Styling + Flexbox

Styling

Styling

Styles can be created one of two ways: inline or in a StyleSheet declaration.

Styling

inline styles

```
<Text style={{ color: 'red' }}>Hello World</Text>
<View style={{ width: 300 }}>
```

Styling

inline styles Object - but better to use StyleSheet (more performant)

```
const styles = {
  text: {
    color: 'red',
  }
}
<Text style={styles.text}>Hello World</Text>
```

Styling

`StyleSheet.create({...})`

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

const styles = StyleSheet.create({
  container: {
    width: 300,
    height: 130,
  },
  text: {
    color: 'red',
  },
});
<View style={styles.container}>
  <Text style={styles.text}>Hello World</Text>
</View>
```

Styling

`StyleSheet.create({...})`

Array of styles

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

const styles = StyleSheet.create({
  text: {
    color: 'red',
  },
  bigText: {
    fontSize: 30,
  },
});
<Text style={[styles.text, styles.bigText]}>Hello World</Text>
```

Styling

Combining inline styles with StyleSheet

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

const styles = StyleSheet.create({
  text: {
     color: 'red',
   },
});
<Text style={[styles.text, { fontSize: 32 } ]}>Hello World</Text>
```

Styling

The style names and values usually match how CSS works on the web, except names are written using camel casing, e.g `backgroundColor` rather than `background-color`.

Styling

Main View non flex styles

zIndex – number

width – number

top / left / bottom / right – number

position – string (absolute, relative)

padding – number

minHeight / minWidth / maxWidth / maxHeight – number

margin – number

borderWidth – number

borderPositionWidth – number

borderColor – color

Styling

Main Text styles

```
color – color
fontFamily – string
fontSize – number
fontStyle – string
fontWeight – string (‘400’, ‘800’, ‘bold’)
lineHeight – number
textAlign – number
textDecorationLine – string
textShadowColor – color
texShadowOffset – {width, height}
textShadowRadius – number
```

Styling

Colors

red-green-blue

```
color: 'rgb(255,153, 0)'
```

red-green-blue-alpha

```
color: 'rgba(255,153, 0, .5)',
```

named color

```
color: 'red',
```

Hex color

```
color: ‘#FB',
```

Styling

Dynamic styling

```
state = {
  warning: true,
}

render() {
  return (
    <Text style={{ color: this.state.warning ? 'red' : 'black' }}>Hello World</Text>
  );
}
```

Styling

Dynamic styling

