**Flexbox Guide (Continued)**

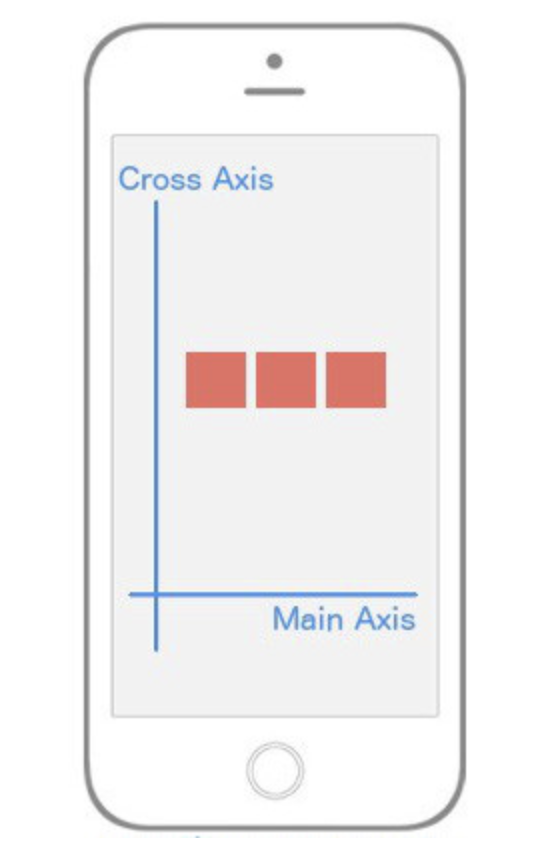
SEND FEEDBACK

There content in this section is broken down into the following sections:

* [**Centering Content**](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61#centering-content)
* [**The Flex Property**](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61#the-flex-property)
* [**Aligning Individual Items**](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61#aligning-individual-flex-items)

**Centering Content**

Let's start off with a view like this:

**[[](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)**

***[Centering content along both the Main Axis and the Cross Axis.](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)***

How would you implement that? Notice that our Main Axis is horizontal; this gives us a clue that we're using flexDirection: row. The boxes are in the center of both axes which means we're using justifyContent: 'center' and alignItems: 'center'.

**const** styles = StyleSheet.create({

container: {

flex: 1,

flexDirection: 'row',

justifyContent: 'center',

alignItems: 'center',

},

box: {

width: 50,

height: 50,

backgroundColor: '#e76e63',

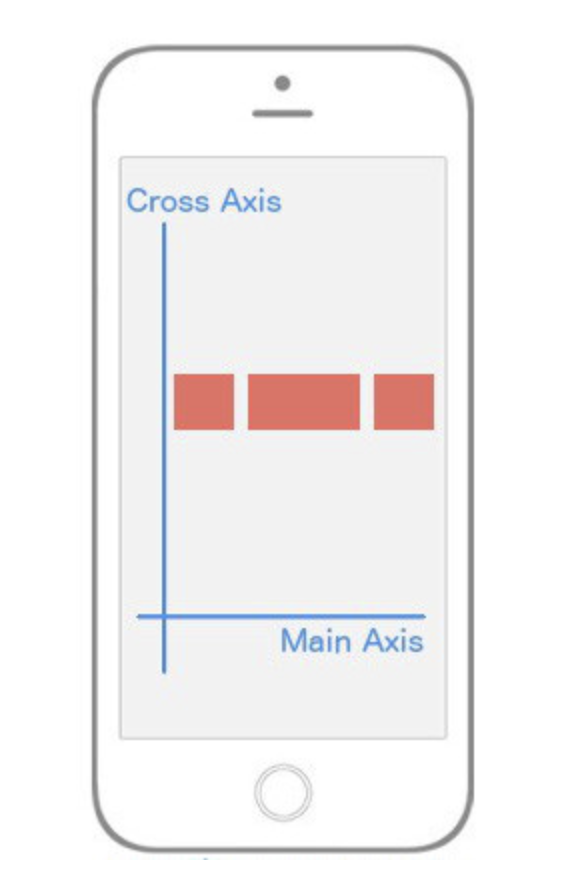
margin: 10,

}

})

**The Flex Property**

But now, what if we wanted to change our UI to look like this:

**[[](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)**

***[Using the flex property to change the rate at which a flex items increases its size comparable to other flex items.](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)***

In the above image, it's exactly the same layout -- but now the middle section is twice as wide as its siblings! This is what the flex property allows us to do. Here’s the code:

