Working with Styles & Layout

Bala Krishna Ragala

The plan

- Exploring Styles and using styles
  - Inline Styles, Style Objects, Stylesheet
- Exploring layout
  - Understanding Box model
  - Understanding FlexBox model
- Styling Text
- Styling Images
- Debugging Styles

Exploring Styles

- Use style prop on the UI element to apply styles
- React Native you can apply styles in following ways
  - Inline - right on the component
  - Object - as JS object with styling properties
  - Stylesheet - Stylesheet.create function

Stylesheet

- An abstraction similar to CSS StyleSheets
- Define

```
const styles = StyleSheet.create({
  container: {
    borderRadius: 4,
    borderWidth: 0.5,
    borderColor: '#d6d7da',
  },
  title: {
    fontSize: 19,
    fontWeight: 'bold',
  },
  activeTitle: {
    color: 'red',
  },
});
```

- Use

```
<View style={styles.container}>
  <Text style={[styles.title, this.props.isActive && styles.activeTitle]} />
</View>
```

What Stylesheet

- Code quality:
    - By moving styles away from the render function, you're making the code easier to understand.
    - Naming the styles is a good way to add meaning to the low level components in the render function.
- Performance:
    - Making a stylesheet from a style object makes it possible to refer to it by ID instead of creating a new style object every time.
    - It also allows to send the style only once through the bridge. All subsequent uses are going to refer an id (not implemented yet).

Inline Styles

```
<Text style={{fontSize:18, fontWeight:'bold'}}>
        Welcome to RN, STYLING!
</Text>
```

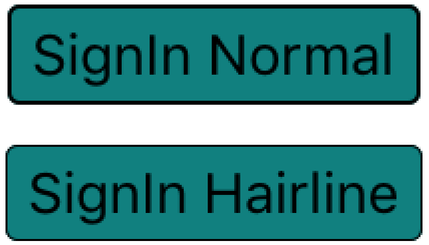
Styles as Objects

```
const textStyles = {
    color: '#fff',
    fontSize: 22,
    fontWeight: '900',
    paddingLeft: 20,
    paddingRight: 20,
    paddingTop: 5,
    paddingBottom: 5
};

<Text style={textStyles}>This is header</Text>
```

Hairline Width

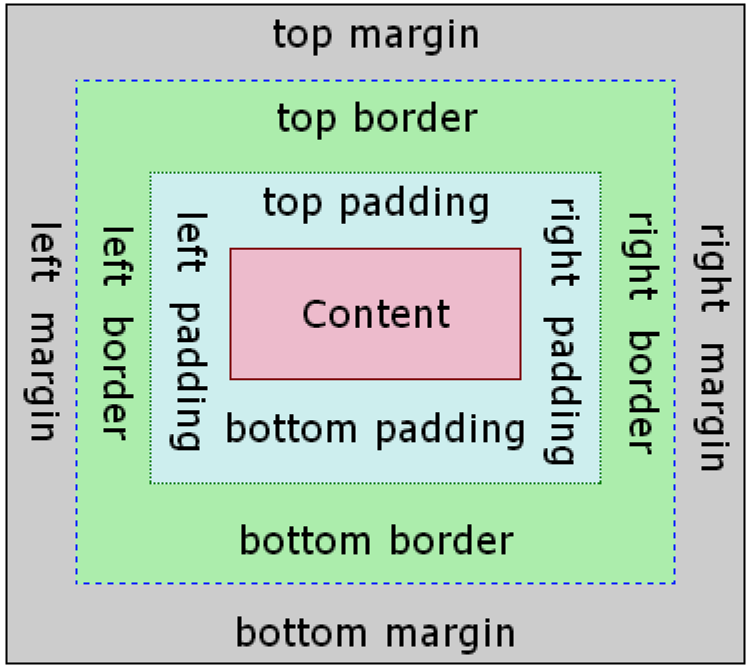
- A constant
- will always be a round number of pixels (so a line defined by it can look crisp) and will try to match the standard width of a thin line on the underlying platform.
- However, you should not rely on it being a constant size, because on different platforms and screen densities its value may be calculated differently



```
hairlineBorder: {
	borderWidth: StyleSheet.hairlineWidth

}
```

Exploring Layout - Box Model



Flexbox – your box

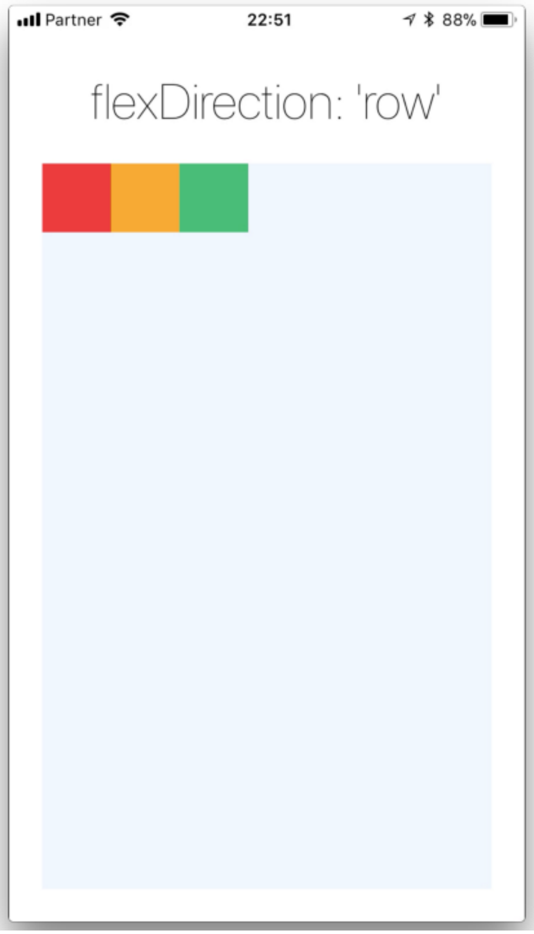
- ***Used to layout the elements in React Native***
- Influenced by web but a downsized to suit mobile dev
- All components on RN and flex containers and are positioned relatively
- In RN display style property take value as flex ONLY and this is implicit
- A relationship between the container and its immediate children
- Items in container are either **horizontally** or **vertically** stacked
- flexDirection property should be used to alter the direction if needed
- Default direction is set to Column in React Native

Dive into axis

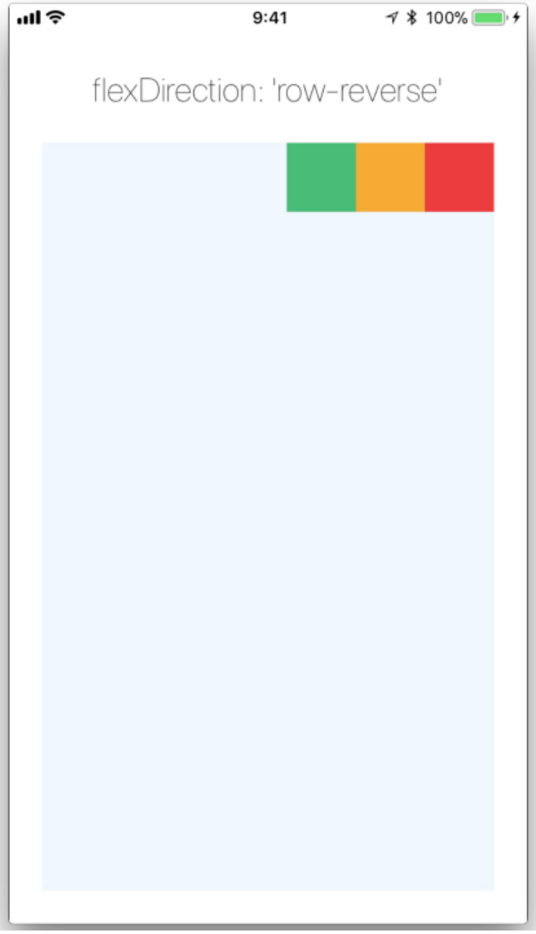
- X and Y axis influence the alignment of elements in flex
- Referred as **mainaxis** and **crossaxis**
- Mainaxis is set by **flexDirection**
- Alignment of items on mainaxis are influenced by **justifyContent**
- Otheraxis referred to **crossaxis**
- Alignment of items on crossaxis are influenced by **alignItems**

