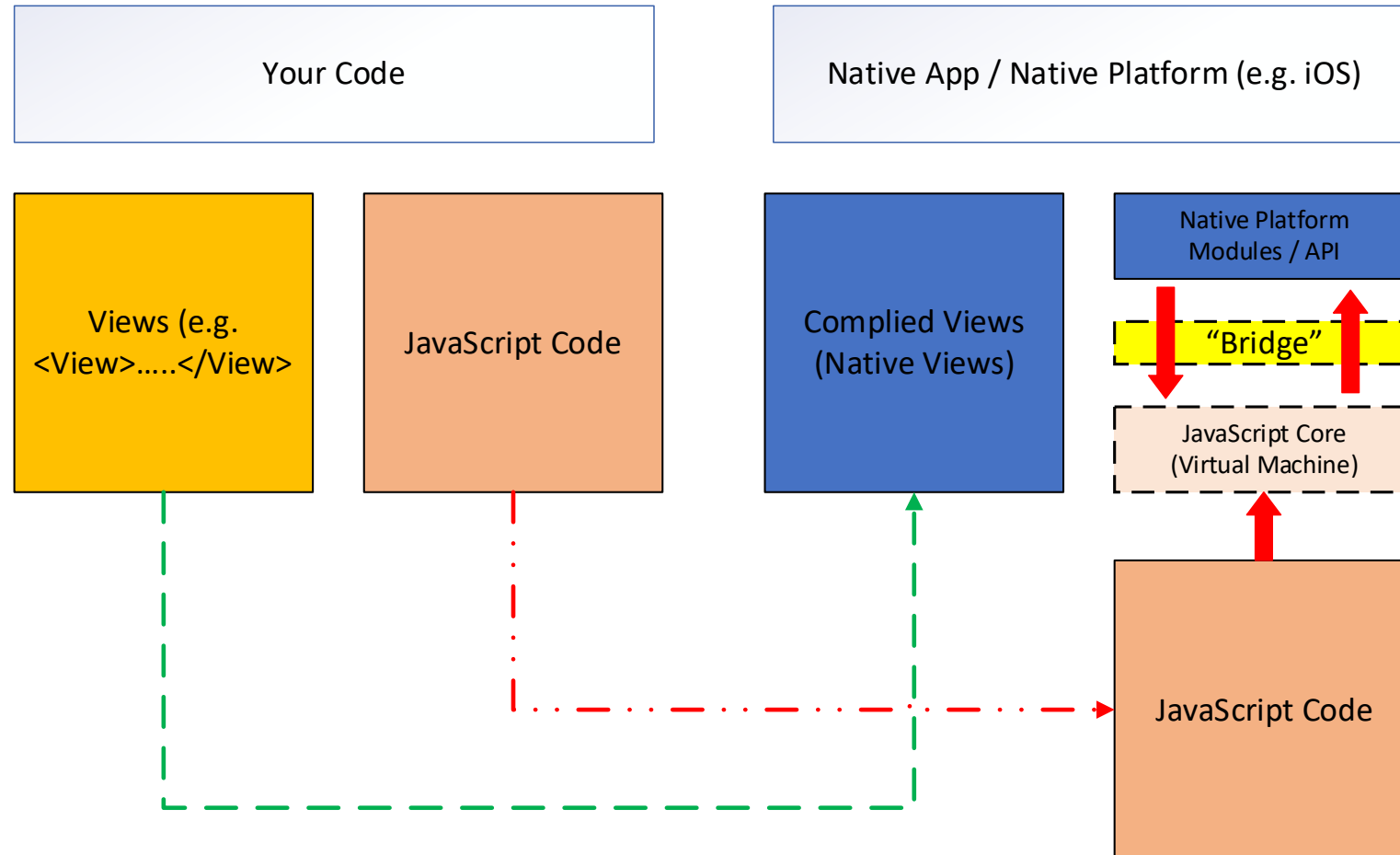


React Native maps re-usable components to the respective platform equivalents.

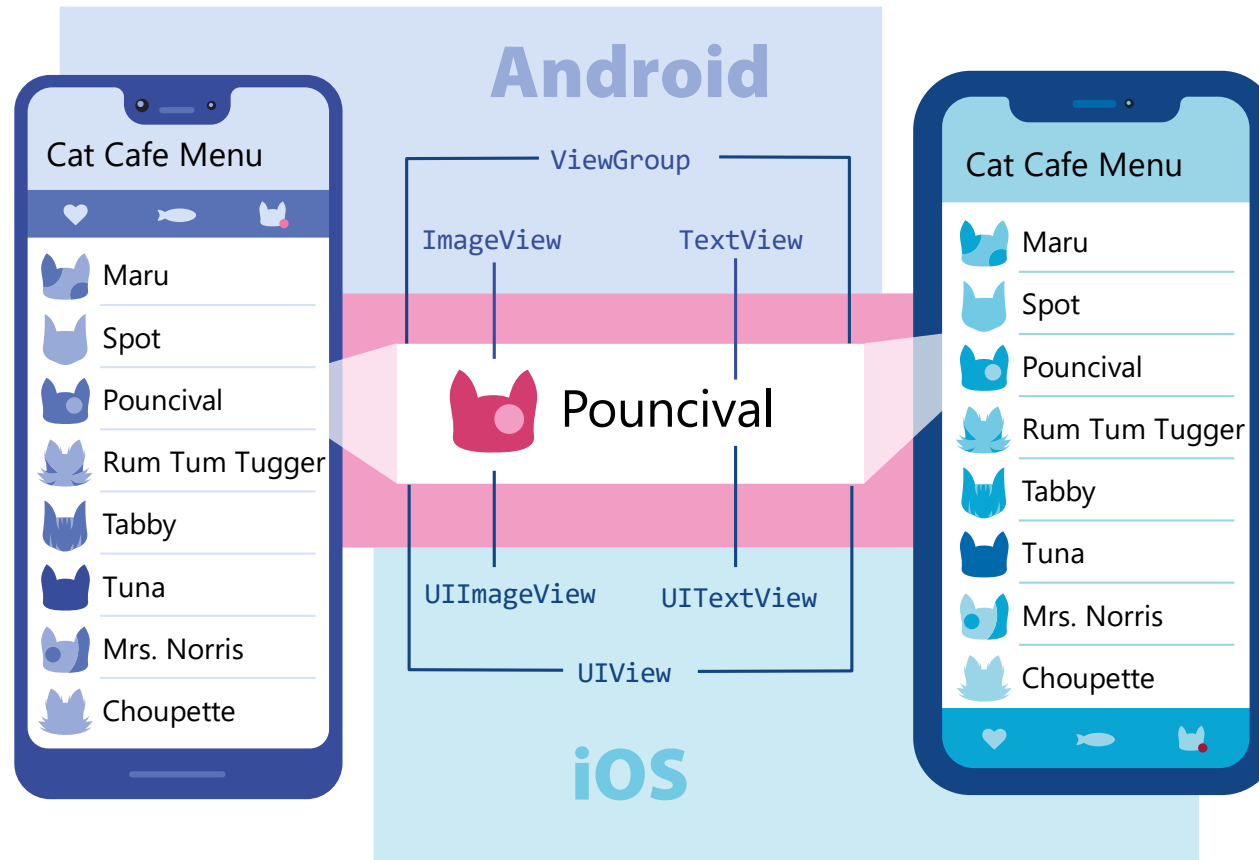
# The JavaScript Part / Our Logic ?



# Behind the Scenes



# Views and mobile development



# Core Components and Native Components

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	DESCRIPTION
<code>&lt;View&gt;</code>	<code>&lt;ViewGroup&gt;</code>	<code>&lt;UIView&gt;</code>	A non-scrolling <code>&lt;div&gt;</code>	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
<code>&lt;Text&gt;</code>	<code>&lt;TextView&gt;</code>	<code>&lt;UITextView&gt;</code>	<code>&lt;p&gt;</code>	Displays, styles, and nests strings of text and even handles touch events
<code>&lt;Image&gt;</code>	<code>&lt;ImageView&gt;</code>	<code>&lt;UIImageView&gt;</code>	<code>&lt;img&gt;</code>	Displays different types of images
<code>&lt;ScrollView&gt;</code>	<code>&lt;ScrollView&gt;</code>	<code>&lt;UIScrollView&gt;</code>	<code>&lt;div&gt;</code>	A generic scrolling container that can contain multiple components and views
<code>&lt;TextInput&gt;</code>	<code>&lt;EditText&gt;</code>	<code>&lt;UITextField&gt;</code>	<code>&lt;input type="text"&gt;</code>	Allows the user to enter text