```
state = {
  warning: true,
}
```

```
render() {
  const fontColor = this.state.warning ? 'red' : 'black';
  return (
    <View>
      <Text style={{ color: fontColor }}>Hello World</Text>
    </View>
  );
}
```

Flexbox

What is Flexbox?

Flexbox

In React Native every screen that you see is a collection of components.

Flexbox

Flexbox is the layout model that React Native uses to allow you to lay out your components without having to rely on absolute positions.

Flexbox

Each screen consists of at least one parent component and a variable number of child components.

Flexbox

A component can specify the layout of its children using the flexbox algorithm. Flexbox is designed to provide a consistent layout on different screen sizes.

You will normally use a combination of flexDirection, alignItems, and justifyContent to achieve the right layout.

Example:

https://github.com/hgale/FlexBoxDemo

Flexbox

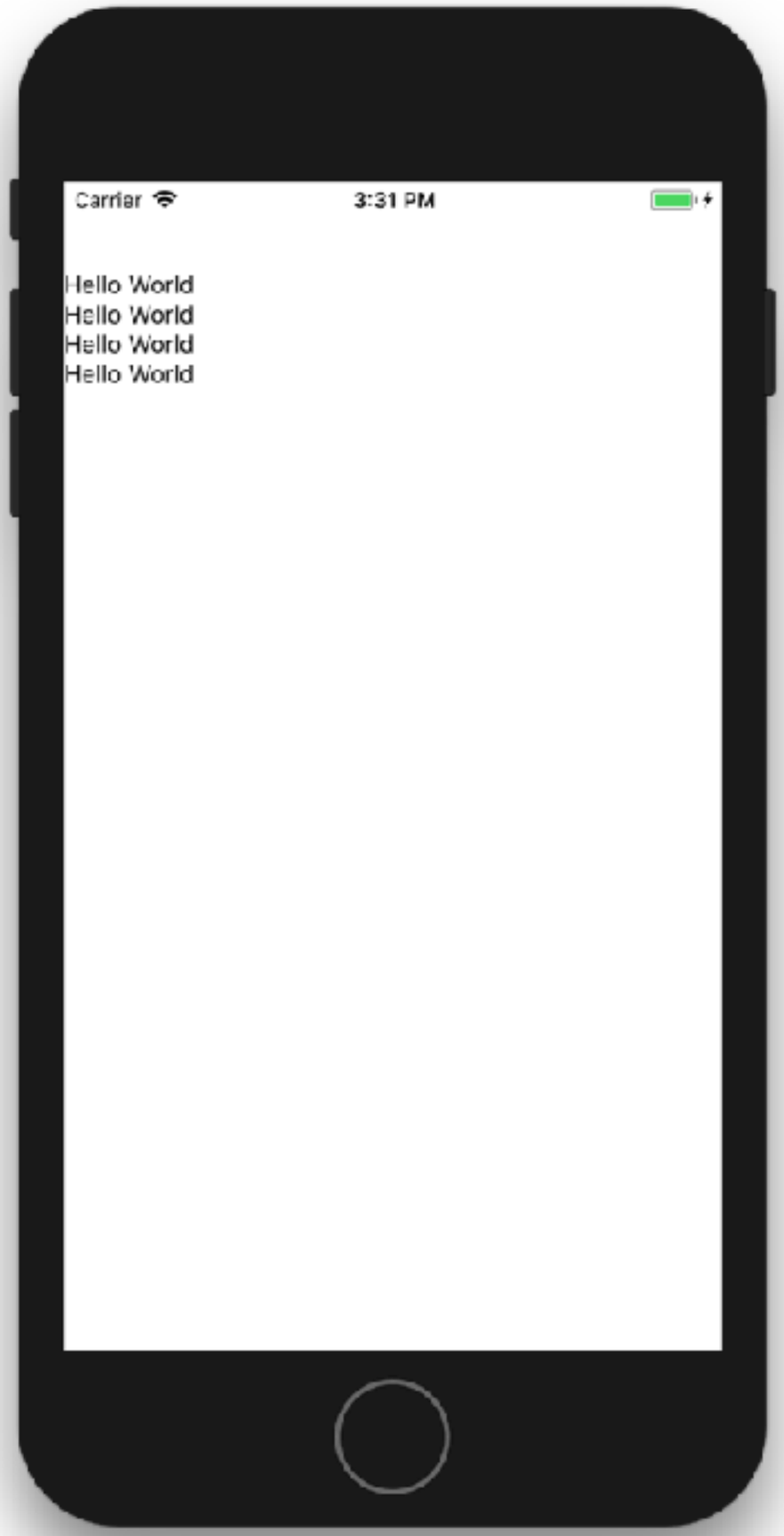
FlexBox

`flexDirection` - defines primary axis.
default is column

`row` or `column`