flexDirection	Main Axis	Cross Axis
column	Y	X
row	X	Y

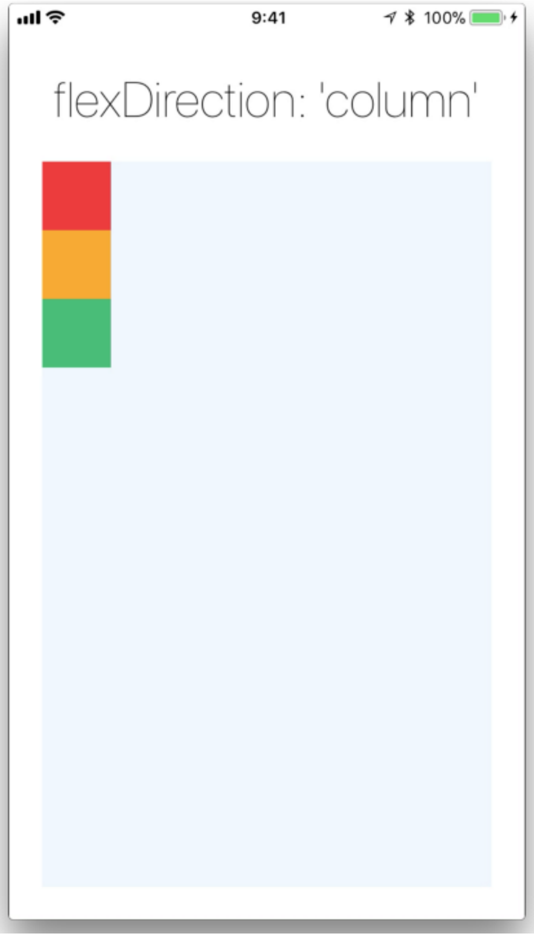
flexDirection



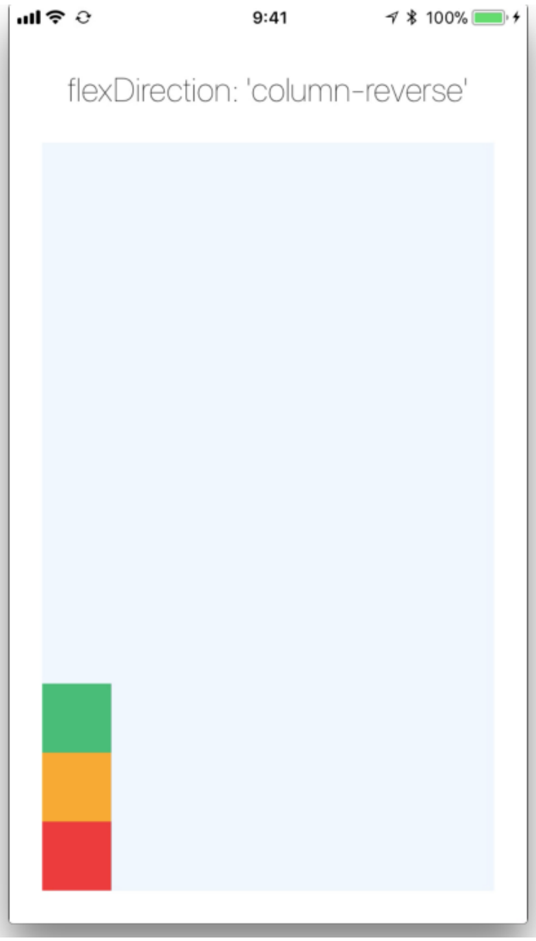
flexDirection: 'row'



flexDirection: 'row-reverse'



flexDirection: 'column'



flexDirection: 'column-reverse'

flex

```
<View style={[{flex: 1}, styles.elementsContainer]}>
    <View style={{flex: 1, backgroundColor: '#EE2C38'}} />
    <View style={{flex: 2, backgroundColor: '#FAA030'}} />
    <View style={{flex: 3, backgroundColor: '#32B76C'}} />
</View>
```

The red view got flex:1 ,

the yellow view got flex:2

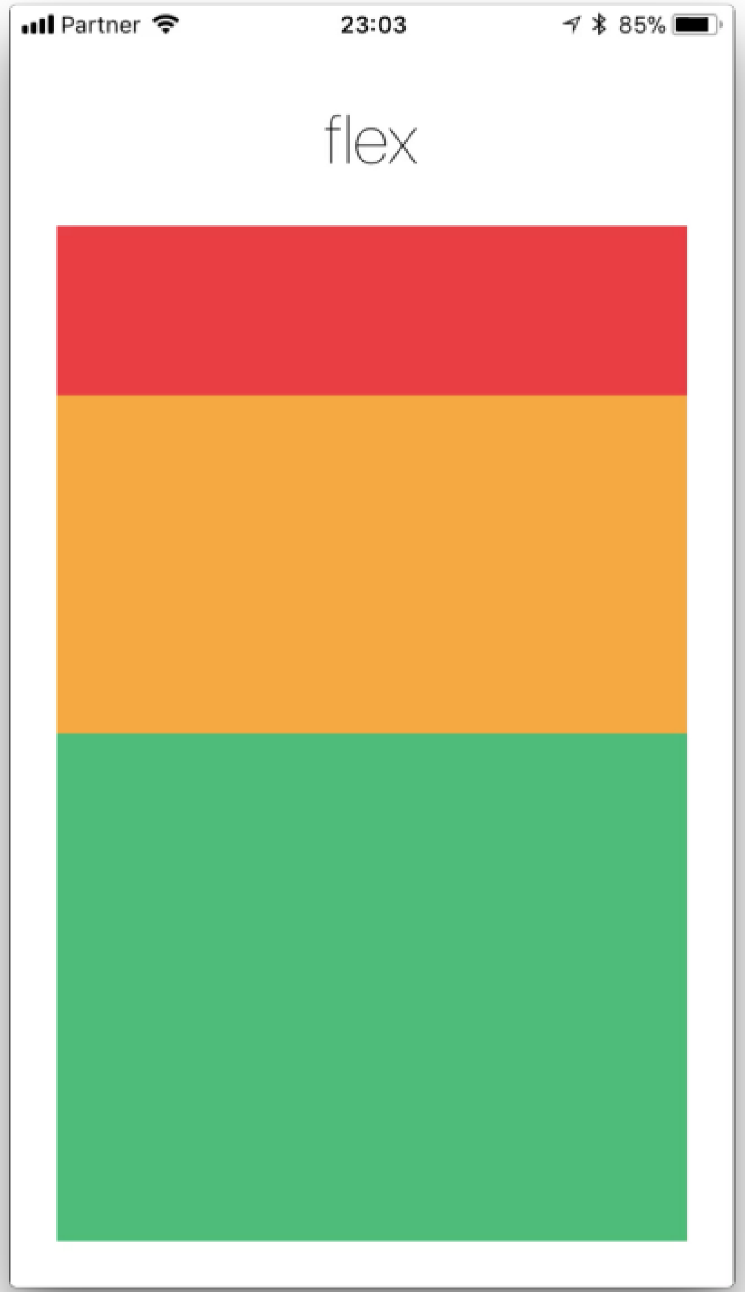
and the green view got flex:3 .

1+2+3=6 which means that

red view will get 1/6 of the space,

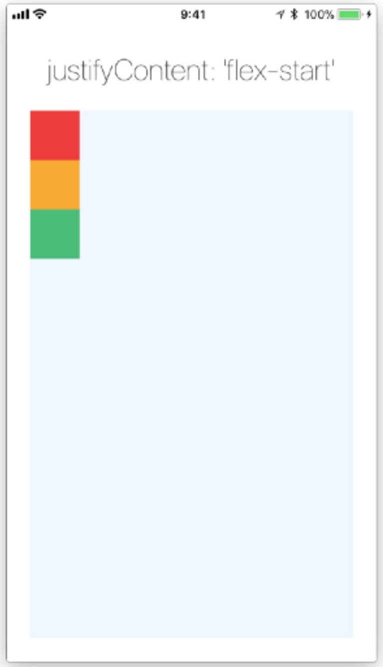
the yellow 2/6 of the space and

the green 3/6 of the space

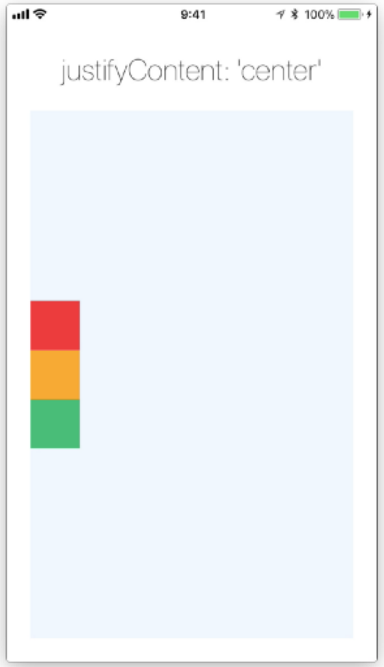


justifyContent

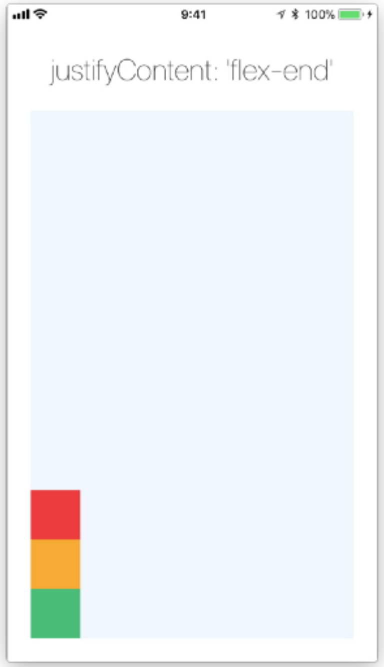
- Aligns content over main axis
- Possible values
  - flex-start
  - center
  - flex-end
  - space-between
  - Space-around