# Core Components

---

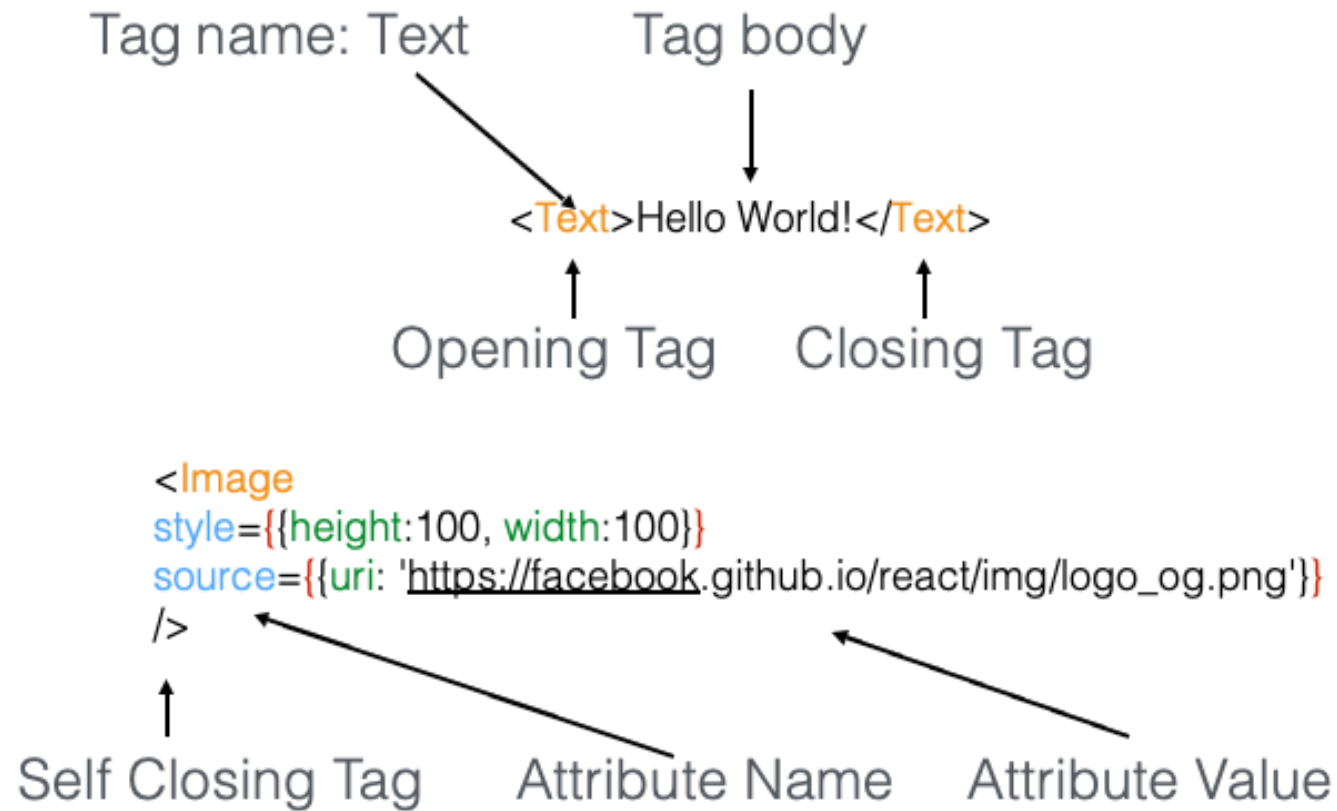
- ❖ React Native ได้เตรียม Core component สำหรับการพัฒนาแอปพลิเคชัน
  - ❖ Basic Components
    - ❖ View - แสดงส่วนติดต่อผู้ใช้ เป็นคอนเทนเนอร์บรรจุคอมโพเนนต์อื่นๆ
    - ❖ Text - แสดงข้อความ
    - ❖ Image - แสดงรูปภาพ
    - ❖ TextInput - รับค่าอินพุตผ่านคีย์บอร์ด
    - ❖ ScrollView – คอนเทนเนอร์บรรจุคอมโพเนนต์อื่นๆ ที่สามารถเลื่อนหน้าจอได้
    - ❖ StyleSheet – คล้าย CSS StyleSheets

# JSX

---

- ❖ JavaScript Extension
- ❖ สำหรับสร้างส่วนติดต่อผู้ใช้ (UI)
- ❖ มีมาร์กอัปลักษณะคล้ายโค้ด HTML (แต่มีกฎที่แตกต่างไป) เป็นรูปแบบที่ช่วยให้สามารถกำหนด element ใน JavaScript ได้
  - ❖ ตัวอย่าง element: `<Text>Hello, I am your cat!</Text>`
- ❖ JSX เป็น JavaScript ทำให้สามารถสร้างตัวแปรต่างๆ ได้ และ embed ตัวแปรใน element ผ่าน `{}` ได้

# JSX Syntax



# Attribute Value

---

- ❖ Using JavaScript Expression as Attribute Value, Use { }

```
<TextInput
  style={{height:40, borderColor: 'gray', borderWidth: 1}}
  value='Useless TextInput'
/>
```

- ❖ Using String as Attribute Value, Use ' '

# Core Components



โครงสร้างใน React Native

โมบายแอป / ส่วนประกอบของ Component

“Translation” ที่อยู่ใน Native UI Widgets ถูกจัดการโดย React Native

ทำการรวบรวม องค์ประกอบหลัก (Core Components) และ องค์ประกอบอื่น ๆ (Other built-in Components) เข้าสู่โมบายแอปพลิเคชัน

<View>

<Text>

<Button> / <Touchable...>

<TextInput>

<Image>

Components

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

Const App = props => {
  Return (
    <View>
      <Text> {props.title}</Text>
    <View>
  );
}
```

# View Component

---

EXAMPLES

# View



❖ Ref. : <https://reactnative.dev/docs/view>

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

const ViewBoxesWithColorAndText = () => {
  return (
    <View
      style={{
        flexDirection: 'row',
        height: 100,
        padding: 20,
      }}>
      <View style={{backgroundColor: 'blue', flex: 0.3}} />
      <View style={{backgroundColor: 'red', flex: 0.5}} />
      <Text>Hello World!</Text>
    </View>
  );
};

export default ViewBoxesWithColorAndText;
```



# Basic CSS

```
const App = () => {
  const style = {
    width: 200,
    height: 200,
    backgroundColor: 'rgb(74,124,226)',
    borderWidth: 2,
    borderColor: 'blue',
    borderRadius: 20,
    padding: 40,
    margin: 80
  }

  const boxStyle = {
    flex: 1,
    backgroundColor: 'pink'
  }

  const textStyle = {
    fontSize: 40,
    fontWeight: 'bold',
    color: 'red'
  }

  return (
    <View style={style}>
      <View style={boxStyle}>
        <Text style={textStyle}>
          Hello!
        </Text>
      </View>
    </View>
  )
}
```



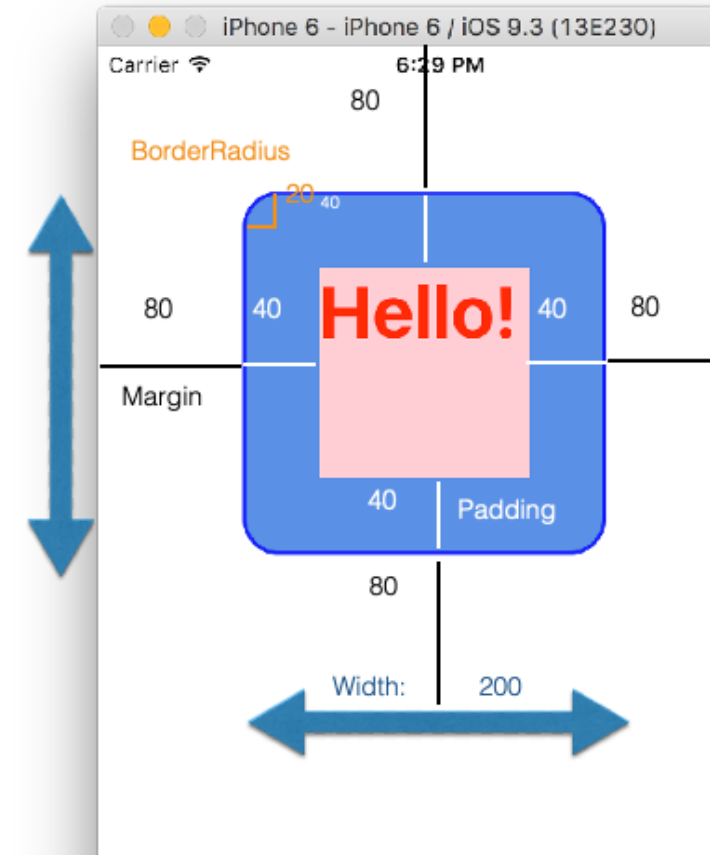


# View : Blue box

```
const style = {
  width: 200,
  height: 200,
  backgroundColor: 'rgb(74,124,226)',
  borderWidth: 2,
  borderColor: 'blue',
  borderRadius: 20,
  padding: 40,
  margin: 80
}

return (
  <View style={style}>
    <View style={boxStyle}>
      <Text style={textStyle}>
        Hello!
      </Text>
    </View>
  </View>
)
```

