React Native 中使用 Flexbox 筆記

Flexbox 排版的全名是Flexible Box Layout Module(彈性的盒排版模組)，是由 W3C 所制定的 CSS3 排版模組。Flexbox 的目的是提供更有效率的排版、對齊與分佈空格在容器之中的項目，甚至是在項目的尺寸大小是未知或動態的情況。

Flexbox 排版適合於應用程式中的元件，以及小型規模的排版。W3C 也另外制定了一個Grid 排版模組，這是適合使用於大型規模排版情況。

> 註: Flex 的中文意思是"彈性的"、"靈活的"，box 的中文意思是"盒子"、"箱子"。

Yoga 專案

Yoga 是一個實作了 Flexbox 的跨平台排版引擎，由 Facebook 主導，前身的名稱為 css-layout。Yoga 主要以 C 語言開發，提供高執行效能與易於整合其它語言，可以讓開發者在不同的平台中使用 Flexbox 來進行排版的工作，主要是應用在手機 App 的開發上，當然它也是目前 React Native 使用的排版引擎。

Yoga 中採用了與 CSS3 標準裡 Flexbox 相近的語法與方式，但因為 CSS3 中的標準與內容相當多，Yogo 中的 Flexbox 並沒有完全支援所有的特性，而進行部份的簡化，在 React Native 中使用的預設值與網頁上也不相同，所以在使用過程中要特別注意的。

Flexbox 使用的英文專業字詞

要學習 Flexbox 並不困難，很多內容都是很直覺式的，如果你已經是對網頁設計或 HTML 語法有一定基礎的開發者，很容易就可以學習開始使用。不過，因為這是外國人寫的標準，有一些專有名詞反倒是會造成一知半解的情況，我先把會用到的幾個英文名詞寫在下面，先理解一下這些名詞所代表的意思，後面學習會容易得多。

- flex: 有"彈性"、"彎曲"的意思，通常作為動詞用。flexible(彈性的)是這個字詞的形容詞。
- axis: 有"坐標軸"、"基準線"的意思。Flexbox 使用 main axis(主基準線)與 cross axis(相反基準線)來區分水平與垂直的兩個不同的基準線，main axis(主基準線)代表的主要的基準線，依照 flexDirection 的屬性來決定主基準線是水平的還是垂直的方向。cross axis(相反基準線)則是與主基準線相反的基準線。基準線代表的是容器(container)中的項目排列時按照的方向規範。

> React Native 預設的方向(flexDirection)是 column，也就是"**垂直的**"，所以它的主基準線(main axis)預設就是"**垂直的**"，相反基準線(cross axis)則是水平的。這會與網頁中 CSS 的預設值相反。

- container 與 items: container(容器)中裝有 items(項目)，這兩個在 DOM 元素中有 父母-子女(parent-children) 關係。Flexbox 的排列、對齊、方向等等屬性是在 **container(容器)** 中定義的，當然這是指對容器中的項目們所作的屬性定義。
- grow、shrink、basis: 這三個是項目的屬性定義值。grow 有"成長"、"變大"的意思，shrink 則有"收縮"、"縮水"的意思。basis 則是"基礎"、"標準"的意思。這三個是用於項目(元件)的"**彈性**"屬性，也就是在版面或外部容器的尺寸變動時，項目應該如何隨之進行變動。

> React Native 中自 0.42 版後才加入 flexGrow、flexShrink、flexBasis 這三個屬性，使用上會有些複雜，請參考flex vs flexGrow vs flexShrink vs flexBasis in React Native?

- wrap、nowrap、wrap-reverse: wrap 有"包裝"的意思，但用在這裡是代表要不要擠在同一行之中， `nowrap` 代表這個容器中的項目是只能擠在同一行之中， `wrap` 則是允許多行。 `wrap-reverse` 與 `wrap` 一樣可以允許多行，但開始與結束會顛倒過來。 `-reverse` 字詞很常見，它是"顛倒"的意思。
- align-items、align-self、align-content: align 是"對齊"的意思。align-self 是用於項目(元件)的屬性，align-items 與 align-content 是用於容器的屬性，都是用來作對齊使用的。這幾個都是應用在 cross axis(相反基準線)時。
- stretch: "伸展"的意思，用在這是是填好填滿的意思。
- justify-content: justify 也有"對齊"的意思，它是用在 main axis(主基準線)時的對齊。這是用於容器的屬性。
- flow: flow 是"流動"的意思，在這裡的意思是排列的情況。flex-flow 是 flex-direction 與 flex-wrap 的組合簡寫法，合稱 `dw` 。(註: React Native 中沒這個屬性)
- order: "順序"的意思，項目(元件)的屬性，代表這個項目(元件)在容器中的排列次序。(註: React Native 中沒這個屬性)