```
container: {
  flex: 1,
  marginTop: 50,
}
```

```
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



Flexbox

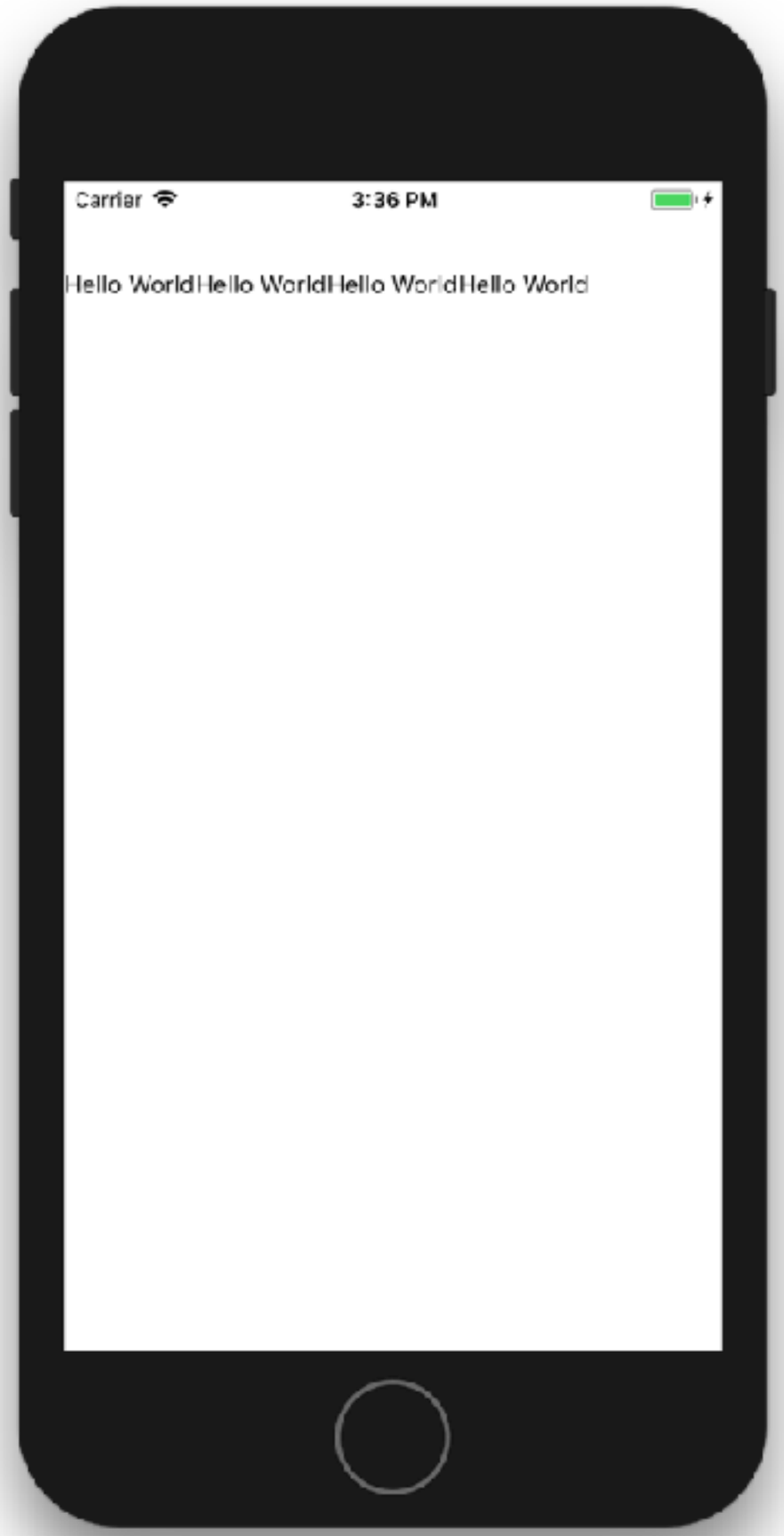
FlexBox

`flexDirection` - defines primary axis.
default is column

`row` or `column`

```
container: {
  flex: 1,
  flexDirection: 'row'
},
```

```
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



`Flexbox`

FlexBox

`justifyContent` - default is flex-start

`justifyContent` determines the distribution of children along the primary axis.

Flexbox

FlexBox

`justifyContent` - default is flex-start

`flex-start, center, flex-end, space-around, and space-between`

```
container: {
  flex: 1,
  justifyContent: 'center',
}
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```