Height: 200

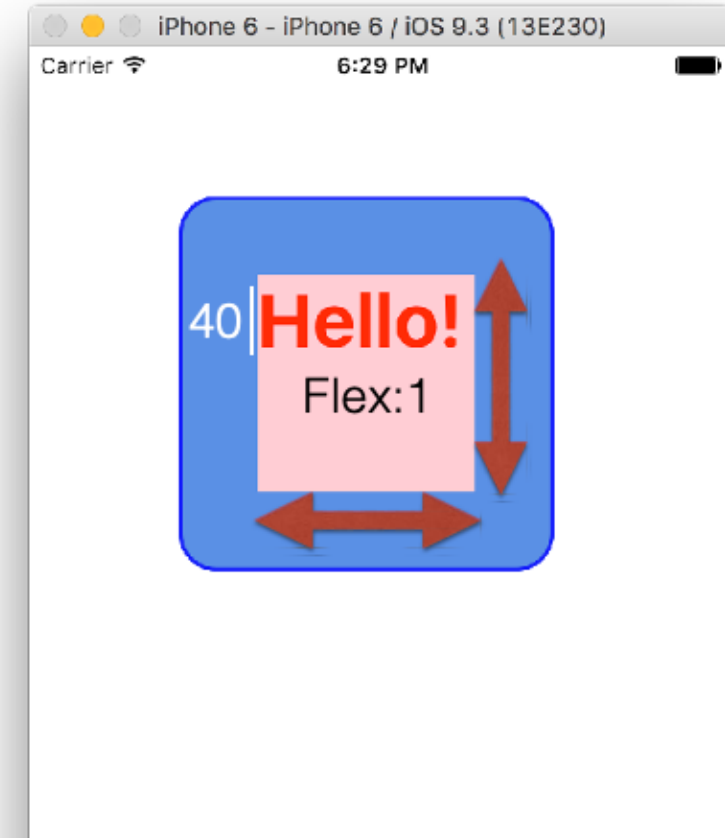


# View : Blue box & text

```
const boxStyle = {
  flex: 1,
  backgroundColor: 'pink'
}

const textStyle = {
  fontSize: 40,
  fontWeight: 'bold',
  color: 'red'
}

return (
  <View style={style}>
    <View style={boxStyle}>
      <Text style={textStyle}>
        Hello!
      </Text>
    </View>
  </View>
)
```



```
return (
  <View style={{flex:1}}>
    <View style={boxStyle}>
    </View>
    <View style={box2Style}>
    </View>
  </View>
)
```

```
const boxStyle = {
  width: 200,
  height: 200,
  backgroundColor: 'rgb(74,124,226)',
  borderWidth: 10,
  borderBottomColor: 'blue',
  borderLeftColor: 'green',
  borderRightColor: 'red',
  borderTopColor: 'red',
  opacity: 0.5,
  borderRadius: 20,
  margin: 20,
  position: 'absolute'
}
```

```
const box2Style = {
  width: 200,
  height: 200,
  backgroundColor: '#faa',
  borderWidth: 10,
  borderColor: 'black',
  opacity: 0.5,
  borderRadius: 20,
  marginTop: 100,
  marginLeft: 40
}
```



```
render() {
  return (
    <View>
      <View style={styles.container}>
        <View style={{height:20}}/>
        { /* Nested Text */ }
        <Text style={{fontWeight: 'bold'}}>
          I am bold
          <Text style={{color: 'red'}}>
            and red and red and red and red
            and red and red and red
          </Text>
        </Text>
        <Text style={{fontSize: 30, fontWeight: '400',
          fontStyle: 'italic'}}>
          Big and italic
        </Text>
        <View style={{width:300, borderColor: '#000',
          borderWidth: 2}}>
          <Text style={{fontWeight: 'bold',
            textAlign: 'right', textDecorationLine: 'underline'}}>
            Underlined Bold and on the right
          </Text>
        </View>
        <Text style={{fontSize:50,
          textShadowOffset:{width:10,height:10},
          textShadowColor: '#aaa', textShadowRadius: 10}}>
          Huge with Shadow
        </Text>
      </View>
    </View>
  );
}
```

```
const styles = StyleSheet.create({
  container: {
    justifyContent: 'center',
    alignItems: 'center'
  }
});
```































































# Colors

---

- ❖ '#f0f' (#rgb)
- ❖ '#f0fc' (#rgba)
- ❖ '#ff00ff' (#rrggbb)
- ❖ '#ff00ff00' (#rrggbbaa)
- ❖ 'rgb(255, 255, 255)'
- ❖ 'rgba(255, 255, 255, 1.0)'
- ❖ 'hsl(360, 100%, 100%)'
- ❖ 'hsla(360, 100%, 100%, 1.0)'
- ❖ 'transparent'
- ❖ 'red'
- ❖ 0xff00ff00 (0xrrggbbaa)

# Colors



-  aliceblue (#f0f8ff)
-  antiquewhite (#faebd7)
-  aqua (#00ffff)
-  aquamarine (#7fffd4)
-  azure (#f0ffff)
-  beige (#f5f5dc)
-  bisque (#ffe4c4)
-  black (#000000)
-  blanchedalmond (#ffebcd)
-  blue (#0000ff)
-  blueviolet (#8a2be2)
-  brown (#a52a2a)
-  burlywood (#deb887)
-  cadetblue (#5f9ea0)
-  chartreuse (#7fff00)
-  chocolate (#d2691e)
-  coral (#ff7f50)
-  cornflowerblue (#6495ed)
-  cornsilk (#fff8dc)
-  crimson (#dc143c)
-  cyan (#00ffff)
-  darkblue (#00008b)
-  darkcyan (#008b8b)
-  darkgoldenrod (#b8860b)
-  darkgray (#a9a9a9)
-  darkgreen (#006400)
-  darkgrey (#a9a9a9)
-  darkkhaki (#bdb76b)
-  darkmagenta (#8b008b)
-  darkolivegreen (#556b2f)
-  darkorange (#ff8c00)
-  darkorchid (#9932cc)
-  darkred (#8b0000)
-  darksalmon (#e9967a)
-  darkseagreen (#8fbc8f)
-  darkslateblue (#483d8b)
-  darkslategray (#2f4f4f)
-  darkslategrey (#2f4f4f)
-  darkturquoise (#00ced1)
-  darkviolet (#9400d3)
-  deeppink (#ff1493)
-  deepskyblue (#00bfff)
-  dimgray (#696969)
-  dimgrey (#696969)
-  dodgerblue (#1e90ff)
-  firebrick (#b22222)
-  floralwhite (#fffaf0)
-  forestgreen (#228b22)
-  fuchsia (#ff00ff)
-  gainsboro (#dcdcdc)
-  ghostwhite (#f8f8ff)
-  gold (#ffd700)
-  goldenrod (#daa520)
-  gray (#808080)
-  green (#008000)
-  greenyellow (#adff2f)
-  grey (#808080)
-  honeydew (#f0ffff)
-  hotpink (#ff69b4)
-  indianred (#cd5c5c)

<https://facebook.github.io/react-native/docs/colors.html>

# Flexbox Layout

---

- ❖ Flexbox => CSS Flexible Box Layout (in W3C Last Call Working Draft)
- ❖ ประสิทธิภาพการแสดงผลของการจัดวาง จัดแนว และกระจายพื้นที่ระหว่างรายการในคอนเทนเนอร์ สามารถทำได้แม้ว่าจะไม่ทราบขนาดและ/หรือ ปรับเปลี่ยนรูปแบบไดนามิก (flex)
- ❖ คอนเทนเนอร์สามารถปรับเปลี่ยนความกว้าง/ความสูงของรายการ และเพื่อให้การใช้พื้นที่ว่างสำหรับการแสดงผลที่มีอยู่ให้ได้ดีที่สุด
- ❖ Flexbox มีลักษณะ direction-agnostic ซึ่งสามารถรองรับการใช้งานที่ซับซ้อน (โดยเฉพาะเมื่อต้องเปลี่ยนการแสดงผล เช่น การปรับเปลี่ยนจากแนวดิ่งไปสู่แนวตั้ง การปรับขนาด การยืด การย่อ และอื่น ๆ)

# Layout with Flexbox

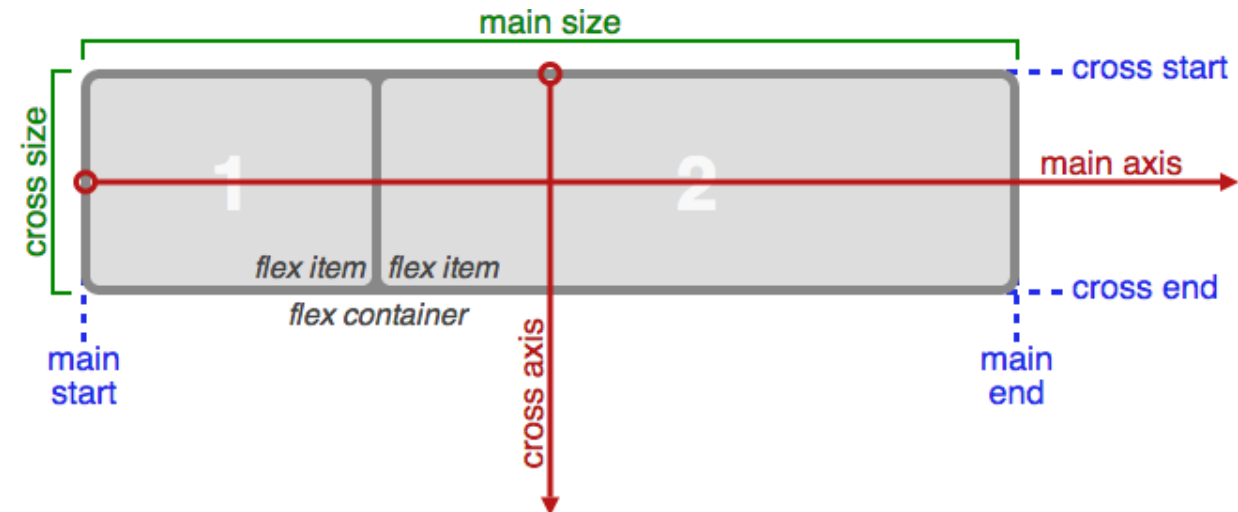
---

- ❖ คอมพิวเตอร์สามารถระบุตำแหน่งของการแสดงผลที่อยู่ภายใต้อัลกอริทึมของ Flexbox ซึ่ง Flexbox ถูกออกแบบมาเพื่อให้สามารถแสดงผลที่สอดคล้องกับขนาดหน้าจอต่าง ๆ
- ❖ ปกติการใช้ Flexbox มีการใช้พารามิเตอร์ flexDirection, alignItems และ justifyContent การผสมผสานกันเพื่อให้ได้ตำแหน่งที่ต้องการแสดงผลได้ถูกต้อง