justifyContent 詳細說明

> 預設值: `flex-start`

`justifyContent` 是針對 main axis(主基準線)的對齊，對 React Native 來說，main axis(主基準線)預設是從上到下的垂直方向。有以下幾個屬性值可使用:

- flex-start (預設值): 以開始線來對齊，例如從右至左的排列方向，最右邊即是開始線。
- flex-end: 以結束線來對齊，和上面的屬性顛倒。
- center: 以中線來對齊。
- space-between: 項目會平均分散對齊，第一個項目會位於開始線，最後一個項目位於結束線。
- space-around: 項目會平均分散對齊，每個項目之間約有二個單位的空格，第一個項目會與開始線之間有一個單位的空格，最後一個項目與結束線之間也有一個單位的空格。

alignItems

> 預設值: `stretch`

> 特別注意: 預設情況(水平方向)如果給定 width 屬性，會導致設置 `stretch` 值無效果

> 重點: alignItems 是 justifyContent 的相反屬性(不要誤以為是 alignContent)

`alignItems` 則是針對 cross axis(相反基準線)的對齊，對 React Native 來說，cross axis(相反基準線)預設是從左到右的水平方向。適合於項目是單行排列的情況。有以下幾個屬性:

- flex-start: 以 cross axis(相反基準線)的開始線來對齊。
- flex-end: 以 cross axis(相反基準線)的結束線來對齊。
- center: 以 cross axis(相反基準線)的中心線來對齊。
- baseline: 項目會依照它們的 baseline 來對齊。(baseline 預設是 DOM 元素呈現的最下面位置)
- stretch (預設值): 伸展填好填滿容器(注意仍然會受到 min-width/max-width 的限制，也就是項目的高度或寬度限制)

alignContent

> 注意: alignContent 只有在項目被 flexWrap 包裹(wrapped)成為多行排列時，才會發生作用

`alignContent` 也是針對 cross axis(相反基準線)的對齊對 React Native 來說，cross axis(相反基準線)預設是從左到右的水平方向。它適合於 cross axis(相反基準線)有空白的情況，以及項目是以多行排列的情況。

屬性值與 `justifyContent` 類似，但多了一個 stretch:

- flex-start: 以開始線來對齊，例如從右至左的排列方向，最右邊即是開始線。
- flex-end: 以結束線來對齊，和上面的屬性顛倒。
- center: 以中線來對齊。
- space-between: 項目會平均分散對齊，第一個項目會位於開始線，最後一個項目位於結束線。
- space-around: 項目會平均分散對齊，每個項目之間約有二個單位的空格，第一個項目會與開始線之間有一個單位的空格，最後一個項目與結束線之間也有一個單位的空格。
- stretch (預設值): 伸展填好填滿容器(注意仍然會受到 min-width/max-width 的限制，也就是項目的高度或寬度限制)

項目的自我對齊 `alignSelf`

> 注意: 這是單個**項目**用來覆蓋**容器**裡的 `alignItems` 值，有必要才會使用

這個自我對齊 `alignSelf` 項目屬性是用來覆蓋在容器裡設定的 `alignItems` ，作為個別項目自己的對齊方式，除非有必要才會使用。預設值是 `auto` ，其他屬性與 `alignItems` 一樣:

- auto (預設值): 自動。
- flex-start: 以 cross axis(相反基準線)的開始線來對齊。
- flex-end: 以 cross axis(相反基準線)的結束線來對齊。
- center: 以 cross axis(相反基準線)的中心線來對齊。
- baseline: 項目會依照它們的 baseline 來對齊。(baseline 預設是 DOM 元素呈現的最下面位置)
- stretch: 伸展填好填滿容器(注意仍然會受到 min-width/max-width 的限制，也就是項目的高度或寬度限制)