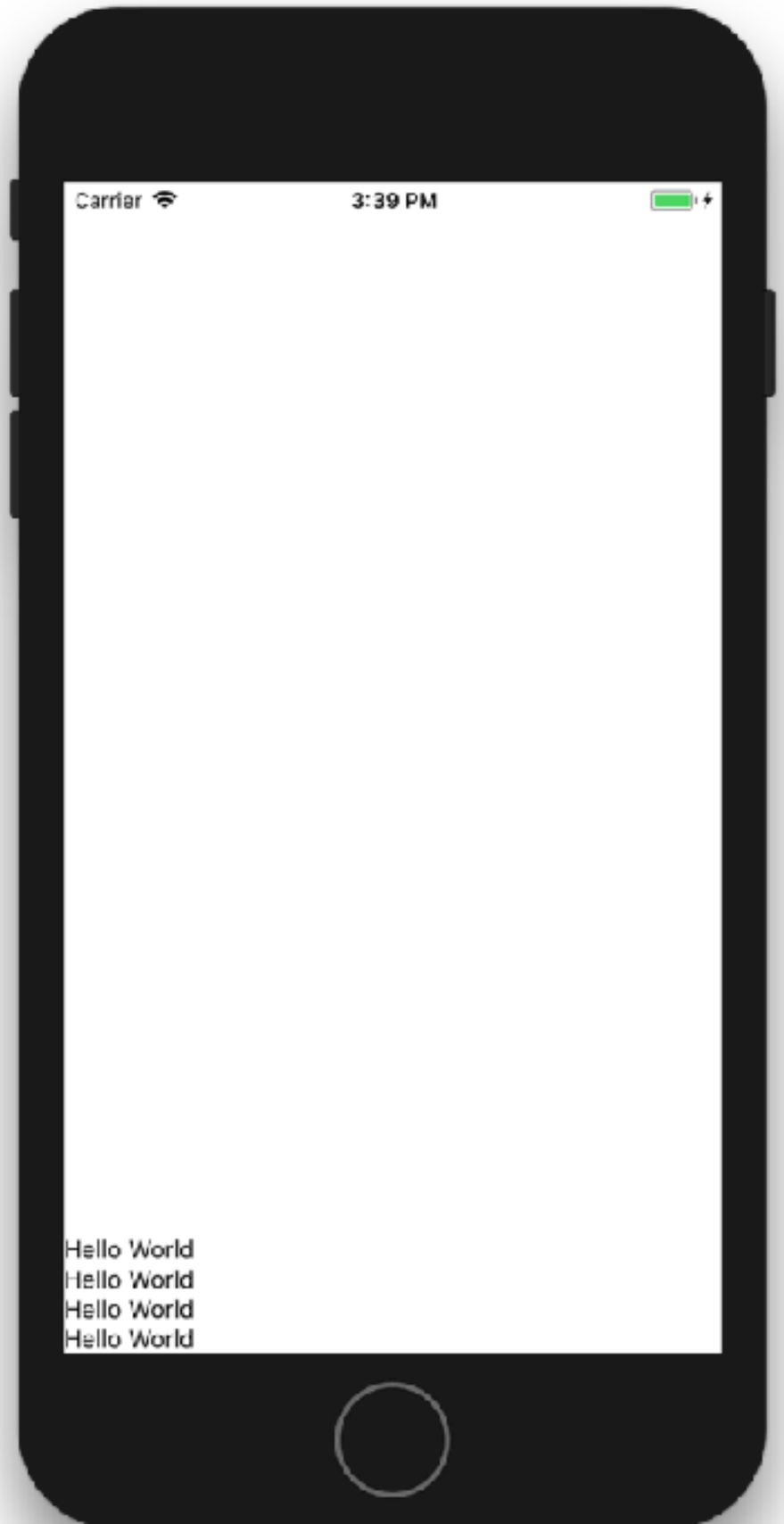
Flexbox

FlexBox

`justifyContent` - default is flex-start

`flex-start, center, flex-end, space-around, and space-between`

```
container: {
  flex: 1,
  justifyContent: 'flex-end',
}
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