# Flexbox concept

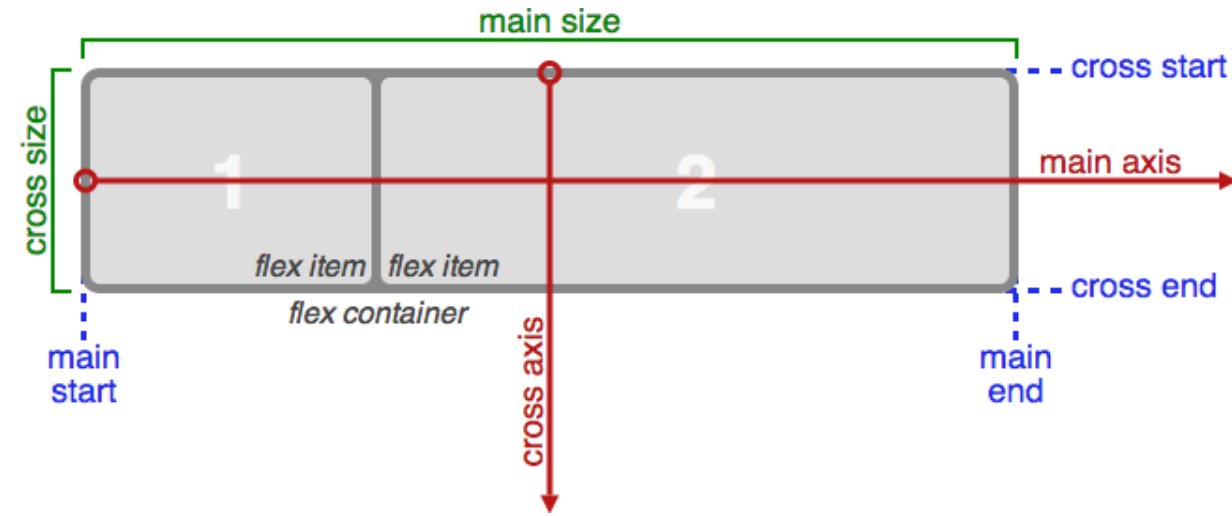
- ❖ **Main axis** : แกนหลักของการของ Flex container ซึ่งถูกกำหนดโดยพารามิเตอร์ flexDirection.
- ❖ **Main-start และ Main-end** : การกำหนดตำแหน่งของการวาง Flex items ในแต่ละ container โดยปกติการแสดงผลเริ่มจาก main-start จนถึง main-end ของหน้าจออุปกรณ์เคลื่อนที่
- ❖ **Main-size** : การกำหนดขนาดของ Flex item เช่น ความกว้าง (width) หรือ ความสูง (height) สำหรับการแสดงผลตามแกนหลักบนหน้าจออุปกรณ์เคลื่อนที่



กรณีนี้ flexDirection = row (horizontal)

# Flexbox concept

- ❖ **Cross axis** : แกนหลักที่สองของการแสดงผล (เส้นสีแดงแนวขวาง) ซึ่งการแสดงผลลัพท์ตรงกันข้ามกับแกนหลักที่อยู่ใน flexDirection
- ❖ **Cross-start และ Cross-end** : การกำหนดตำแหน่งของการวาง Flex items ในแต่ละแถวหรือคอลัมน์ (Flex lines) ของ Container โดยปกติการแสดงผลเริ่มจาก Cross-start จนถึง Cross-end ของหน้าจออุปกรณ์เคลื่อนที่
- ❖ **Cross-size** : การกำหนดขนาดของ Flex item เช่น ความกว้าง (width) หรือ ความสูง (height) ซึ่งแสดงผลแนวขวางบนหน้าจออุปกรณ์เคลื่อนที่

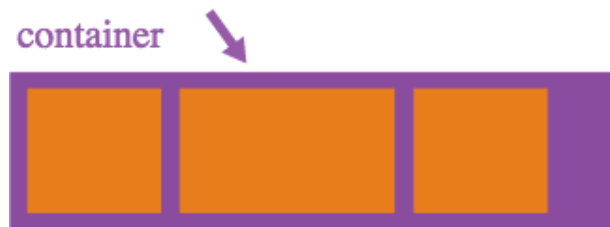


กรณีนี้ flexDirection = row (horizontal)

# ประเภทของ Flex properties

## ❖ Containers

- ❖ flexDirection
- ❖ justifyContent
- ❖ alignItems
- ❖ flexWrap



## ❖ Items

- ❖ flex
- ❖ alignSelf



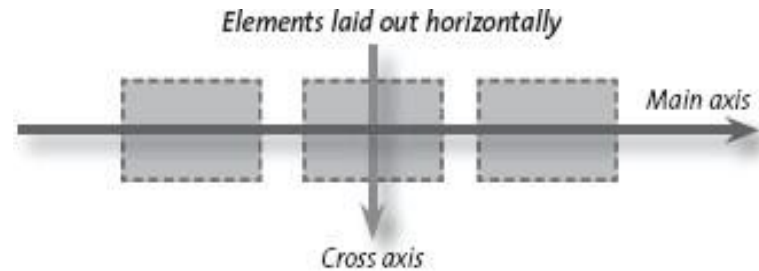
# Containers

---

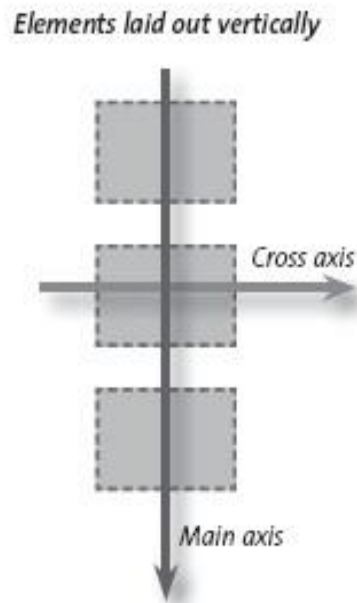
- ❖ flexDirection
- ❖ justifyContent
- ❖ alignItems
- ❖ flexWrap



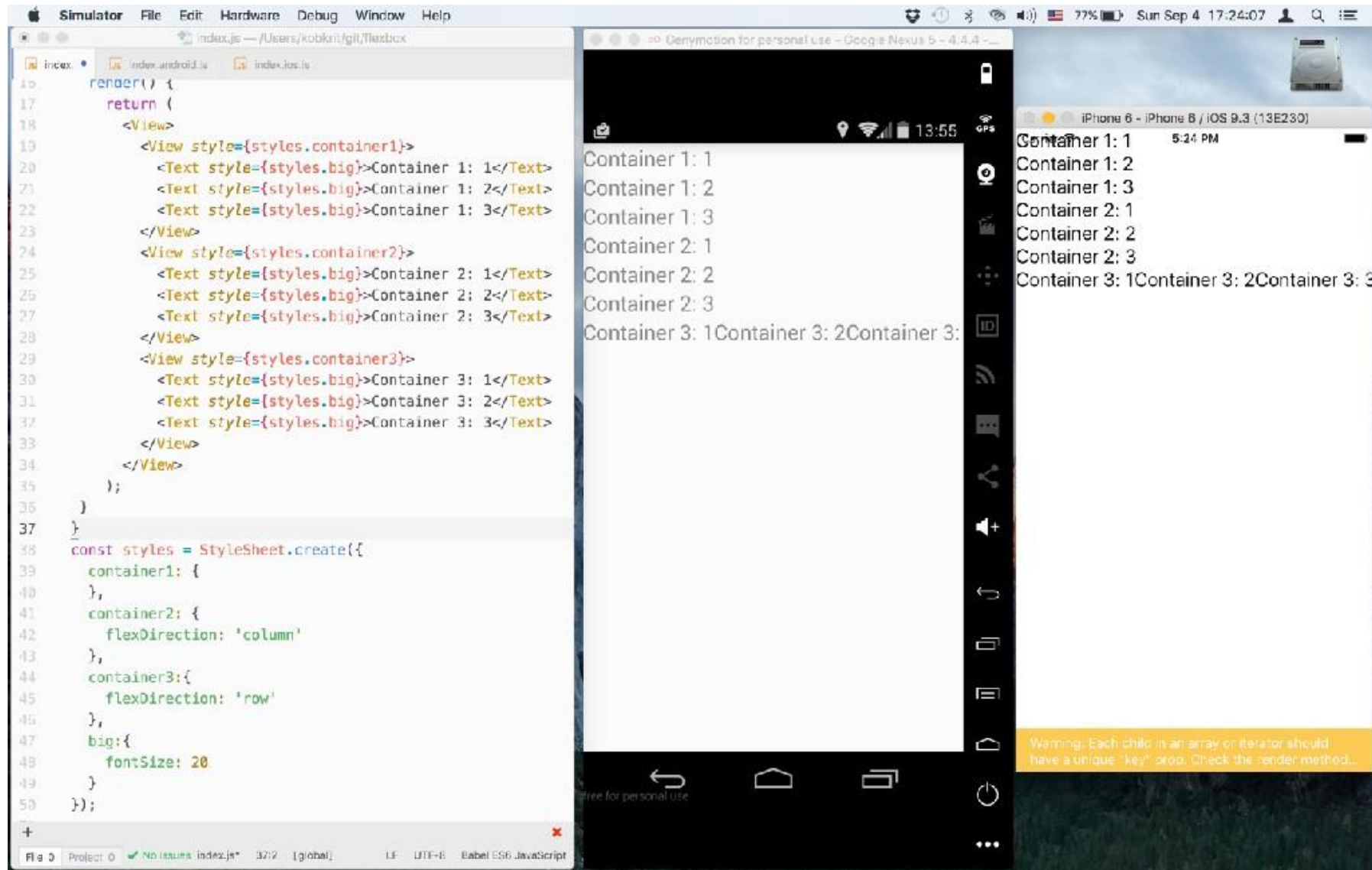
# FlexDirection : (Container)



