


微信小程序学习记录《三》：视图容器

1. Viewflex-direction:row:横向布局模式；如果不设置该属性，默认为横向 flex-direction:column:纵向布局。界面样式代码注意：1.如果想要改模式有效，父控件必须设置显示方式为 flex 模式，display:flex; 2.要想控件的 ...

1. View

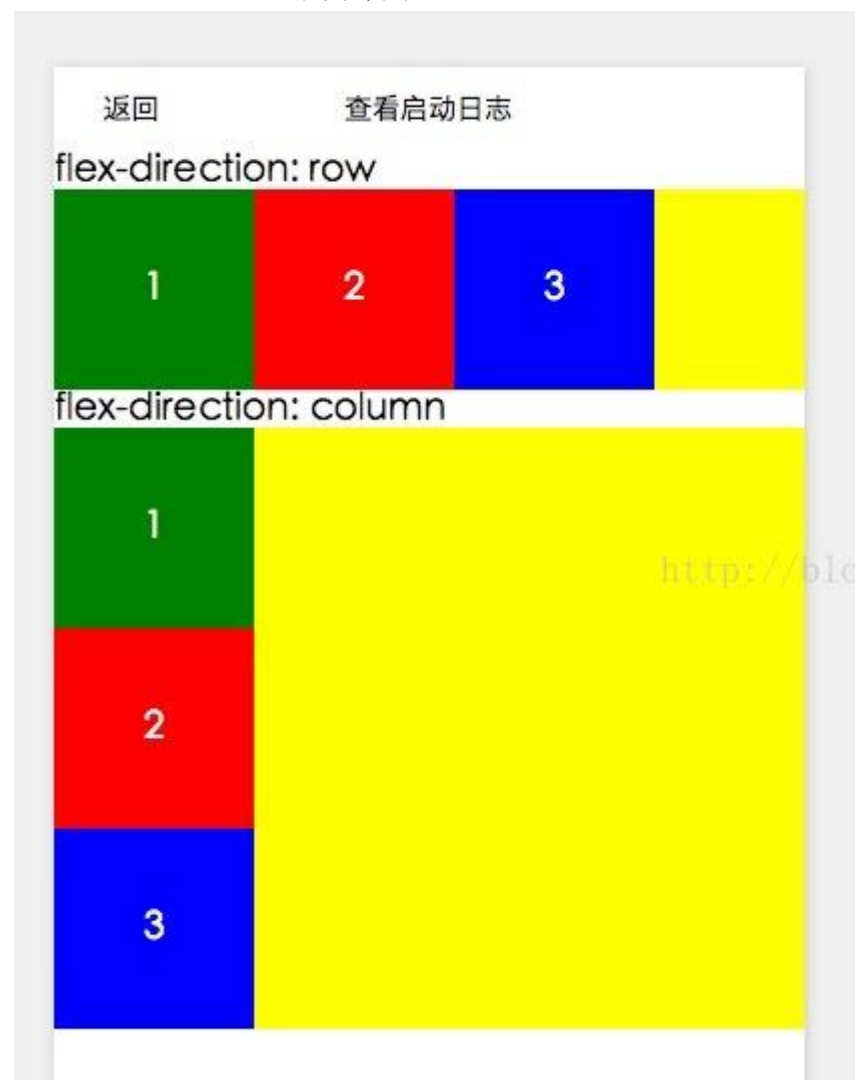


The screenshot displays the WeChat Mini Program IDE interface. On the left, a simulator shows the rendered UI. The top section, titled 'flex-direction: row', contains three colored boxes (green, red, blue) labeled 1, 2, and 3. The bottom section, titled 'flex-direction: column', contains the same three colored boxes stacked vertically. On the right, the WXML code is shown. Two red boxes highlight the styling and structure of the flex containers. The first box highlights the row layout code, and the second box highlights the column layout code. Red arrows point from the code boxes to the corresponding UI elements in the simulator.

```
1 <view class="section">
2   <view class="section__title">flex-direction: row</view>
3   <view class="flex-wrp" style="flex-direction:row;">
4     <view class="flex-item bc_green">1</view>
5     <view class="flex-item bc_red">2</view>
6     <view class="flex-item bc_blue">3</view>
7   </view>
8 </view>
9 <view class="section">
10  <view class="section__title">flex-direction: column</view>
11  <view class="flex-wrp" style="height: 300px;flex-direction:column;">
12    <view class="flex-item bc_green">1</view>
13    <view class="flex-item bc_red">2</view>
14    <view class="flex-item bc_blue">3</view>
15  </view>
16 </view>
17
```

flex-direction:row:横向布局模式；如果不设置该属性，默认为横向

flex-direction:column:纵向布局。



```
1
2  /*父控件必须设置为flex模式*/
3  .flex-wrp {
4      display: flex;
5      background-color: yellow;
6  }
7
8  .flex-item {
9      height: 100px;
10     width: 100px;
11     color: white;
12     line-height: 100px;
13     text-align: center;
14 }
15
16 .bc_green {
17     background-color: green;
18 }
19
20 .bc_red {
21     background-color: red;
22 }
23
24 .bc_blue {
25     background-color: blue;
26 }
```

界面样式代码

注意：1.如果想要改模式有效，父控件必须设置显示方式为 flex 模式，display:flex;

2.要想控件的背景颜色显示出来，必须 view 设置大小，否则 background-color 属性无效。