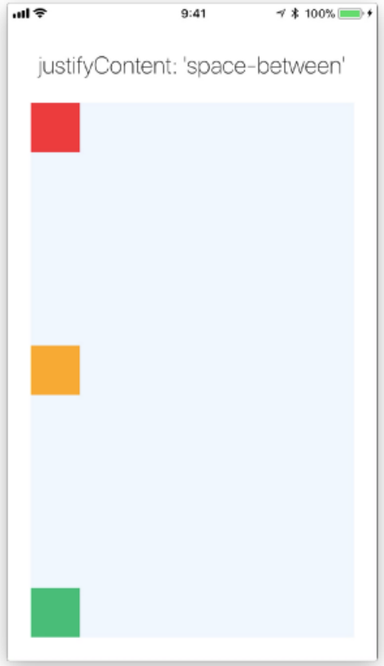
justifyContent: 'flex-start'



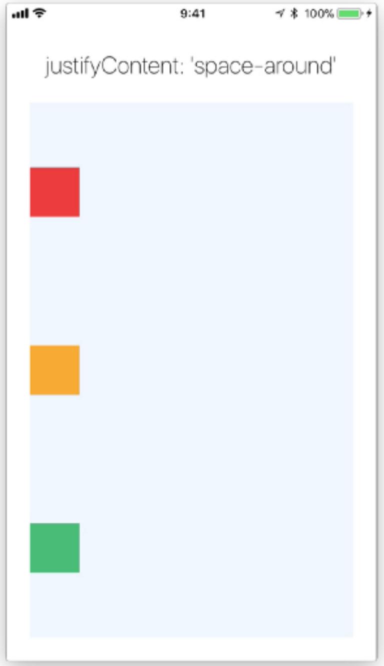
justifyContent: 'center'



justifyContent: 'flex-end'



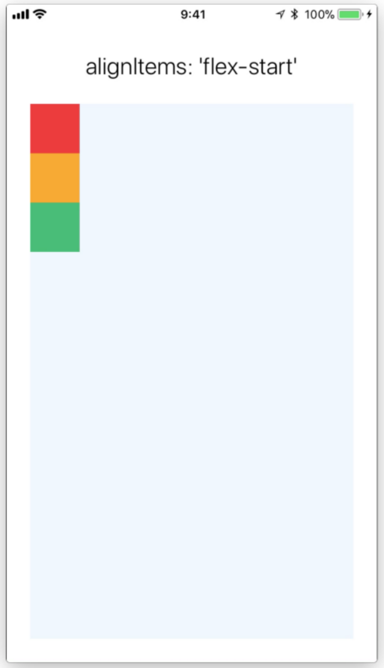
justifyContent: 'space-between'

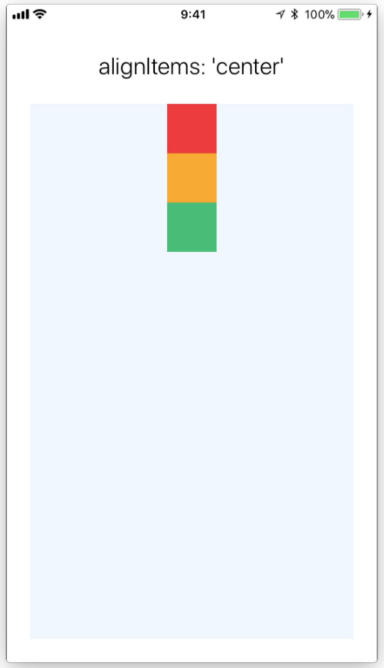


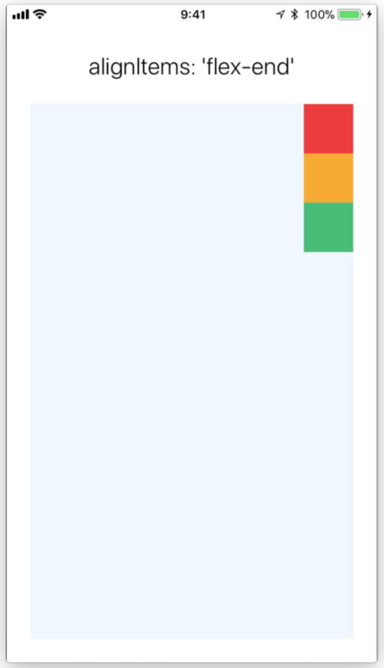
justifyContent: 'space-around'

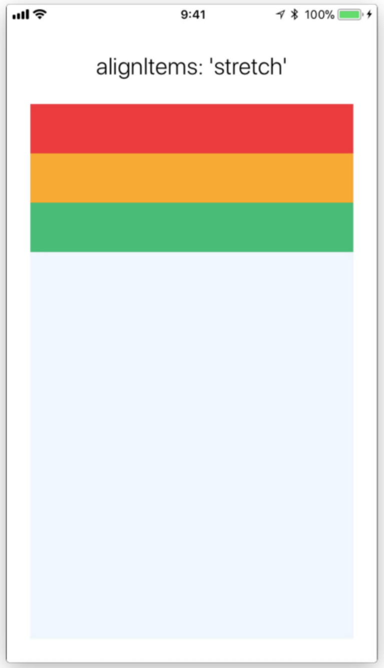
alignItems

- Aligns content over cross axis
- Possible values
  - flex-start
  - center
  - flex-end
  - Stretch (makes flex item to take available width if no width specified)



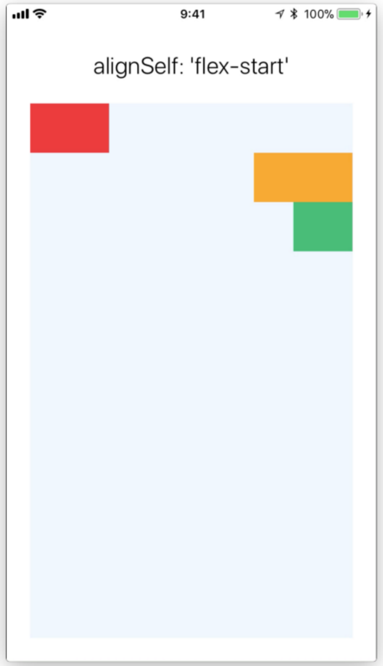


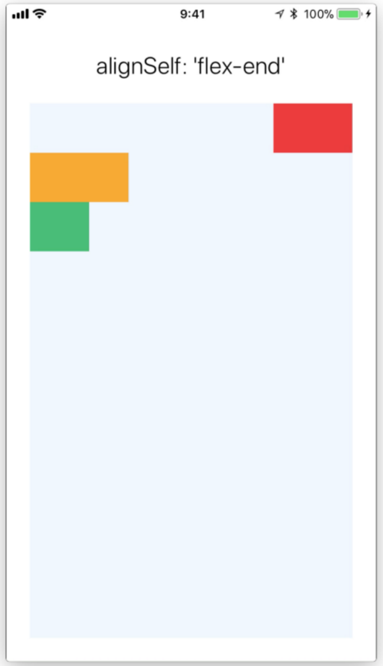


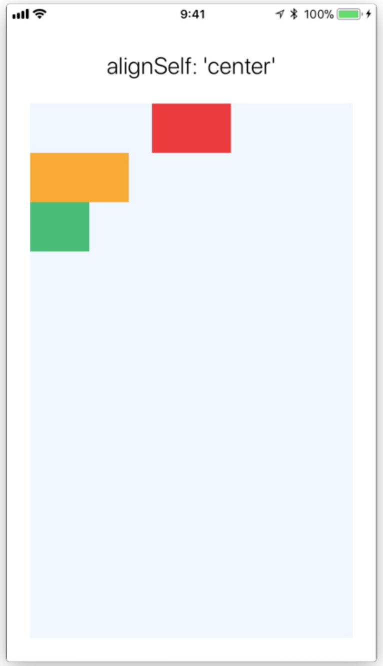


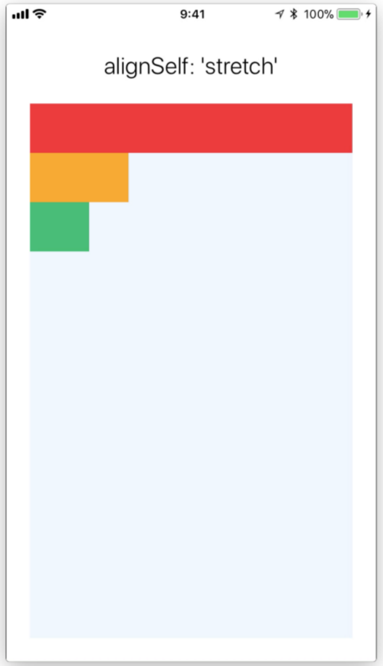
alignSelf

- Align an item along the **cross axis overwriting** his **parent alignItems** property
- Possible values
  - flex-start
  - center
  - flex-end
  - Stretch (makes flex item to take available width if no width specified)