- ❖ Horizontally → flexDirection: row;
- ❖ row (default)
- ❖ row-reverse



- ❖ Vertically → flexDirection: column;
- ❖ column (default)
- ❖ column-reverse

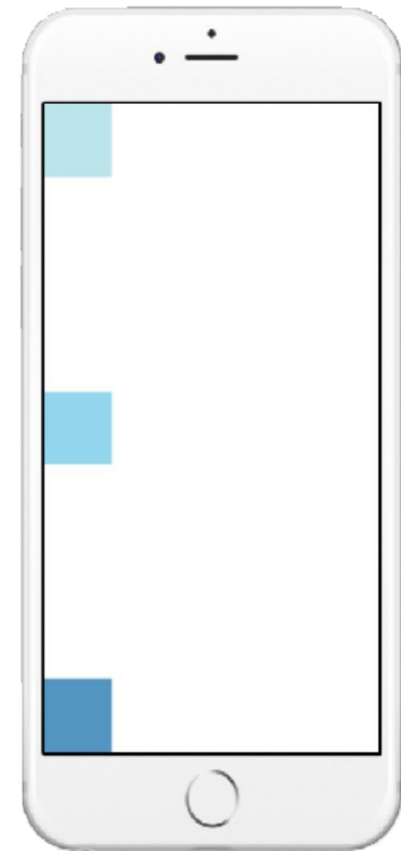


Default flexDirection in React Native is column

# JustifyContent : (Container)

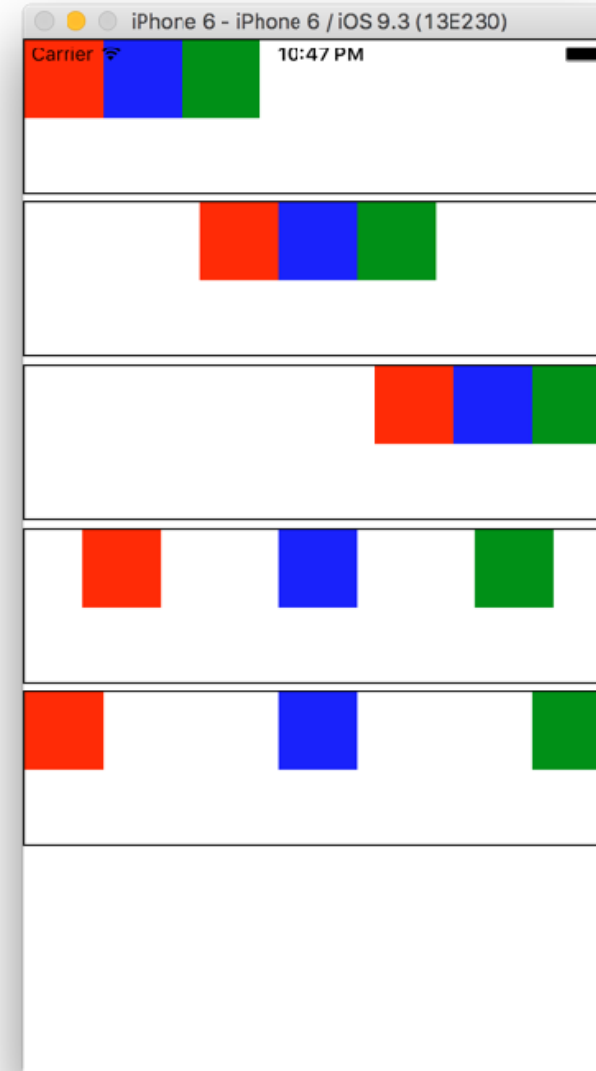
- ❖ **JustifyContent** : การกำหนดการแสดงผลของสิ่งที่ต้องการที่อยู่ภายใน Component แสดงผลในรูปแบบของการกระจายบนแกนหลัก
- ❖ รูปแบบการแสดงผลสามารถกำหนดเป็น start, center, end, หรือ spaced ได้ โดยสามารถระบุพารามิเตอร์ ดังนี้
  - ❖ flex- start
  - ❖ center
  - ❖ flex-end
  - ❖ space-around
  - ❖ space-between.

❖ ค่าเริ่มต้น คือ **flex-start**



`flexDirection: 'column', justifyContent: 'space-between'`

```
class flexbox extends Component {
  render() {
    return (
      <View>
        <View style={[[styles.container, {justifyContent:'flex-start'}]]}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={[[styles.container, {justifyContent:'center'}]]}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={[[styles.container, {justifyContent:'flex-end'}]]}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={[[styles.container, {justifyContent:'space-around'}]]}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={[[styles.container, {justifyContent:'space-between'}]]}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
      </View>
    );}}
  const styles = StyleSheet.create({
    container: {
      marginBottom:5, height:100,
      flexDirection:'row',
      borderColor:'black', borderWidth:1
    },
    item1:{
      width:50, height:50, backgroundColor: 'red'
    },
    item2:{
      width:50, height:50, backgroundColor: 'blue'
    },
    item3:{
      width:50, height:50, backgroundColor: 'green'
    }
  });
}
```





# AlignItems : (Container)

❖ **AlignItems** : การกำหนดการแสดงผลของสิ่งที่ต้องการที่อยู่ภายใน Component แสดงผลในรูปแบบของการกระจายบนแกนรอง (การแสดงผลลัพท์จะแสดงตรงกันข้ามกัน หากแกนหลักถูกกำหนดแบบแนวนอน [row] แกนรองจะแสดงผลแนวตั้ง [column] )

❖ รูปแบบการแสดงผลสามารถกำหนดเป็น start, center, end, หรือ spaced ได้ โดยสามารถระบุพารามิเตอร์ ดังนี้