2.scroll-view

scroll-view

可滚动视图区域。

属性名	类型	默认值	说明
scroll-x	Boolean	false	允许横向滚动
scroll-y	Boolean	false	允许纵向滚动
upper-threshold	Number	50	距顶部/左边多远时（单位px），触发 scrolltoupper 事件
lower-threshold	Number	50	距底部/右边多远时（单位px），触发 scrolltolower 事件
scroll-top	Number		设置竖向滚动条位置
scroll-left	Number		设置横向滚动条位置
scroll-into-view	String		值应为某子元素id，则滚动到该元素，元素顶部对齐滚动区域顶部
bindscrolltoupper	EventHandle		滚动到顶部/左边，会触发 scrolltoupper 事件
bindscrolltolower	EventHandle		滚动到底部/右边，会触发 scrolltolower 事件
bindscroll	EventHandle		滚动时触发，event.detail = {scrollLeft, scrollTop, scrollHeight, scrollWidth, deltaX, deltaY}

返回

查看启动日志

vertical scroll

垂直滚动部分

click me to scroll into view

click me to scroll

horizontal scroll: //blog.csdn.net/

水平滚动部分

两个按钮，点击的时候通过改变
scroll-into-view参数的值，动态
滑动，下面按钮改变滚动距离

[html] [view plain copy](#)

```
1. <view class="section">
2.   <view class="section__title">vertical scroll</view>
3.   <!--如果是垂直滚动，必须设置 scrollview 的高度，切记-->
4.   <scroll-view scroll-y="true" style="height: 200px;" bindscrolltoupper="upper" bindscrolltolower="lower" bindscroll="scroll" scroll-into-
      view="{{toView}}" scroll-top="{{scrollTop}}">
5.     <view id="green" class="scroll-view-item bc_green"></view>
6.     <view id="red" class="scroll-view-item bc_red"></view>
7.     <view id="yellow" class="scroll-view-item bc_yellow"></view>
8.     <view id="blue" class="scroll-view-item bc_blue"></view>
9.   </scroll-view>
10.
11. <view class="btn-area">
12.   <button size="mini" bindtap="tap">click me to scroll into view </button>
13.   <button size="mini" bindtap="tapMove">click me to scroll</button>
14. </view>
15. </view>
16. <view class="section section_gap">
17.   <view class="section__title">horizontal scroll</view>
18.   <scroll-view class="scroll-view_H" scroll-x="true" style="width: 100%">
19.     <view id="green" class="scroll-view-item_H bc_green"></view>
20.     <view id="red" class="scroll-view-item_H bc_red"></view>
21.     <view id="yellow" class="scroll-view-item_H bc_yellow"></view>
22.     <view id="blue" class="scroll-view-item_H bc_blue"></view>
23.   </scroll-view>
24. </view>
```