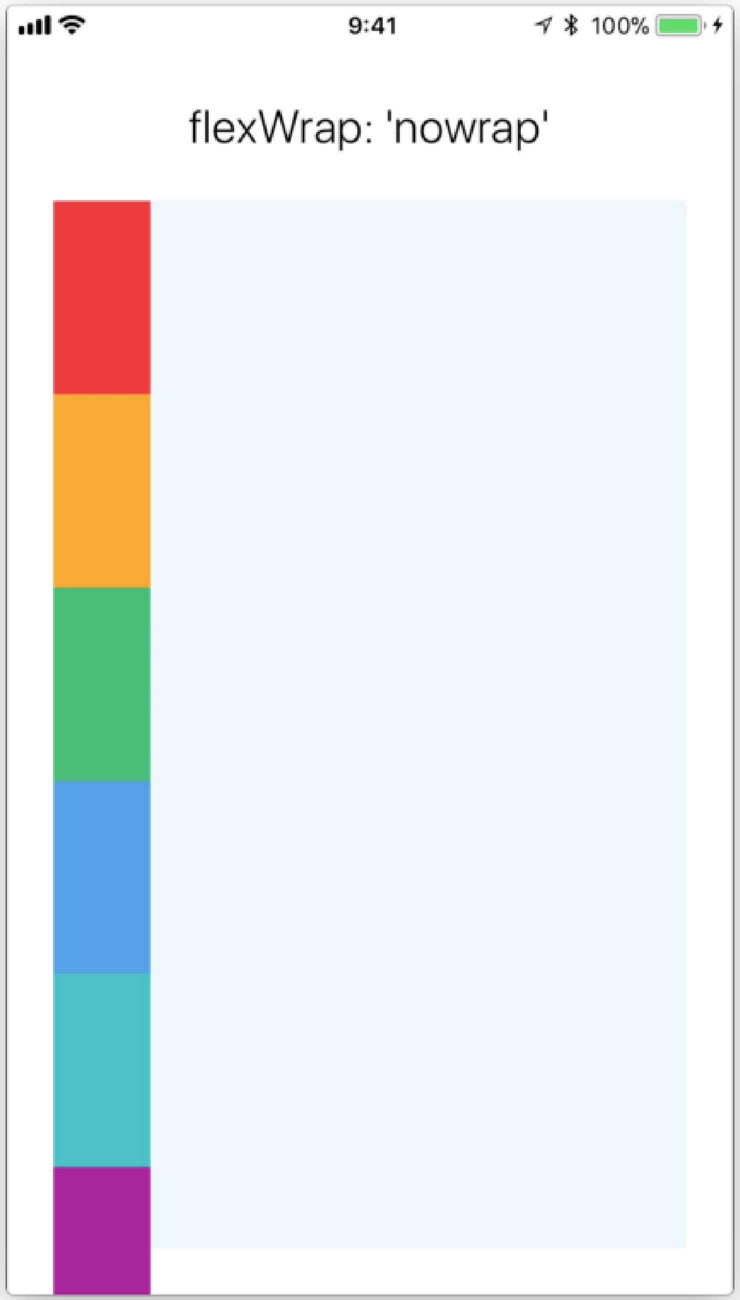


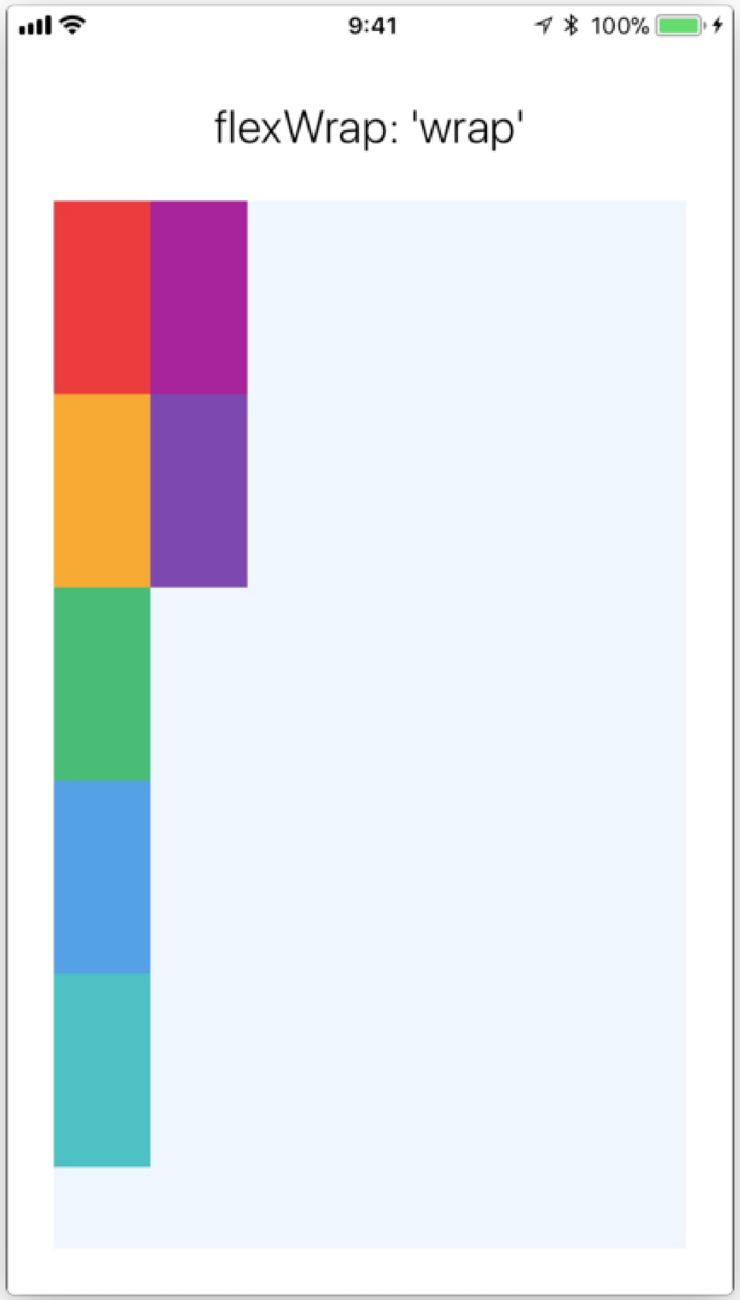




flexWrap

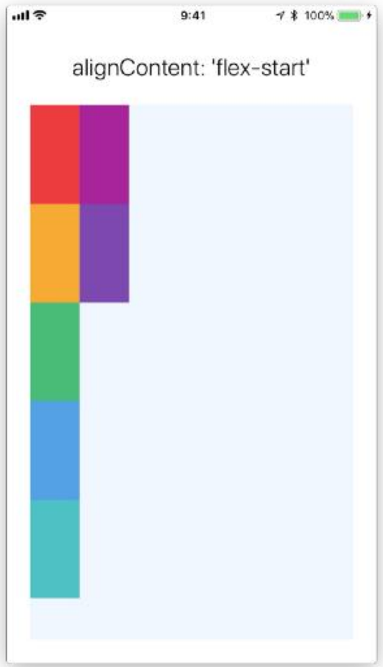
- Controls whether flex items are forced on a single line or can be wrapped on multiple lines.
- Possible values
  - wrap
  - nowrap