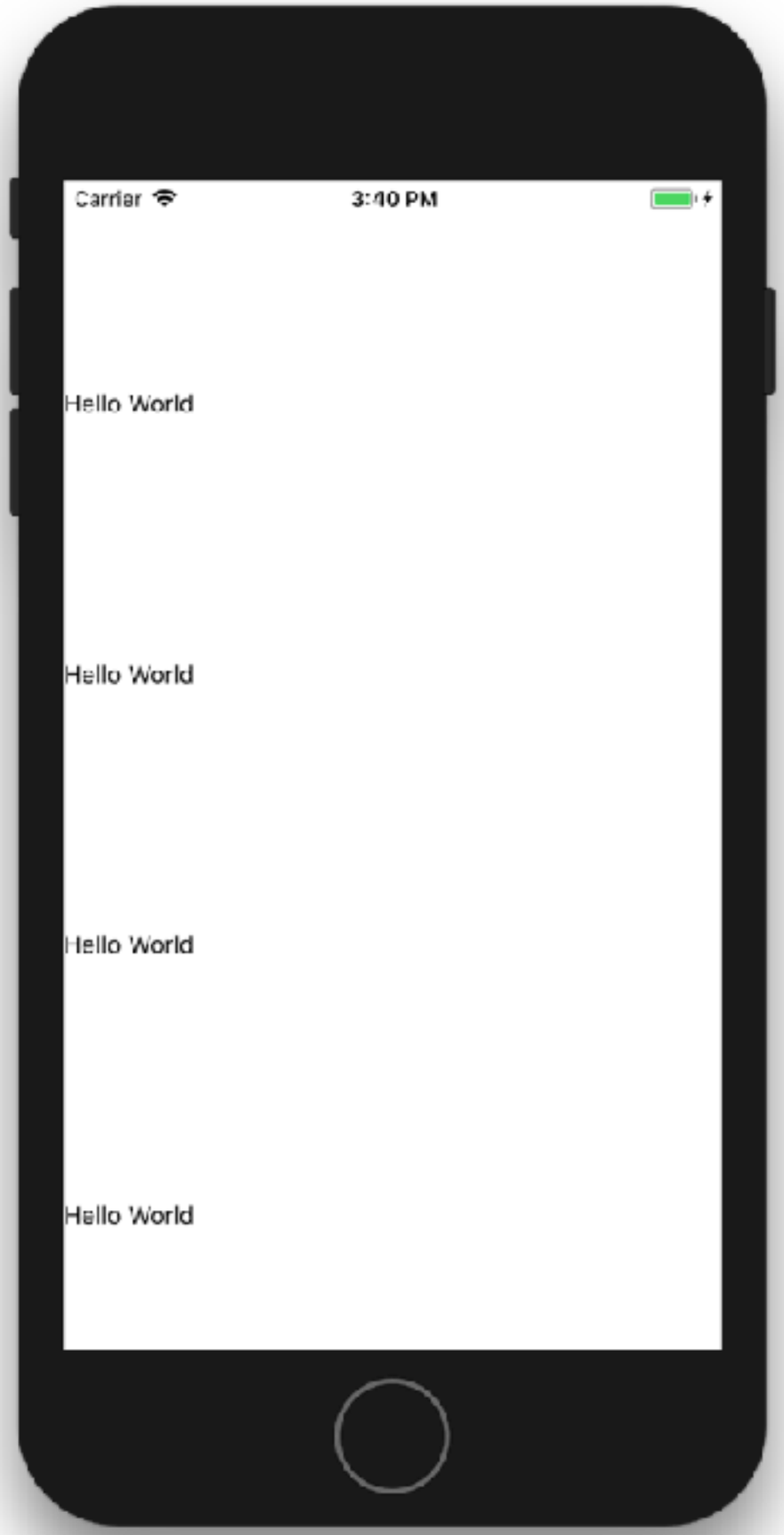
Flexbox

FlexBox

`justifyContent` - default is flex-start

`flex-start, center, flex-end, space-around, and space-between`

```
container: {
  flex: 1,
  justifyContent: 'space-around',
}
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