Flexbox 定義步驟說明

第零步: 容器與項目的標記

Flexbox 需要有容器與項目的層級結構，才能進行相對的排版，例如以下的範例:

```
<View style={styles.container}>
  <View style={styles.row}>
    <Text>Row-1</Text>
  </View>
  <View style={styles.row}>
    <Text>Row-2</Text>
  </View>
  <View style={styles.row}>
    <Text>Row-3</Text>
  </View>
  <View style={styles.row}>
    <Text>Row-4</Text>
  </View>
</View>
```

第一步: 定義容器&項目的屬性 `flex: 1`

所有的 View 元件預設是 `flex: 0` ，所以你要自行定義 `flex` 值才能進行排版。定義之後，容器會有一些預設的屬性加上，即使你沒有寫上

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#ccc',
  },
  row: {
    flex: 1,
  },
})
```

container 會有以下的自動(預設)的屬性:

- flexDirection: 'column'
- flexWrap: 'nowrap'
- justifyContent: 'flex-start'
- alignItems: 'stretch'
- alignContent: 'flex-start' (註: 需要配合 `flexWrap: 'wrap'` 出現多列時才會有效果)

由此可知這樣設定，會出現以下的排版:

- `flex:1` 相當於 `height: 100%`
- `alignItems: 'stretch'` 相當於 `width: 100%` (每個項目)

第二步: 定義容器的 `flexDirection` ，控制主基準線是垂直還水平方向

flexDirection 在 RN 中預設是 column，也就是垂直排列(由上到下)的。

第三步: 定義容器的 `flexWrap` ，控制裡面的項目是單行還多行排序。

flexWrap 在 RN 中預設是 nowrap，也就是單列或單行的排列。

項目所佔比例

> flex 的數值相當於 flex-grow

利用 flex 的數值，可以劃分出每個項目所佔的比例值，例如下面的例子會讓後面兩列佔的比例為 2/(1+1+2+2) 也就是 1/3，前面兩列為 1/6:

```
<View style={styles.container}>
  <View style={{ flex: 1 }}>
    <Text>Row-1</Text>
  </View>
  <View style={{ flex: 1 }}>
    <Text>Row-2</Text>
  </View>
  <View style={{ flex: 2 }}>
    <Text>Row-3</Text>
  </View>
  <View style={{ flex: 2 }}>
    <Text>Row-4</Text>
  </View>
</View>
```

使用對應的比例數字也是相同的結果:

```
<View style={styles.container}>
  <View style={{ flex: 1 / 6 }}>
    <Text>Row-1</Text>
  </View>
  <View style={{ flex: 1 / 6 }}>
    <Text>Row-2</Text>
  </View>
  <View style={{ flex: 2 / 6 }}>
    <Text>Row-3</Text>
  </View>
  <View style={{ flex: 2 / 6 }}>
    <Text>Row-4</Text>
  </View>
</View>
```

或

```
<View style={styles.container}>
  <View style={{ flex: 0.16 }}>
    <Text>Row-1</Text>
  </View>
  <View style={{ flex: 0.16 }}>
    <Text>Row-2</Text>
  </View>
  <View style={{ flex: 0.33 }}>
    <Text>Row-3</Text>
  </View>
  <View style={{ flex: 0.33 }}>
    <Text>Row-4</Text>
  </View>
</View>
```

> 註: 當 flex 數字為小數點數字，但沒有加起來剛好為 1 時，會呈現有多餘空間的情況。所以使用整數可能會是比較好的比例劃分方式。

> 註: 由於在垂直排列(flexDirection: column)時，flexbox 取代了 height 的樣式定義方式，所以再對 View 定義 height 是不會有作用的，例如將上面例子中的 Row-1 改為下面的設定，你會發現 height 不管設大或設小都沒作用。

```
<View
  style={{
    width: 100,
    height: 10,
    backgroundColor: '#ff0',
    flex: 1,
  }}
>
  <Text>Row-1</Text>
</View>
```

> 註: 只有 View 元件可以使用 flexbox 排版，Text 元件有自己的排版方式

> 註: 使用 StyleSheet 建立和直接使用物件字面定義的樣式有不同之處，上面範例只是為了展示之用，正常情況下仍然建議使用 StyleSheet 來建立。

flexbox in RN

與 CSS3 相較的不同處

- flexDirection 預設為 `column`
- alignItems 預設為 `stretch`
- flex 只能用**一個數字**

預設情況