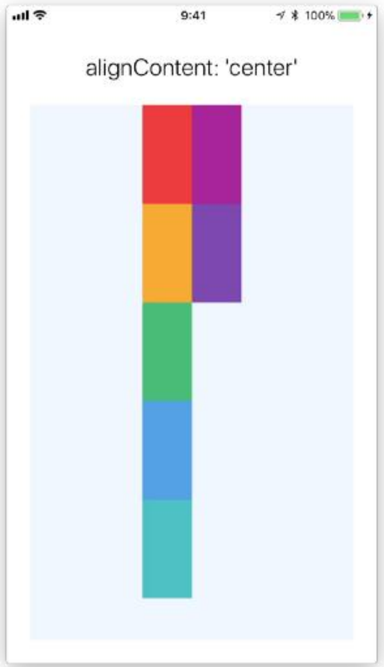


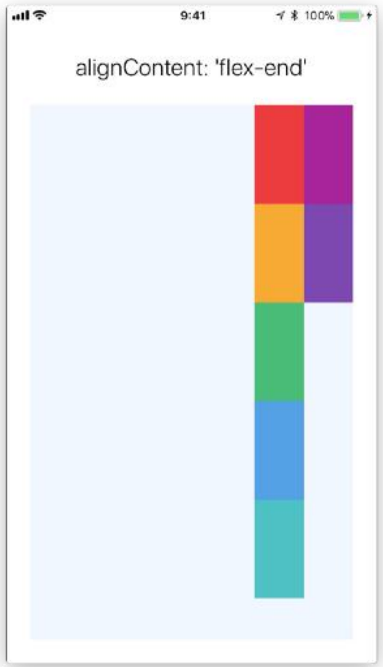


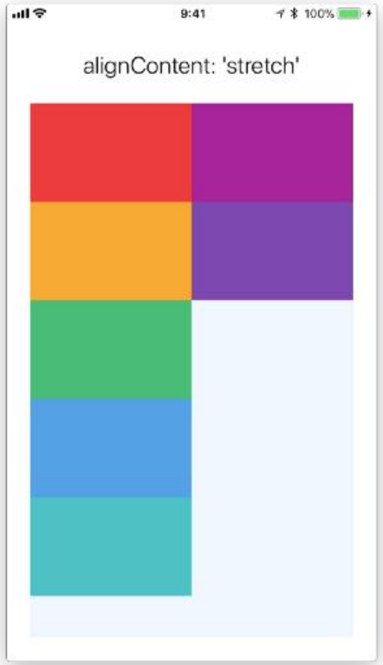
alignContent

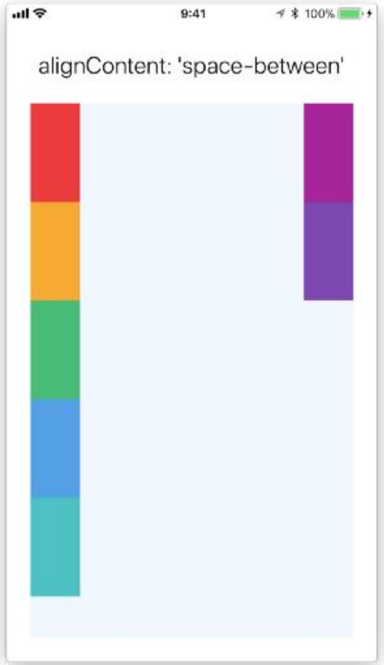
- So if you went with flexWrap:'wrap' you have multiple lines of items, this property will help you align the lines on the cross-axis.
- Possible values
  - flex-start
  - center
  - flex-end
  - stretch
  - space-between
  - Space-around

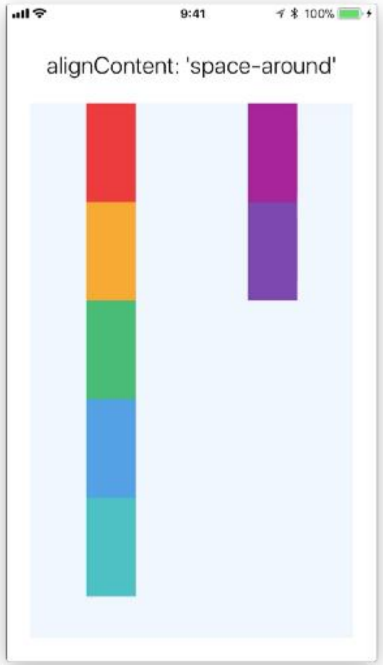






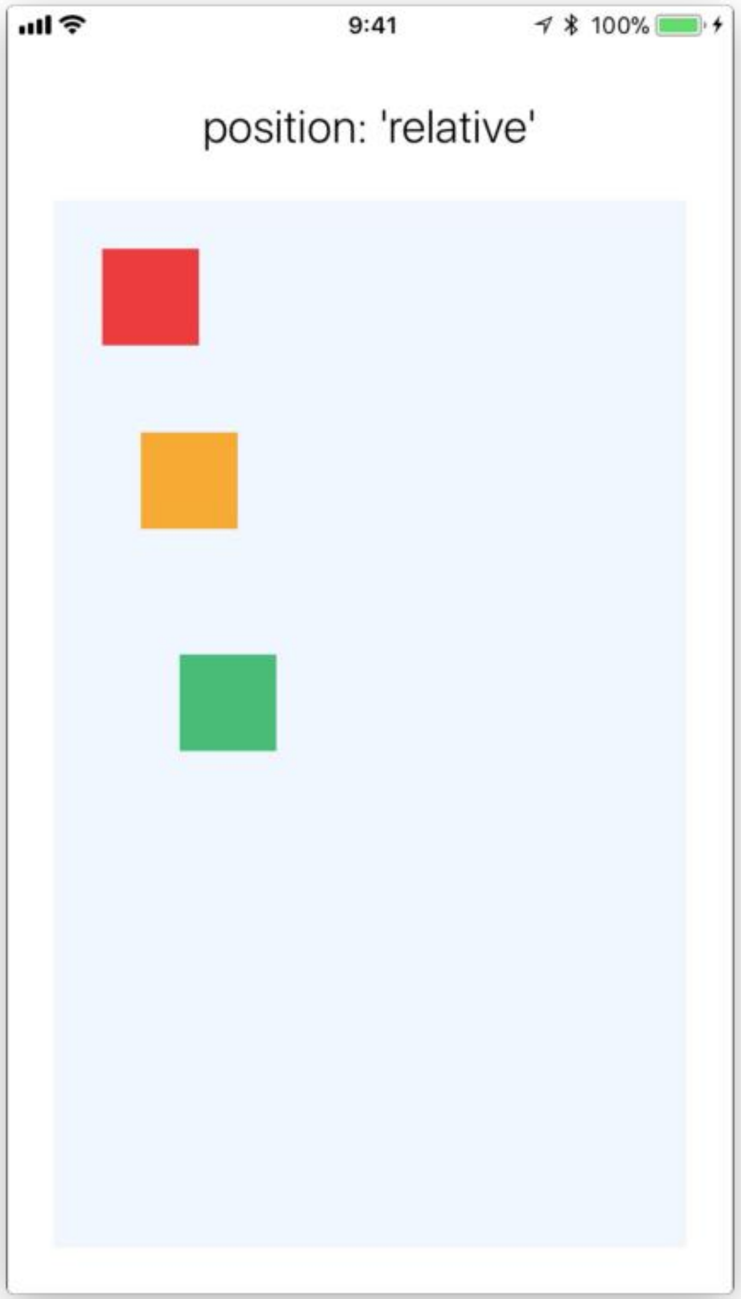


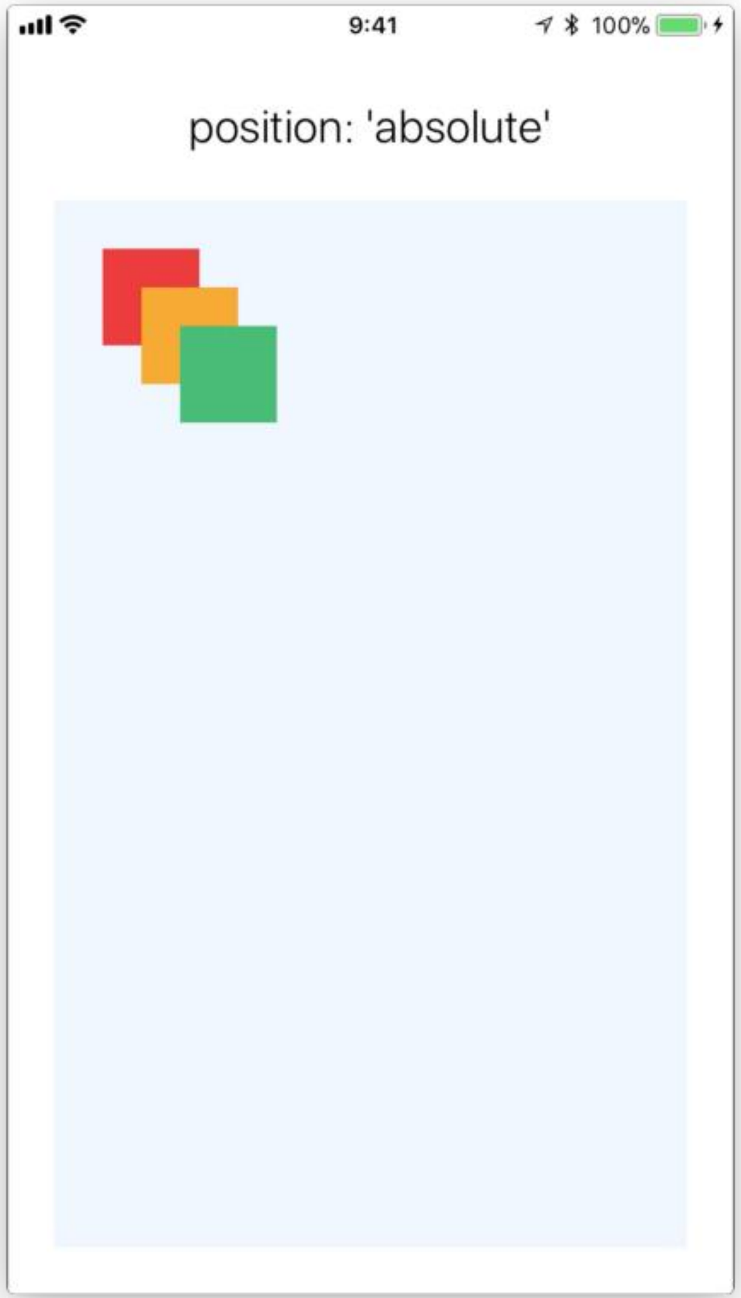




position

- The position property specifies how an element is positioned in a view
- Possible Values
  - relative
  - absolute