Flexbox

FlexBox

`alignItems` - default is stretch

`alignItems` determines the alignment of children along the secondary axis

Flexbox

FlexBox

`alignItems` - default is stretch

`flex-start, center, flex-end, and stretch.`

```
container: {
  flex: 1,
  alignItems: 'center',
},
```

```
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```

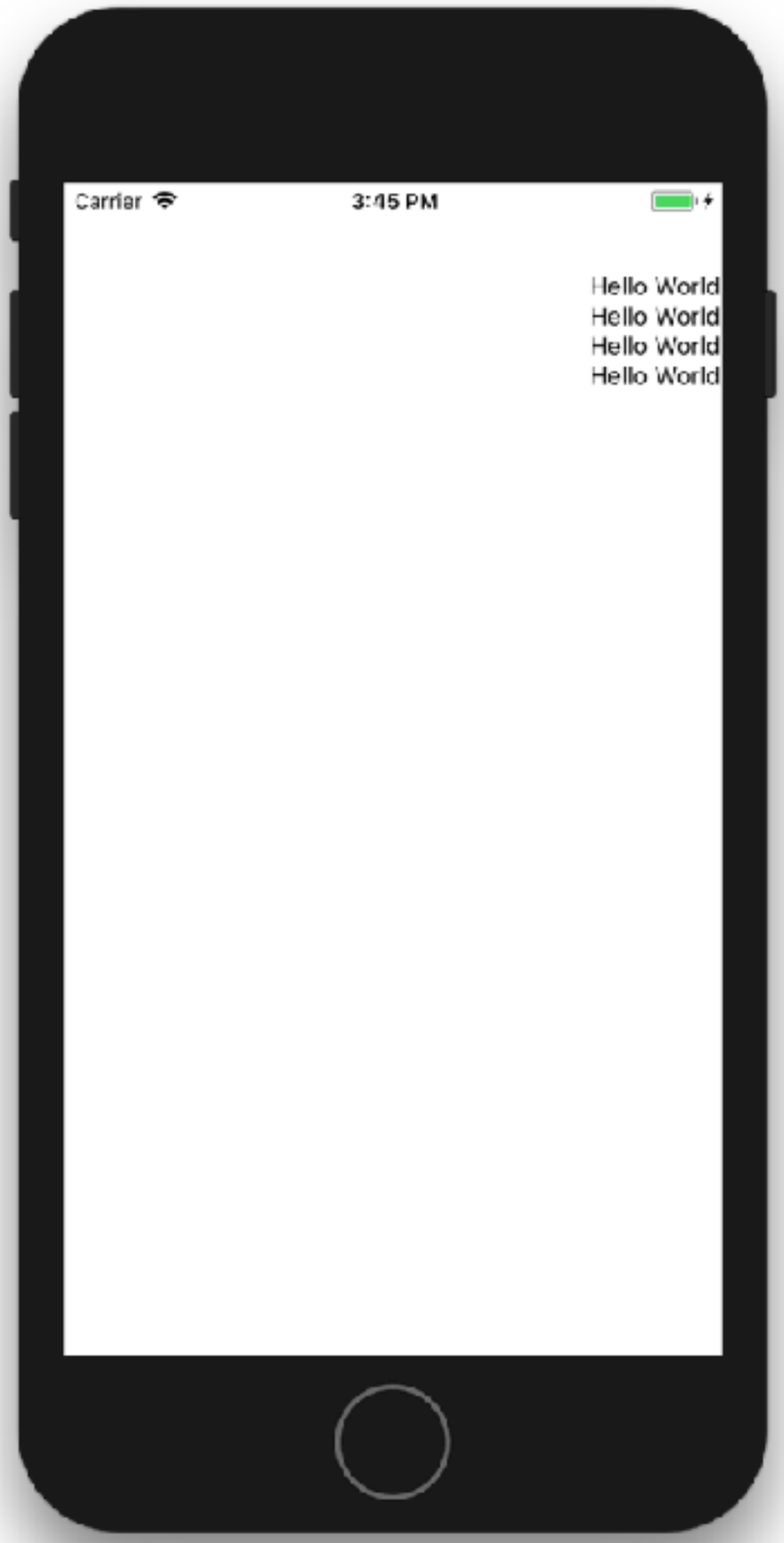
Flexbox

FlexBox

`alignItems` - default is stretch

`flex-start, center, flex-end, and stretch.`