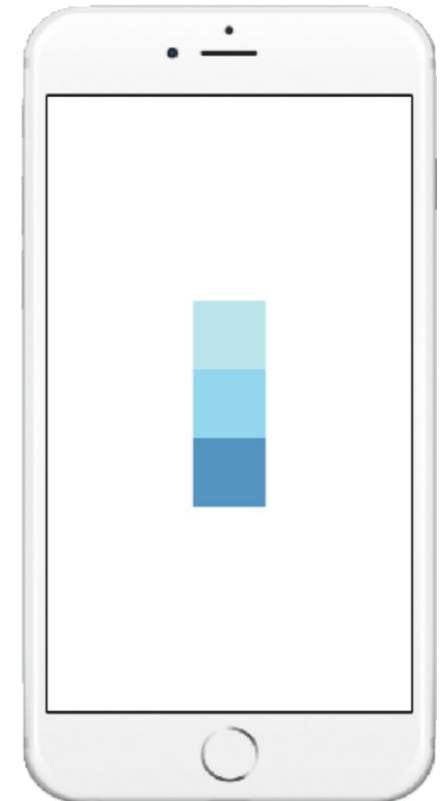
❖ flex- start

❖ center

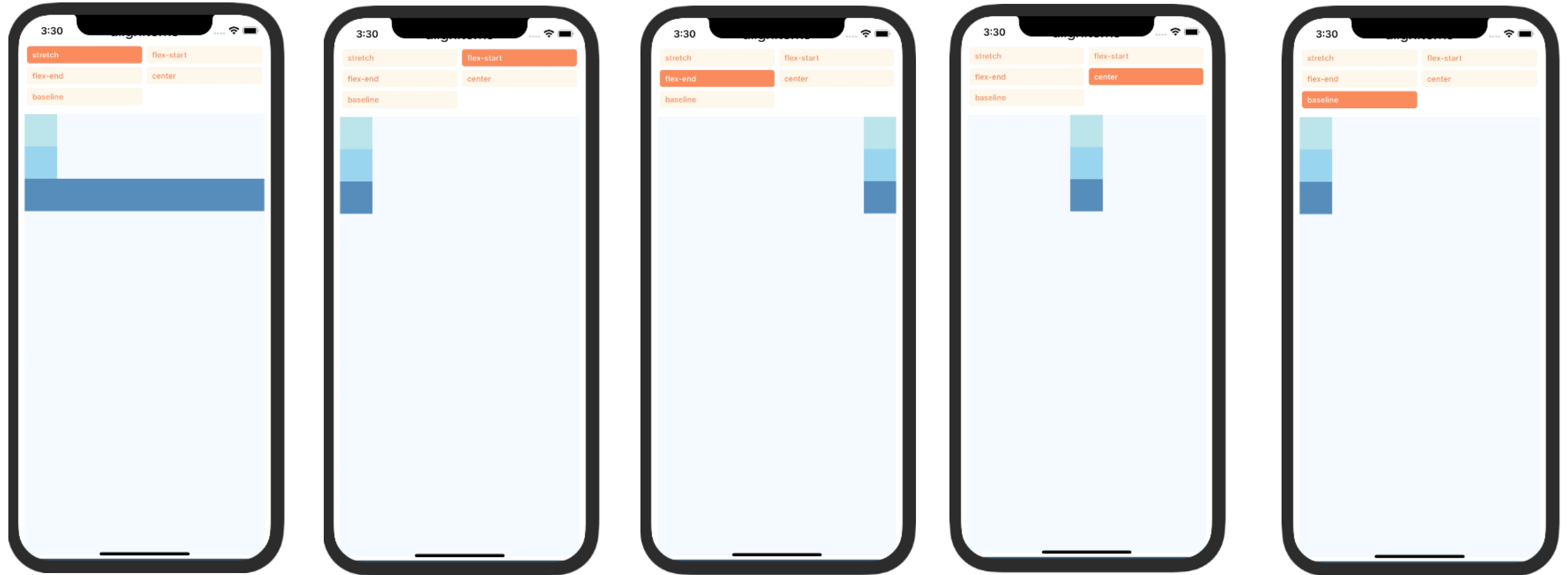
❖ flex- end

❖ stretch

❖ ค่าปกติคือ **flex-start**



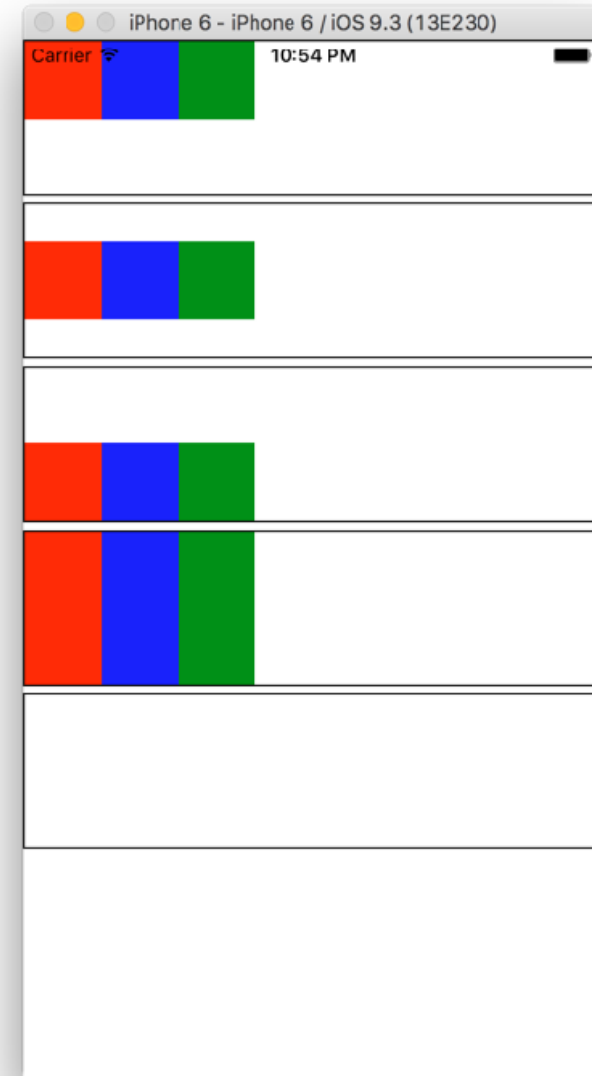
flexDirection: 'column', justifyContent: 'center', alignItems: 'center'



Source Code : <https://reactnative.dev/docs/flexbox>

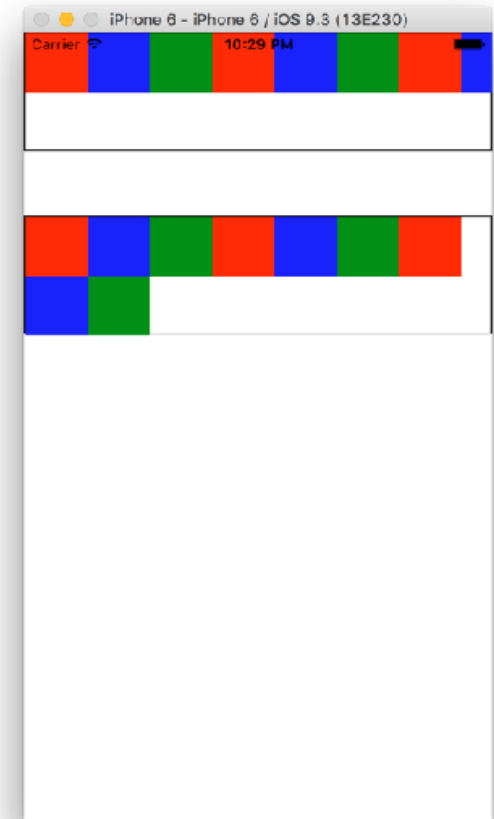
```
class flexbox extends Component {
  render() {
    return (
      <View>
        <View style={styles.container, {alignItems:'flex-start'}}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={styles.container, {alignItems:'center'}}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={styles.container, {alignItems:'flex-end'}}>
          <View style={styles.item1}></View><View style={styles.item2}></View><View style={
        </View>
        <View style={styles.container, {alignItems:'stretch'}}>
          <View style={styles.item4}></View><View style={styles.item5}></View><View style={
        </View>
        <View style={styles.container, {alignItems:'flex-start'}}>
          <View style={styles.item4}></View><View style={styles.item5}></View><View style={
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    marginBottom:5, height:100,
    flexDirection:'row',
    borderColor:'black', borderWidth:1
  },
  item1:{width:50, height:50, backgroundColor: 'red'},
  item2:{width:50, height:50, backgroundColor: 'blue'},
  item3:{width:50, height:50, backgroundColor: 'green'},
  item4:{width:50, height:50, backgroundColor: 'red'},
  item5:{width:50, height:50, backgroundColor: 'blue'},
  item6:{width:50, height:50, backgroundColor: 'green'}
});
```



# FlexWrap : (Container)

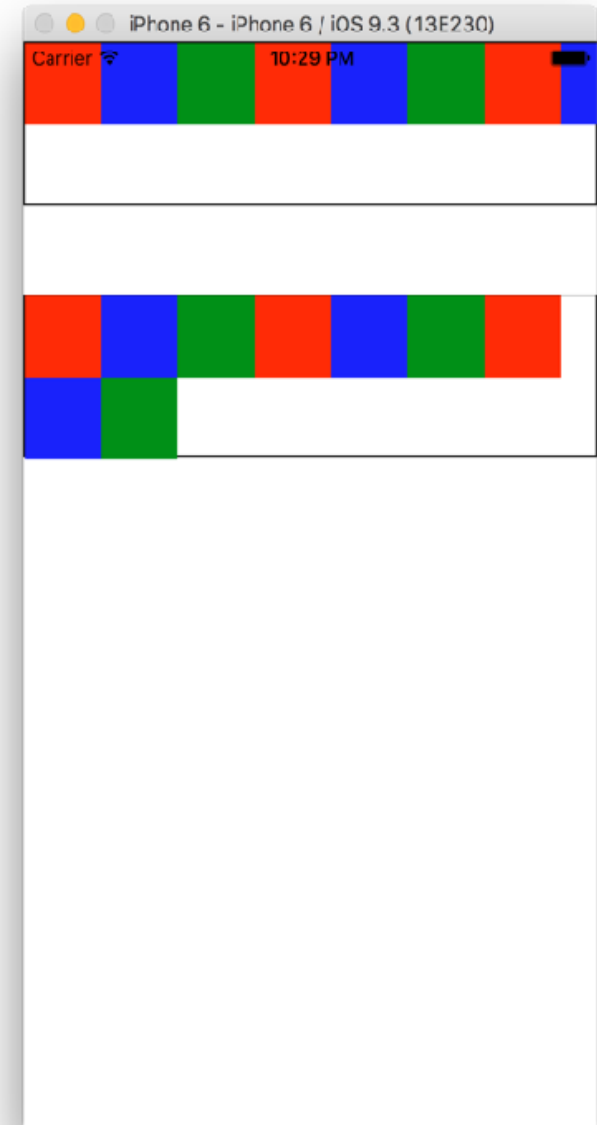
- ❖ โดยปกติ Flex Item จะทำการแสดงผลให้อยู่ภายในบรรทัดเดียวกัน
- ❖ FlexWrap คือ พารามิเตอร์สามารถเปลี่ยนรูปแบบการแสดงผลได้ตามพารามิเตอร์ที่กำหนดใน property
- ❖ ตำแหน่งของการแสดงผลจะแสดงตามบทบาทที่ได้รับจากการกำหนดทิศทางการแสดงผล เช่น แสดงผลให้อยู่ในบรรทัดเดียว หรือ แสดงผลโดยขึ้นบรรทัดใหม่หากการแสดงผลไม่ครอบคลุมหนึ่งบรรทัด เป็นต้น การกำหนดพารามิเตอร์ มีดังนี้
  - ❖ nowrap
  - ❖ wrap
- ❖ ค่าปกติ คือ **nowrap**



nowrap vs wrap

```
class flexbox extends Component {
  render() {
    return (
      <View>
        <View style={styles.container, {flexWrap: 'nowrap'}}>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View><View style={styles.item1}></View>
          <View style={styles.item2}></View><View style={styles.item3}></View>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View>
        </View>
        <View style={{height:50}}/>
        <View style={styles.container, {flexWrap: 'wrap'}}>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View><View style={styles.item1}></View>
          <View style={styles.item2}></View><View style={styles.item3}></View>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View>
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    marginBottom:5, height:100,
    flexDirection:'row',
    borderColor:'black', borderWidth:1
  },
  item1:{ width:50, height:50, backgroundColor: 'red'},
  item2:{ width:50, height:50, backgroundColor: 'blue' },
  item3:{ width:50, height:50, backgroundColor: 'green' }
});
```