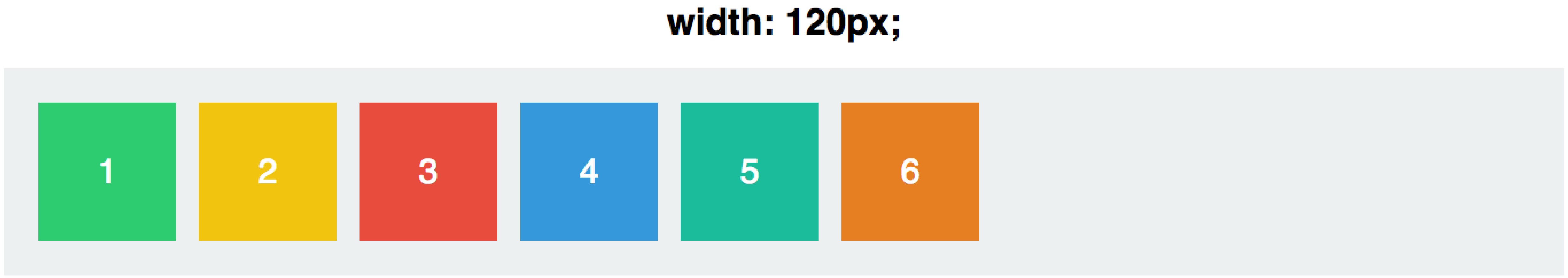


Shrink and grow flexbox (advanced)

- 3 properties - flexGrow, flexShrink, flexBasis
- flexBasis - used to set basis for size calculation for **flexGrow** and **flexShrink**
    - Flex-basis affects an element's size *across the main axis.*
    - Possible values column / row based on the flexDirection
- flexGrow - Grow in size as per flexBasis
- flexShrink - Shrink in size as per flexBasis

flowGrow

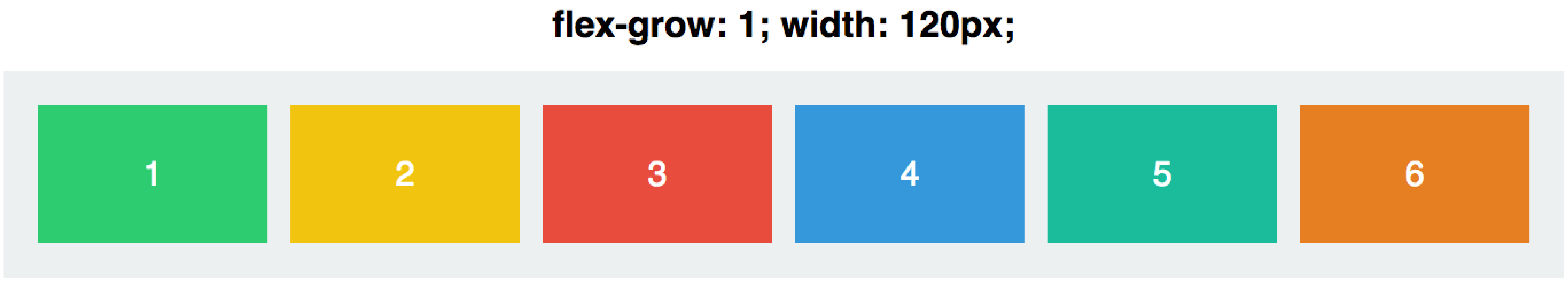
- All squares to the same width, 120px



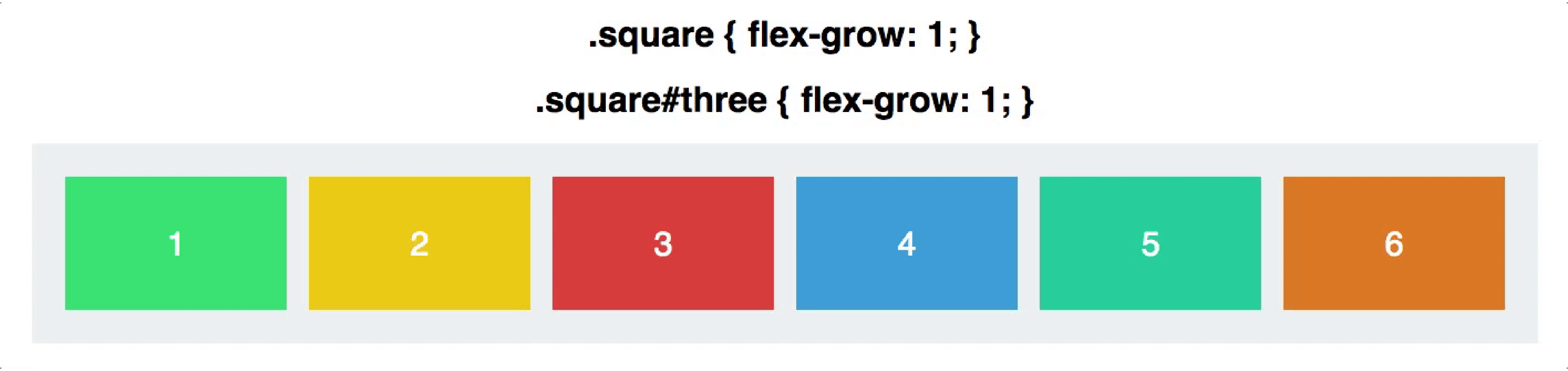
https://medium.freecodecamp.org/even-more-about-how-flexbox-works-explained-in-big-colorful-animated-gifs-a5a74812b053

flowGrow

- when it comes to the property called **flexGrow**, the default is **0**. That means the squares are not allowed to grow to take up the space in the container
- Incrementing flex-grow to **1** for every square, result is below
- **flexGrow** *value overrides the width*
- **Flex-grow is all about proportions**.



flowGrow - live

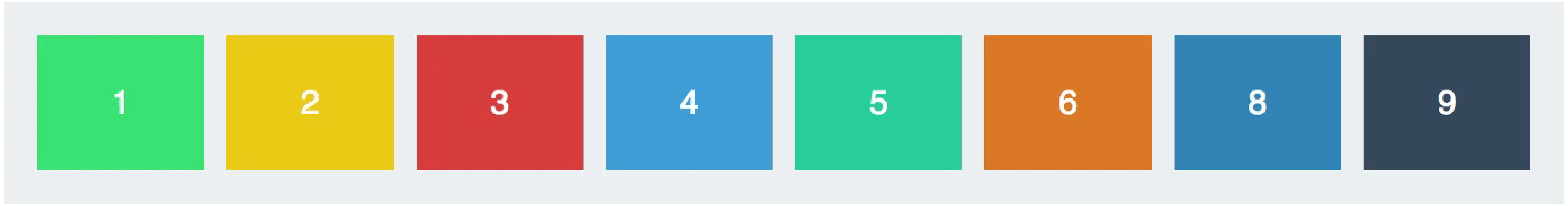


```
.square { flex-grow: 1; }
.square#three { flex-grow: 1; }
```

flexShrink

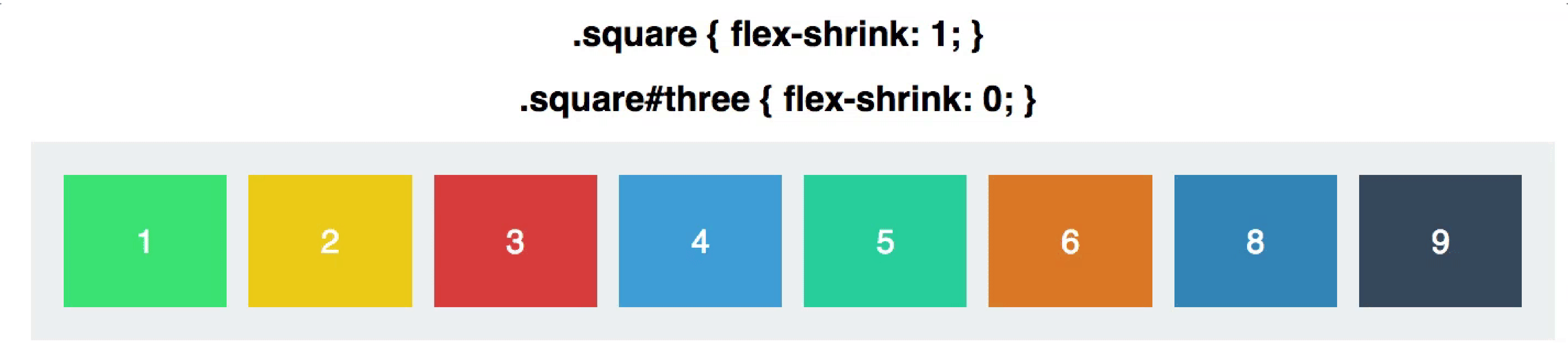
- flexShrink is the opposite of flexGrow, determining how much a square is allowed to shrink
- Its main use is to specify which items you want to shrink, and which items you don't. By default, every square has a flex-shrink of 1 — which means it will shrink as the box contracts.