**class** **FlexboxExamples** **extends** **Component** {

render() {

**return** (

<View style={styles.container}>

<View style={[styles.box, {flex: 1}]}/>

<View style={[styles.box, {flex: 2}]}/>

<View style={[styles.box, {flex: 1}]}/>

</View>

)

}

}

**const** styles = StyleSheet.create({

container: {

flex: 1,

flexDirection: 'row',

justifyContent: 'center',

alignItems: 'center',

},

box: {

width: 50,

height: 50,

backgroundColor: '#e76e63',

margin: 10,

}

})

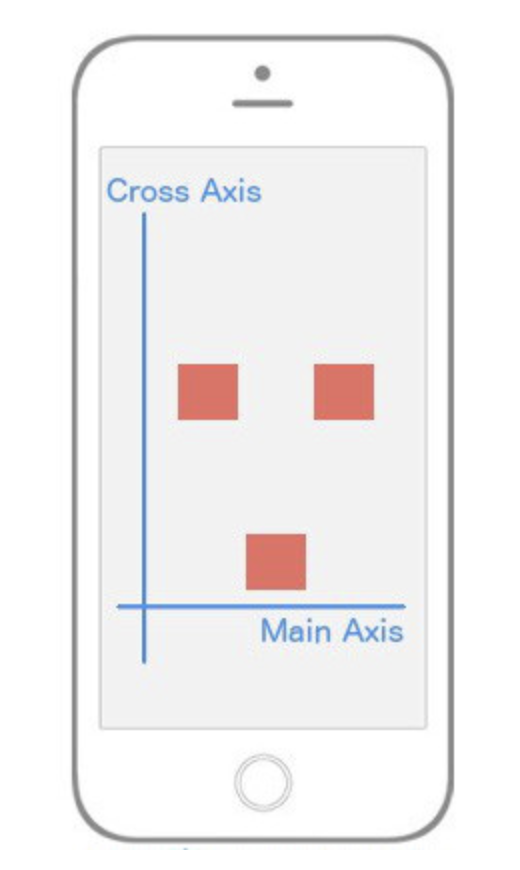
**export** **default** FlexboxExamples;

Notice I didn't add any styles; I just made the middle sibling have flex: 2 while the other siblings have flex: 1. This basically says "make sure that the middle sibling is twice as large along the Main Axis as the first and third children". This is the reason why flex can replace percentages because usually a percent-based layout is just one where specific elements are relative to other elements, exactly like we're doing above. It's also important to note that if you place flex: 1 on an element, that element is going to take up as much space as its parent takes up. That's why in most of our examples above because we want our "layout area" to be the size of the parent, which in our examples was the whole viewport.

Let's go even deeper!

**Aligning Individual Flex Items**

What if we wanted a layout like this?

**[[](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)**

***[alignSelf: flex-end changes the flex item it targets to appear at the end of the Cross Axis.](https://classroom.udacity.com/nanodegrees/nd019/parts/1187639a-82e2-4dd8-9d7b-477af14e5dd7/modules/cc51381c-49d3-43b4-86c3-dd55f9b8a3f6/lessons/c6957bbe-51e4-4d4e-b634-0aee13b4d11b/concepts/55f4f0ad-6736-4a54-9fa9-1af5e9ed4d61)***

It's as if the first and third element are centered both vertically and horizontally, but that second element has a mind of its own and is using flex-end along the Cross Axis. To implement this, we'll need a way to have the child element override a specific positioning it received from its parent. Good news: that's exactly what alignSelf allows us to do! Notice it begins with *align*, so just like alignItems, it's going to position itself along the Cross Axis. It also has the exact same options as alignItems (flex-start, flex-end, center, stretch).

The code to implement the image above is:

**class** **FlexboxExamples** **extends** **Component** {

render() {

**return** (

<View style={styles.container}>

<View style={styles.box}/>

<View style={[styles.box, {alignSelf: 'flex-end'}]}/>

<View style={styles.box}/>

</View>

)

}

}

**const** styles = StyleSheet.create({

container: {

flex: 1,

flexDirection: 'row',

justifyContent: 'center',

alignItems: 'center',

},

box: {

width: 50,

height: 50,

backgroundColor: '#e76e63',

margin: 10,

}

})

**export** **default** FlexboxExamples;

Note that all we've done is add alignSelf: flex-end to the second child element and that overrode what it was instructed to do by the parent (alignItems: 'center').

If you've made it all the way through this, great work! I realize that was a lot to cover but I hope it's helped you get up and running with styling (and specifically flexbox) on React Native.