# Items

---

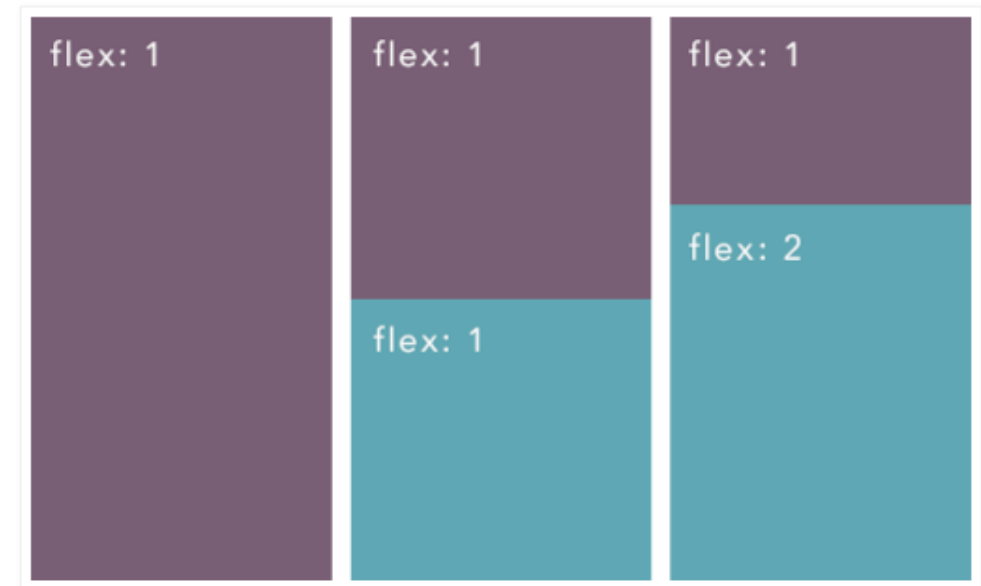
- ❖ flex
- ❖ alignSelf
- ❖ AlignContent



# Flex : (Item)

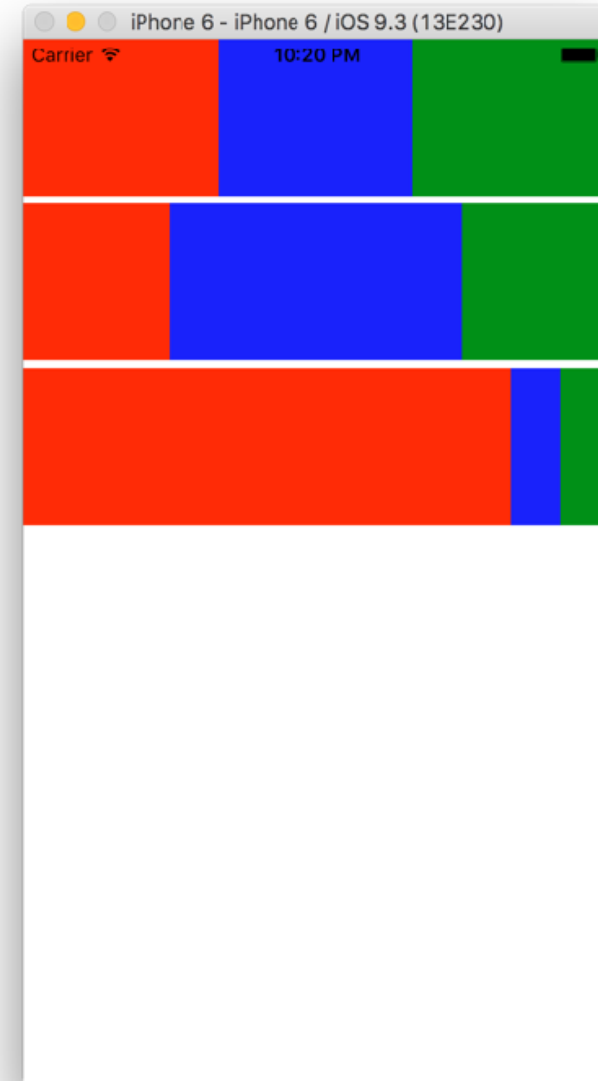
❖ “Flex” คือ รูปแบบของ CSS ที่ถูกนำมาประยุกต์ใช้สำหรับการแสดงผลของแต่ละ element ที่อยู่ภายใน Item หลักการทำงานจะทำอยู่บนแกนหลักเพื่อแสดงผลตามแต่ละ Item ที่กำหนด โดยทำการคำนวณหาพื้นที่ว่างทั้งหมดของจอภาพ และแสดงผลจะแสดงตาม element ที่มีอยู่ใน Item ซึ่งหากมีมากกว่า 1 element โปรแกรมจะทำการแบ่งหน้าจอเพื่อแสดงผลลัพท์

- ❖ {flex: (positive integer number)}, e.g.,
- ❖ flex : 1
- ❖ flex: 1, backgroundColor: "red"
- ❖ flex: 2, backgroundColor: "darkorange"
- ❖ flex: 3, backgroundColor: "green"



```
class flexbox extends Component {
  render() {
    return (
      <View>
        <View style={styles.container}>
          <View style={[styles.item1, {flex:1}]}></View>
          <View style={[styles.item2, {flex:1}]}></View>
          <View style={[styles.item3, {flex:1}]}></View>
        </View>
        <View style={styles.container}>
          <View style={[styles.item1, {flex:1}]}></View>
          <View style={[styles.item2, {flex:2}]}></View>
          <View style={[styles.item3, {flex:1}]}></View>
        </View>
        <View style={styles.container}>
          <View style={[styles.item1, {flex:10}]}></View>
          <View style={[styles.item2, {flex:1}]}></View>
          <View style={[styles.item3, {flex:1}]}></View>
        </View>
      </View>
    );
  }
}

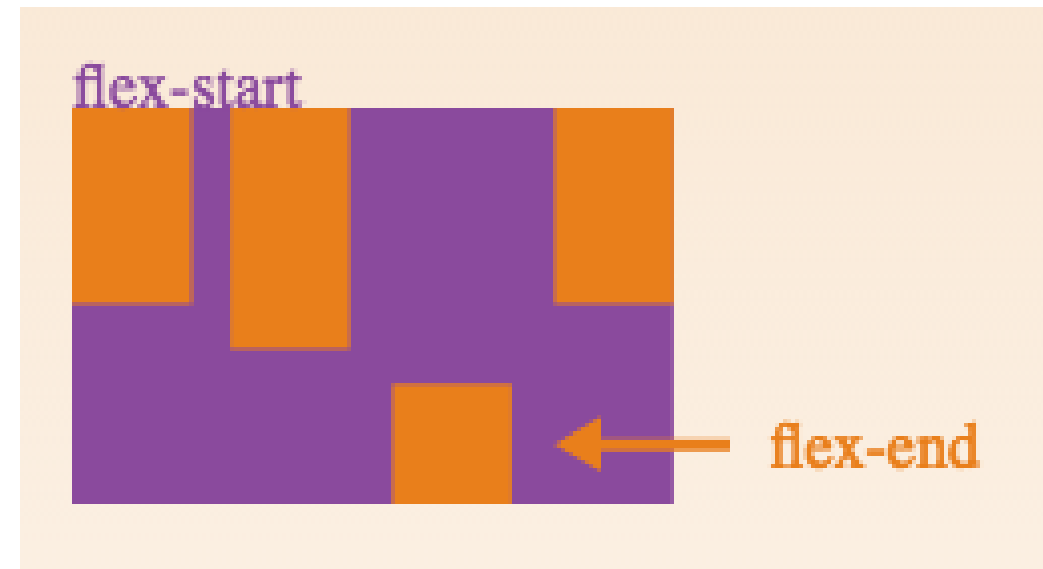
const styles = StyleSheet.create({
  container: {
    marginBottom:5,
    height:100,
    flexDirection:'row'
  },
  item1:{
    backgroundColor: 'red'
  },
  item2:{
    backgroundColor: 'blue'
  },
  item3:{
    backgroundColor: 'green'
  }
});
```