界面渲染代码:

[css] [view plain copy](#)

```
1. .section {
2.     display: flex;
3.     flex-direction: column;
4. }
5. /*设置垂直滚动每个滑动块高度*/
6. .scroll-view-item {
7.     height: 100px;
8. }
9. /*设置中间两个按钮的位置*/
10. .btn-area {
11.     height: 80px;
12.     justify-content: space-around;
13.     display: flex;
14.     flex-direction: column;
15. }
16. /*设置水平滚动视图, 该属性必须设置为 nowrap, 表示超出范围也不换行
17. white-space 属性的值
18.     normal: 自动换行(默认.内容超过父控件宽度换行)
19.     pre: 保持 HTML 源代码的空格与换行, 等同与 pre 标签, 识别空格和换行符
20.     nowrap: 不换行,超出范围的隐藏
21.     pre-wrap: 同 pre 属性,但是遇到超出容器范围的时候会自动换行
22.     pre-line: 同 pre 属性,但是遇到连续空格会被看作一个空格
23.     inherit: 继承
24. */
25. .scroll-view_H {
26.     white-space: nowrap;
27. }
28. /*设置水平滚动的所有元素的大小,view 默认为块元素, 会自动换行, 所以必须设置为行内元素*/
29. .scroll-view-item_H {
30.     width: 150px;
31.     height: 100px;
32.     display: inline-block;
```

```
33. }
34. /*设置每个 View 的背景颜色*/
35. .bc_green {
36.   background-color: green;
37. }
38. .bc_red {
39.   background-color: red;
40. }
41. .bc_blue {
42.   background-color: blue;
43. }
44. .bc_yellow {
45.   background-color: yellow;
46. }
```

页面交互代码:

[javascript] [view plain copy](#)

```
1. //logs.js
2. var util = require('../utils/util.js')
3. var order = ['red', 'yellow', 'blue', 'green', 'red']
4. Page({
5.   data: {
6.     //元素的 IDID, 通过修改这个, 实现点击把这个元素滚动到顶部
7.     toView: 'red',
8.     //默认你已经滚动的距离
9.     scrollTop: 100
10.  },
11.  upper: function(e) {
12.    console.log(e)
13.  },
14.  lower: function(e) {
15.    console.log(e)
```

```
16.  },
17.  scroll: function(e) {
18.      console.log(e)
19.  },
20.  //点击改变要滚动到顶部的元素 ID
21.  tap: function(e) {
22.      for (var i = 0; i < order.length; ++i) {
23.          if (order[i] === this.data.toView) {
24.              //动态刷新数据，向上滚动一页
25.              this.setData({
26.                  toView: order[i + 1]
27.              })
28.              break
29.          }
30.      }
31.  },
32.  //点击一次滚动 10
33.  tapMove: function(e) {
34.      this.setData({
35.          scrollTop: this.data.scrollTop + 10
36.      })
37.  }
38. })
```