```
{
  box-sizing: border-box;
  position: relative;

  display: flex;
  flex-direction: column;
  align-items: stretch;
  flex-shrink: 0;
  align-content: flex-start;
  justify-content: flex-start;

  border: 0 solid black;
  margin: 0;
  padding: 0;
  min-width: 0;
}
```

- flex 是預設的，所有呈現都是 flexbox。只是預設值是 `flex: 0`
- 上面的每個元素都是 `position: relative`

額外說明: flex 數字意義

> 註: flex 通常是 `flex: 1` 。這與 CSS3 中的 flex 定義方式不同。

只能使用一個數字。(flex ReactPropTypes.number)

正數代表元件是有"彈性的"，會隨著尺寸等比例放大，例如設為 2 的元件佔的大小是 1 的兩倍。類似於 `flexGrow` 。

當為**0**時，代表元件沒有彈性，只是對應所設置的寬與高度。

當為**-1**時，代表元件一般情況對應所設置的寬與高度，當空間不足時會縮小(shrink)到 minWidth 與 minHeight。類似於 `flexShrink` 。

如果你想要某個元件佔 25%，另一個元件佔 75%，就用 `flex: 1/4` 與 `flex: 3/4` 分別指定兩個元件。

額外說明: alignItems

alignItems 要有作用(stretch，填好填滿)，子元素必須沒有設定對應 cross axis(相反基準線)的寬度(或高度)才會有作用。

小抄

> 最前面一個值為 RN 預設值

```
# Choose the main axis:
flexDirection: 'column' || 'row'

# Align the main axis:
justifyContent: 'flex-start' || 'flex-end' || 'center' || 'space-around' || 'space-between'

# Align the cross axis:
```

```
alignItems: 'stretch' || 'flex-start' || 'flex-end' || 'center'

# Align individual elements:
alignSelf: 'flex-start' || 'flex-end' || 'center' || 'stretch' || 'auto'

# Give it a wrap:
flexWrap: 'nowrap' || 'wrap'

# Define relative fluidity of an element:
flex: number (e.g. 1, 1/2)
```

議題

全寬度 & 全高度

方法一: 用 alignSelf: "stretch"可以設定某個元件用全寬度或全高度:

```
line1: {
    backgroundColor: '#FDD7E4',
    alignSelf: 'stretch',
    textAlign: 'center',
},
```

方法二: 用 Dimensions 可以獲得，不一定得要用 flex 來設定:

```
var {
  AppRegistry,
  ...
  Dimensions
} = React;

var width = Dimensions.get('window').width; //full width
var height = Dimensions.get('window').height; //full height


line1: {
    backgroundColor: '#FDD7E4',
    width: width,
},
```

一些密訣

If the element has a parent container, child element can be positioned with Flex item property— `align-self` inside parent container still being Flex container, behaving and accepting Flex container properties.

- Flex container children always follows parent container Main Axis direction
- Hint: to prevent expanding full width after declaring flex: 1, use maxWidth/maxHeight property.

狀態列高度

由於目前手機都有狀態列在最上面(垂直排列時)，使用 expo 所提供的常數 Constants.statusBarHeight 可以取得這個高度，範例如下:

```
import React, { Component } from 'react'
import { View, StyleSheet } from 'react-native'
import { Constants } from 'expo'
import { Button } from 'react-native-elements'

import '@expo/vector-icons'

export default class App extends Component {
  render() {
```

```
    return (
      <View style={styles.container}>
        <Button title="Yea" containerViewStyle={{ width: '100%', marginLeft: 0 }} />
      </View>
    )
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    paddingTop: Constants.statusBarHeight,
    backgroundColor: '#ecf0f1',
  },
})
```

flex vs flexGrow vs flexShrink vs flexBasis in React Native?

https://stackoverflow.com/questions/43143258/flex-vs-flexgrow-vs-flexshrink-vs-flexbasis-in-react-native