```
container: {
  flex: 1,
  alignItems: ‘flex-end',
}
<View style={styles.container}>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
  <Text>Hello World</Text>
</View>
```



What does the flex property do?

Flexbox

Flex Dimensions:

The use of the flex property in a component's style causes the component to expand and shrink dynamically based on available space.

Normally you will use flex: 1, which tells a component to fill all available space, shared evenly amongst each other component with the same parent. The larger the flex given, the higher the ratio of space a component will take compared to its siblings.

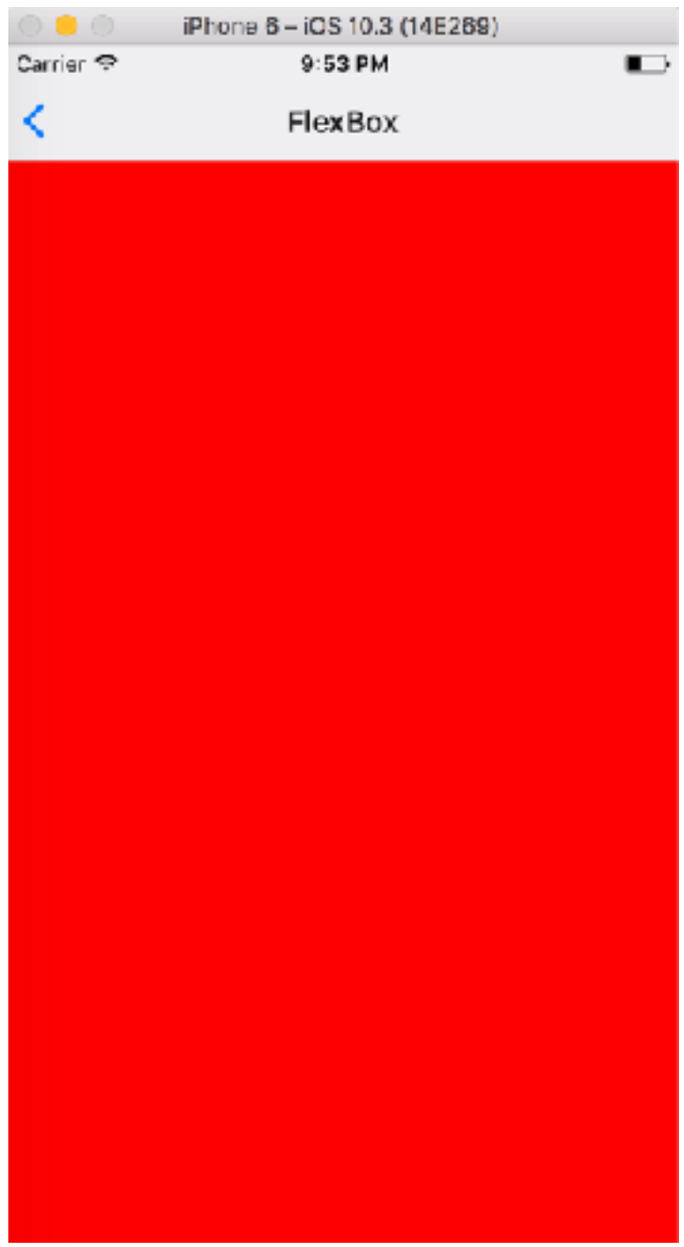
Example:
https://github.com/hgale/FlexBoxDemo/pull/1

Flexbox

FlexBox

`flex: number`