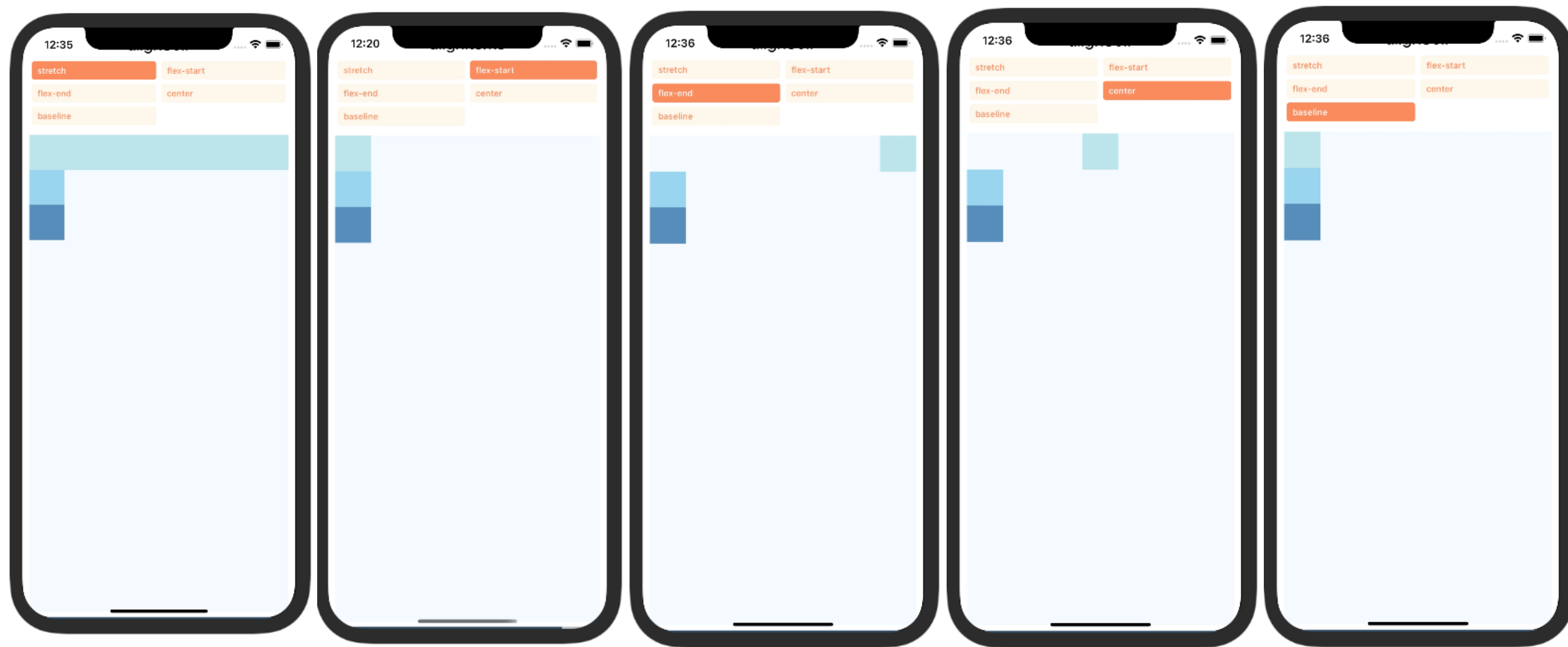




# AlignSelf : (Item)

- ❖ **AlignSelf** : การกำหนดการแสดงผลของตัว Item นั้น ๆ ที่ต้องการซึ่งอยู่ภายใน Component แสดงผลในรูปแบบของการกระจายบนแกนรอง (การแสดงผลลัพธ์จะแสดงผลทับ alignItems ที่อยู่ภายใน Container)
- ❖ รูปแบบการแสดงผลสามารถกำหนดพารามิเตอร์ ดังนี้
  - ❖ auto
  - ❖ flex- start
  - ❖ center
  - ❖ flex- end
  - ❖ stretch
- ❖ ค่าปกติคือ **auto** (การแสดงผลจะทำตามที่กำหนดไว้ใน alignItems ของ container นั้น ๆ )

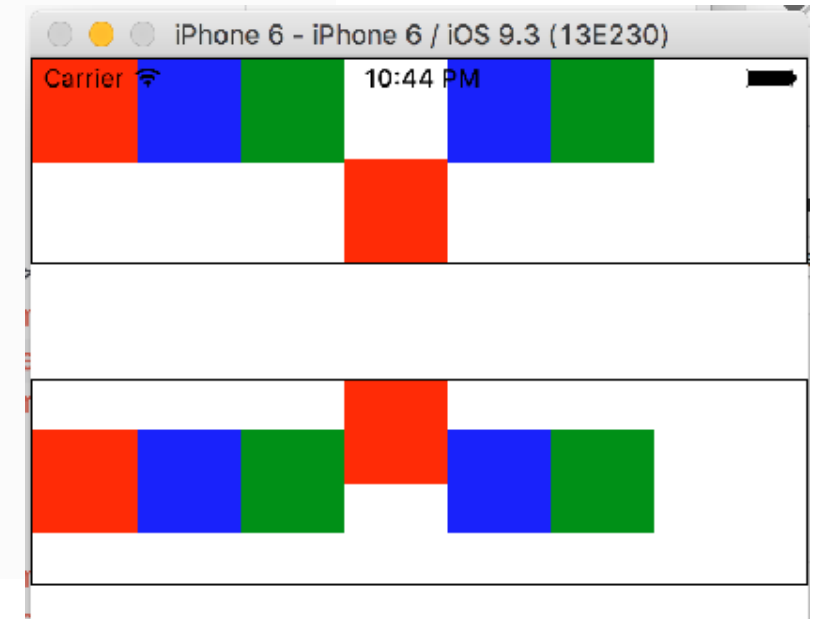




Source Code : <https://reactnative.dev/docs/flexbox>

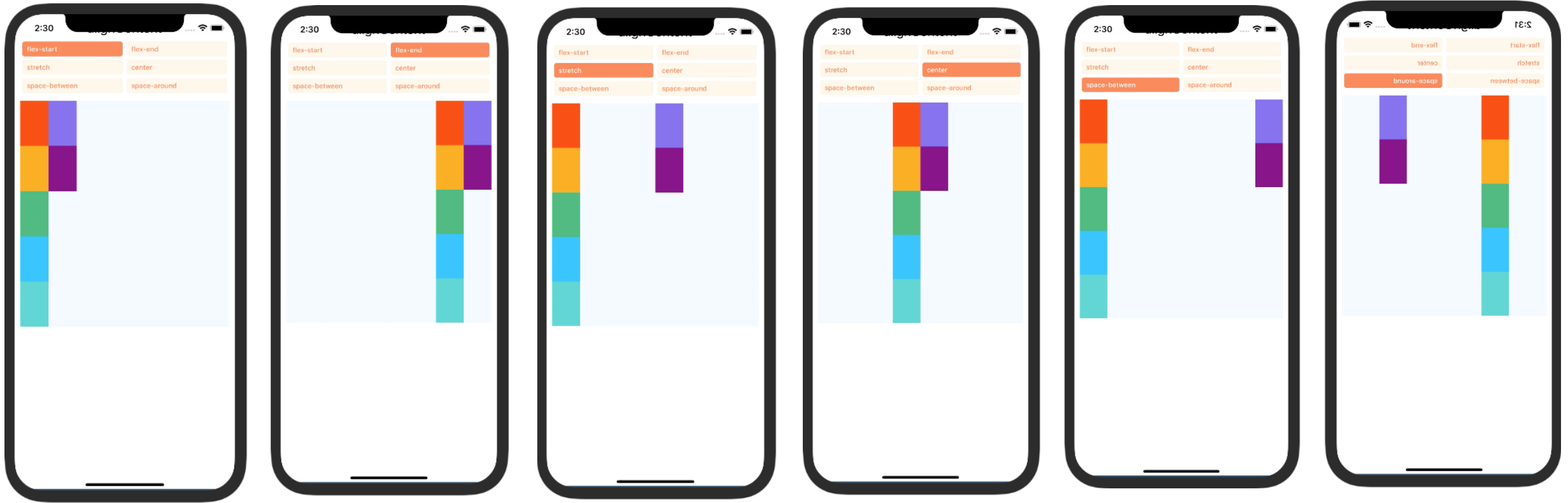
```
class flexbox extends Component {
  render() {
    return (
      <View>
        <View style={ [styles.container, {alignItems:'flex-start'}]}>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View><View style={ [styles.item1, {alignSelf:'flex-end'}]}></View>
          <View style={styles.item2}></View><View style={styles.item3}></View>
        </View>
        <View style={{height:50}}/>
        <View style={ [styles.container, {alignItems:'center'}]}>
          <View style={styles.item1}></View><View style={styles.item2}></View>
          <View style={styles.item3}></View><View style={ [styles.item1, {alignSelf:'flex-start'}]}></View>
          <View style={styles.item2}></View><View style={styles.item3}></View>
        </View>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    marginBottom:5, height:100,
    flexDirection:'row',
    borderColor:'black', borderWidth:1
  },
  item1:{ width:50, height:50, backgroundColor: 'red'},
  item2:{ width:50, height:50, backgroundColor: 'blue' },
  item3:{ width:50, height:50, backgroundColor: 'green' }
});
```



# Align Content : (Item)

- ❖ **AlignContent** : การกำหนดการแสดงผลของตัว Item ที่ต้องการภายใน Component แสดงผลในรูปแบบของแถวรอง ซึ่งการแสดงผลลัพท์ในแนวตั้งและขึ้นบรรทัดใหม่หากพื้นที่แสดงผลไม่พอ หรือ ระบุคำสั่งเป็น flexWrap
- ❖ รูปแบบการแสดงผลสามารถกำหนดพารามิเตอร์ ดังนี้
  - ❖ flex- start
  - ❖ flex- end
  - ❖ center
  - ❖ stretch
  - ❖ space-between or space-around
- ❖ ค่าปกติคือ **stretch** (เมื่อถูกใช้แสดงผลใน Yoga บน web )



Source Code : <https://reactnative.dev/docs/flexbox>

# Other Core Components

---

- ❖ Text Component
  - ❖ <https://reactnative.dev/docs/text>
- ❖ Image Component
  - ❖ <https://reactnative.dev/docs/image>
- ❖ TextInput
  - ❖ <https://reactnative.dev/docs/textinput>
- ❖ ScrollView
  - ❖ <https://reactnative.dev/docs/scrollview>
- ❖ StyleSheet
  - ❖ <https://reactnative.dev/docs/stylesheet>

```
const App = () => {
  return (
    <ScrollView>
      <Text>Some text</Text>
      <View>
        <Text>Some more text</Text>
        <Image
          source={{
            uri: 'https://reactnative.dev/docs/assets/p_cat2.png',
          }}
          style={{width: 200, height: 200}}
        />
      </View>
      <TextInput
        style={{
          height: 40,
          borderColor: 'gray',
          borderWidth: 1,
        }}
        defaultValue="You can type in me"
      />
    </ScrollView>
  );
};

export default App;
```

Some text  
Some more text



You can type in me



My Device

iOS

Android

Web

## Lab 2.1

❖ ให้นศ.ทดลองสร้างคอมโพเนนต์ที่แสดงหน้าจอ  
ดังนี้





## Lab 2.2

- ❖ ให้ศ.ทดลองสร้างคอมโพเนนต์ที่แสดงหน้าจอแสดงหลักสูตรระดับปริญญาตรี ดังนี้
- ❖ ผู้ใช้สามารถเลื่อนหน้าจอเพื่อดูข้อมูลด้านล่างได้
- ❖ เมื่อมีการกดบริเวณข้อมูลของแต่ละหลักสูตร บริเวณที่ถูกกดจะแสดงสีจางลง และเป็นปกติ เมื่อเลิกกด
  - ❖ <https://reactnative.dev/docs/touchableopacity>



เมื่อเลื่อนหน้าจอ