3.swiper 主要用于图片轮播，广告 banner 的展示

返回

查看启动日志



indicator-dots

autoplay

interval

duration

```
1 //logs.js
2 var util = require('../../utils/util.js')
3 Page({
4   data: {
5     imgUrls: [
6       'http://img02.tooopen.com/images/20150928/tooopen_sy_143912755726.jpg',
7       'http://img06.tooopen.com/images/20160818/tooopen_sy_175866434296.jpg',
8       'http://img06.tooopen.com/images/20160818/tooopen_sy_175833047715.jpg'
9     ],
10    indicatorDots: false,
11    autoplay: false,
12    interval: 5000,
13    duration: 1000
14  },
15  //设置是否显示指示点
16  changeIndicatorDots: function(e) {
17    this.setData({
18      indicatorDots: !this.data.indicatorDots
19    })
20  },
21  //设置是否自动播放
22  changeAutoplay: function(e) {
23    this.setData({
24      autoplay: !this.data.autoplay
25    })
26  },
27  //监测slider的滑动，设置自动切换的时间
28  intervalChange: function(e) {
29    this.setData({
30      interval: e.detail.value
31    })
32  },
33  //监测slider的滑动，设置每次切换页面的动画时长
34  durationChange: function(e) {
35    this.setData({
36      duration: e.detail.value
37    })
38  }
39 })
40
```

监听按钮点击，设置指示点是否！

监听按钮点击，设置是否自动播放

监听slider滑动，设置页面切换的时间，和页

滑块视图容器。

属性名	类型	默认值	说明
indicator-dots	Boolean	false	是否显示面板指示点
autoplay	Boolean	false	是否自动切换
current	Number	0	当前所在页面的 index
interval	Number	5000	自动切换时间间隔
duration	Number	1000	滑动动画时长
bindchange	EventHandle		current 改变时会触发 change 事件，event.detail = {current: current}

iPhone 6

wifi

正在调试2个页面



logs.wxml

logs.wxss

logs.js

index.js

index.wxss

index.js

返回

查看启动日志

指示点，表示有
三个页面

indicator-dots

设置是否显示指示点

autoplay

设置是否自动播放

修改页面切换时间间隔

500

interval

修改页面每次滚动时动画执行时间

1000

duration

```

1 <view class="btn">
2   <swiper indicator-dots="{{indicatorDots}}"
3     autoplay="{{autoplay}}" interval="{{interval}}" duration="{{duration}}">
4     <block wx:for="{{imgUrls}}">
5       <swiper-item>
6         <image src="{{item}}" class="slide-image" width="355" height="150"/>
7       </swiper-item>
8     </block>
9   </swiper>
10  <button bindtap="changeIndicatorDots"> indicator-dots </button>
11  <button bindtap="changeAutoplay"> autoplay </button>
12  <slider bindchange="intervalChange" show-value min="500" max="2000"/> interval
13  <slider bindchange="durationChange" show-value min="1000" max="10000"/> duration
14 </view>

```

<http://blog.csdn.net/>

返回

查看启动日志



indicator-dots

autoplay

interval

duration

```
1 //logs.js
2 var util = require('../../utils/util.js')
3 Page({
4   data: {
5     imgUrls: [
6       'http://img02.tooopen.com/images/20150928/tooopen_sy_143912755726.jp
7       'http://img06.tooopen.com/images/20160818/tooopen_sy_175866434296.jp
8       'http://img06.tooopen.com/images/20160818/tooopen_sy_175833047715.jp
9     ],
10    indicatorDots: false,
11    autoplay: false,
12    interval: 5000,
13    duration: 1000
14  },
15  //设置是否显示指示点
16  changeIndicatorDots: function(e) {
17    this.setData({
18      indicatorDots: !this.data.indicatorDots
19    })
20  },
21  //设置是否自动播放
22  changeAutoplay: function(e) {
23    this.setData({
24      autoplay: !this.data.autoplay
25    })
26  },
27  //监测slider的滑动，设置自动切换的时间
28  intervalChange: function(e) {
29    this.setData({
30      interval: e.detail.value
31    })
32  },
33  //监测slider的滑动，设置每次切换页面的动画时长
34  durationChange: function(e) {
35    this.setData({
36      duration: e.detail.value
37    })
38  }
39 })
40
```

监听按钮点击，设置指示点是否!

监听按钮点击，设置是否自动播放

监听slider滑动，设置页面切换的时间，和页

界面渲染代码:

[css] [view plain copy](#)

```
1. .btn {  
2.   display: flex;  
3.   flex-direction: column;  
4.   height: 400px;  
5.   justify-content: space-around;  
6.   text-align: center;  
7. }
```