.square { flex-shrink: 1; }



flexShrink

- Now let’s set the flex-shrink of Square #3 to 0. It’s forbidden to shrink, so it while it grows to fit the container, it refuses to dip below its set 120px width.

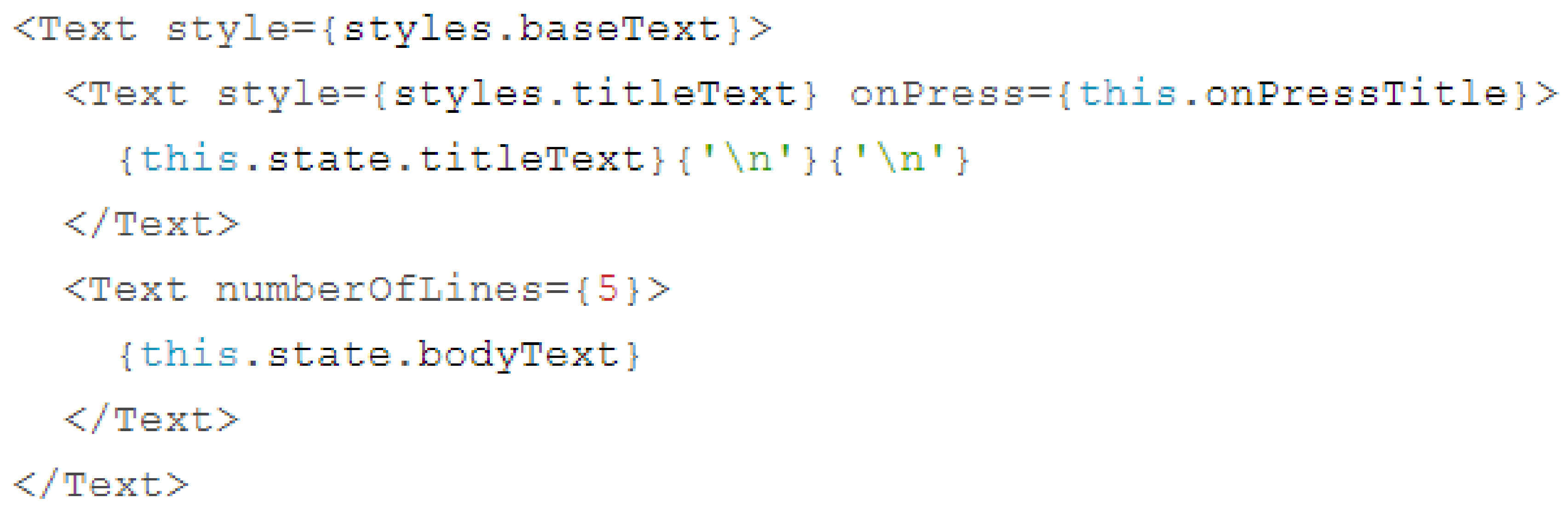


```
.square { flex-shrink: 1; }

.square#three { flex-shrink: 0; }
```

Text (Native Component)

- A React component for displaying text.
- Supports nesting, styling, and touch handling



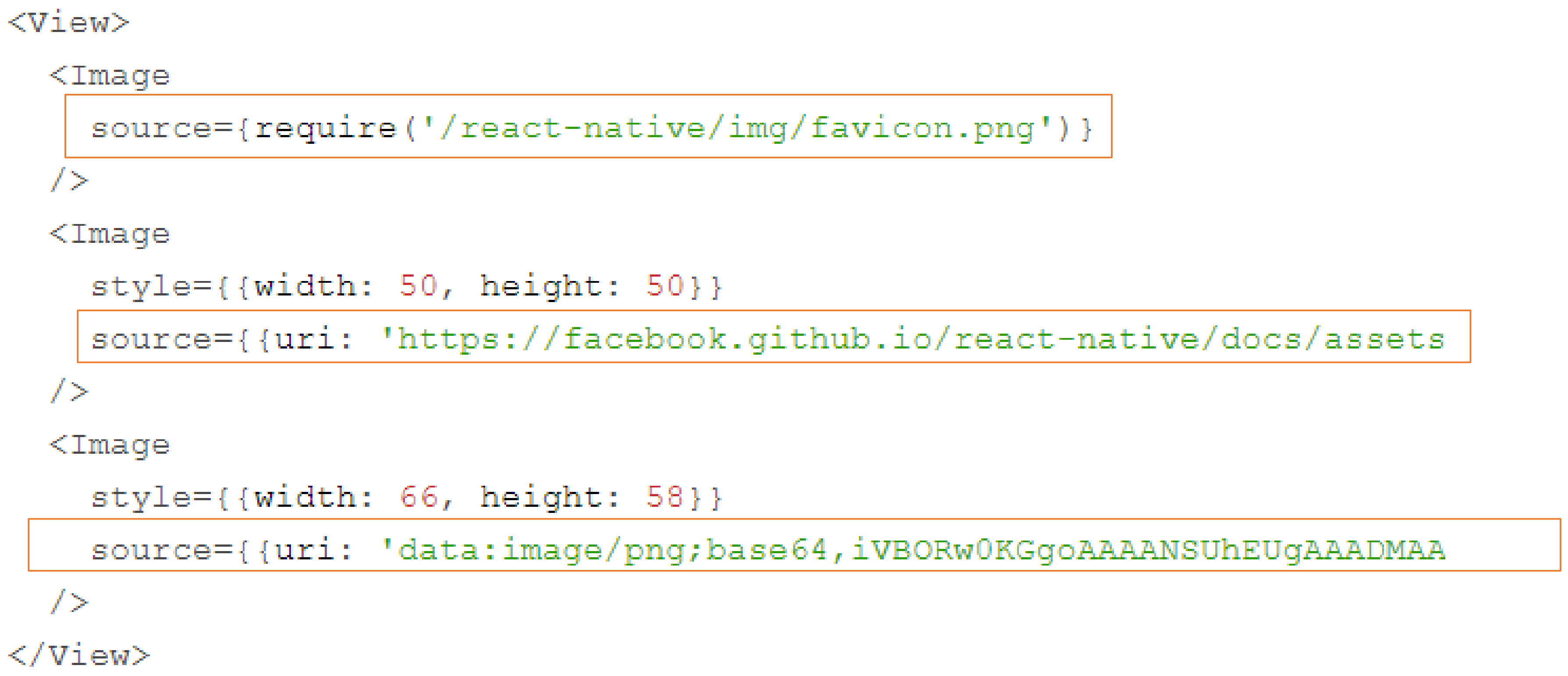
```
<Text style={styles.baseText}>
  <Text style={styles.titleText} onPress={this.onPressTitle}>
    {this.state.titleText}{'\n'}{'\n'}
  </Text>
  <Text numberOfLines={5}>
    {this.state.bodyText}
  </Text>
</Text>
```

Styling Text

- Most of the rules around React Native styling are equally applicable to text
- Some of the props
  - fontSize
  - fontWeight
  - fontStyle
  - textAlign
  - fontFamily
  - lineHeight
  - fontVariant
  - letterSpacing
  - writingDirection

Image (Native Component)

- A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.