```
container: {
  flex: 1,
},
box1: {
  flex: 1,
  backgroundColor: 'red',
},
<View style={styles.container}>
  <View style={styles.box1} />
</View>
```



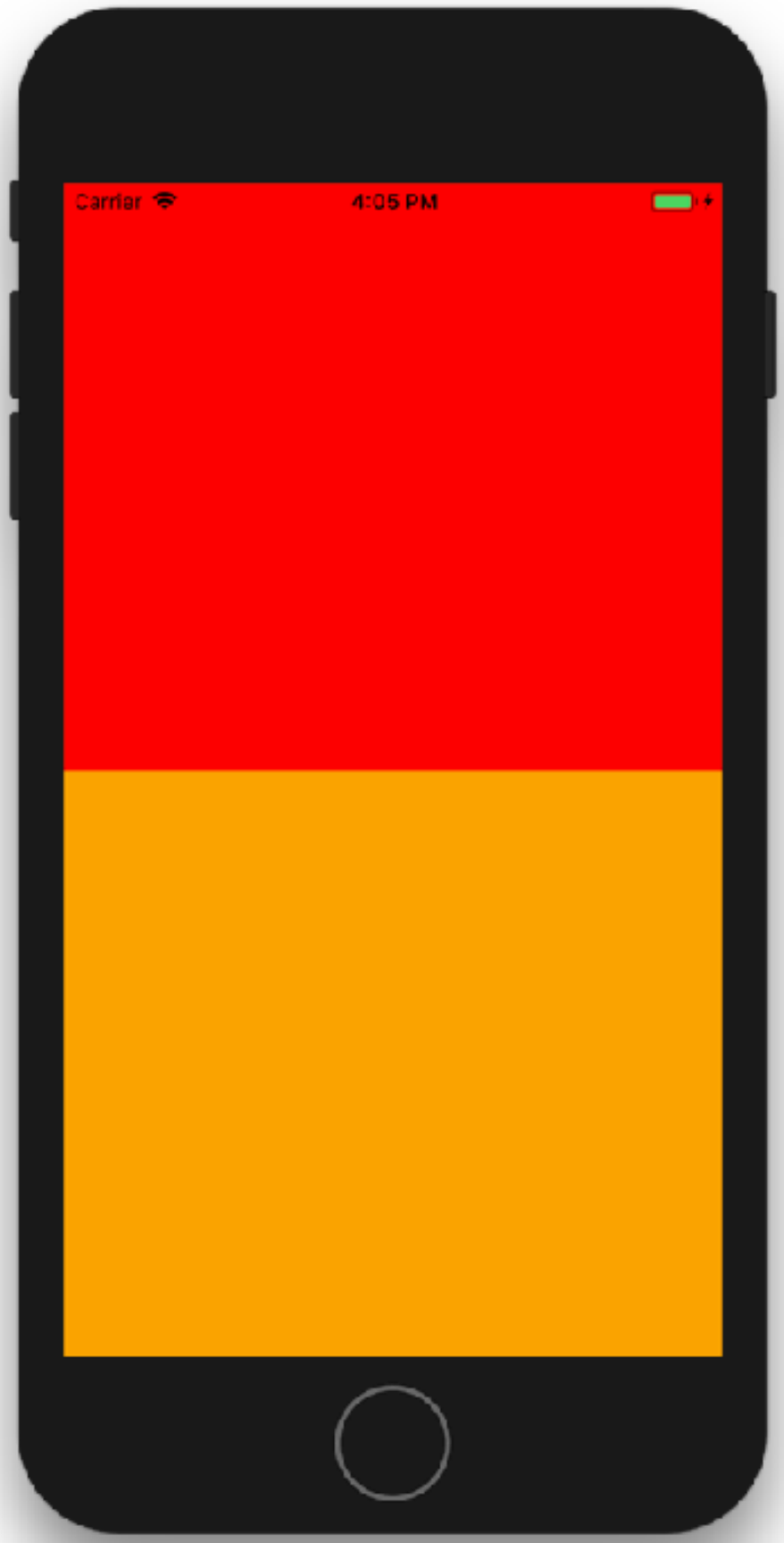
Flexbox

FlexBox

```
flex: number
```

```
container: {
// 1:1 ratio,
// give container the whole screen
  flex: 1,
},
  box1: {
    flex: 1, // 1:1 ratio
    backgroundColor: 'red',
  },
  box2: {
    flex: 1, // 1:1 ratio
    backgroundColor: 'orange',
  },
```

```
<View style={styles.container}>
  <View style={styles.box1} />
  <View style={styles.box2} />
</View>
```



Flexbox

FlexBox

flex: number

```
container: {
  flex: 1, // 1:1
},
box1: {
  flex: 3, // 3:5
  backgroundColor: 'red',
},
box2: {
  flex: 1, // 1:5
  backgroundColor: 'orange',
},
box3: {
  flex: 1, // 1:5
  backgroundColor: 'yellow',
},
```

```
<View style={styles.container}>
  <View style={styles.box1} />
  <View style={styles.box2} />
  <View style={styles.box3} />
</View>
```