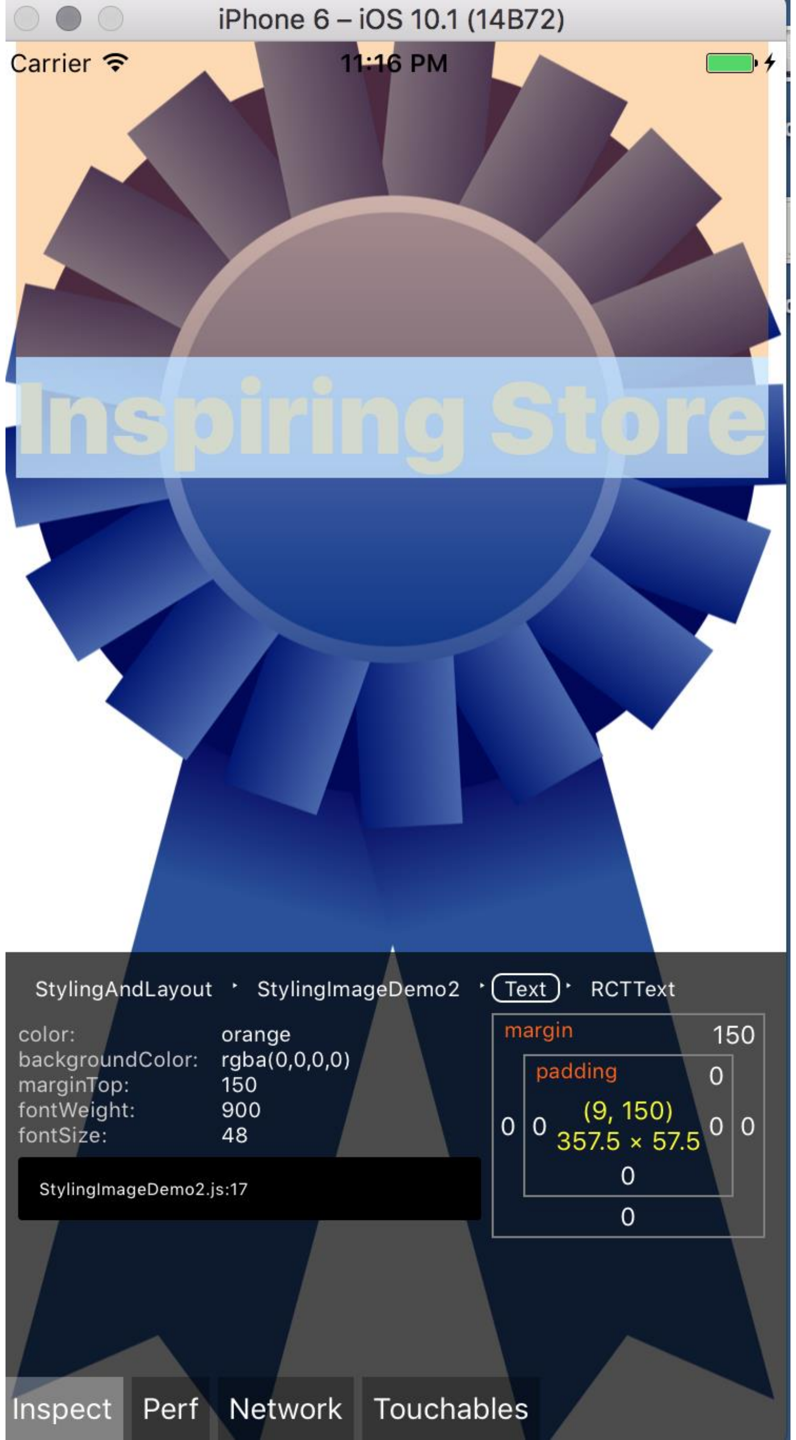


```
<View>
  <Image
    source={require('/react-native/img/favicon.png')}
  />
  <Image
    style={{width: 50, height: 50}}
    source={{uri: 'https://facebook.github.io/react-native/docs/assets
  />
  <Image
    style={{width: 66, height: 58}}
    source={{uri: 'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAADMAA
  />
</View>
```

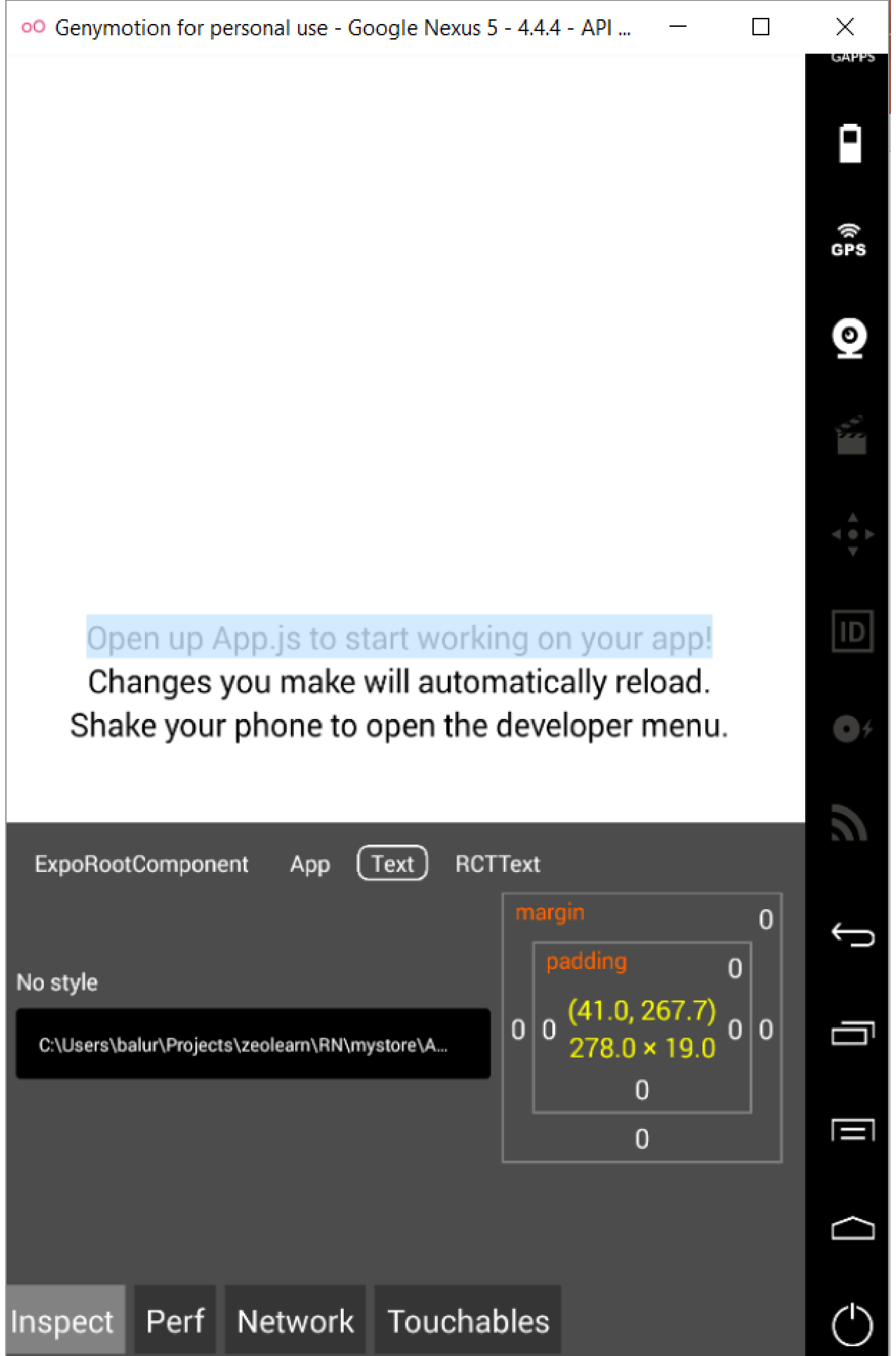
Styling Images

- Use Image component to render images
- Use width and height props to manage imageview size
- Without width and height images take default 0 X 0 view size
- Use resizeMode to control the view of the images
- resizeMode
  - contain
  - stretch
  - center
  - cover
  - repeat

Debugging Styles



```
color:            orange
backgroundColor:  rgba(0,0,0,0)
marginTop:        150
fontWeight:       900
fontSize:         48
StylingImageDemo2.js:17
```



Open up App.js to start working on your app!
Changes you make will automatically reload.
Shake your phone to open the developer menu.

No style

```
C:\Users\balur\Projects\zeolearn\RN\mystore\A...
```

Platform Specific Code

- Platform module
  - import {Platform} from 'react-native';

```
const styles = StyleSheet.create({
  height: Platform.OS === 'ios' ? 200 : 100,
});
```

1

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    ...Platform.select({
      ios: {
        backgroundColor: 'red',
      },
      android: {
        backgroundColor: 'blue',
      },
    }),
  },
});
```

2

```
const Component = Platform.select({
  ios: () => require('ComponentIOS'),
  android: () => require('ComponentAndroid'),
})();

<Component />;
```

3

Platform-specific extensions

- When your platform-specific code is more complex, you should consider splitting the code out into separate files.
- React Native will detect when a file has a .ios. or .android. extension and load the relevant platform file when required from other components
- For example
  - BigButton.**ios**.js
  - BigButton.**android**.js
- Import {BigButton} from "./BigButton"
  - React Native will automatically pick up the right file based on the running platform.

Question Time

- When flexDirection is row what is main axis?
- Which property aligns content on main axis?
- Name 3 different styles of styling react native UI?
- Name resize modes of images?
- What is hairline width?
- Normal, italic, bold – which style property accepts these values?
- What is default width and height of image if not specified?
- Name 2 axis of flexbox
- In box model – padding vs margin?
- Name the module which helps to read info about current running platform?