

首页 > iOS开发

react-native 之布局篇

2015-04-20 09:02 编辑: lansekuangtu 分类: iOS开发 来源: tmlife的Github

0 33033

react-native 布局

招聘信息: 微信书城开发

宽度单位和像素密度

react的宽度不支持百分比，设置宽度时不需要带单位 {width: 10}，那么10代表的具体宽度是多少呢？

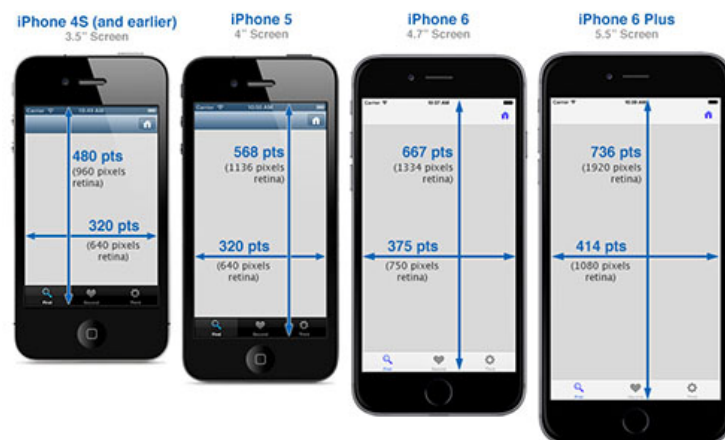
不知道是官网文档不全还是我眼瞎，反正是没找到，那做一个实验自己找吧：

```
var Dimensions = require('Dimensions');
<Text style={styles.welcome}>
  window.width={Dimensions.get('window').width + '\n'}
  window.height={Dimensions.get('window').height + '\n'}
  pxieRatio={PixelRatio.get()}
</Text>
```

默认用的是iPhone6的模拟器结果是：

```
window.width=375
window.height=667
```

我们知道iPhone系列的尺寸如下图：



可以看到iphone 6的宽度为 375pt，对应了上边的375，由此可见react的单位为pt。那如何获取实际的像素尺寸呢？这对图片的高清化很重要，如果我的图片大小为100*100 px. 设置宽度为100 * 100. 那在iphone上的尺寸就是模糊

热门资讯



一步一步构建你的iOS网络层 - HTTP篇

点击量 8911



两步搞定，App断网提醒设计

点击量 7774



iOS超全开源框架、项目和学习资料汇总--数

点击量 7660



iOS开发经验总结

点击量 7232



一步一步构建你的iOS网络层 - TCP篇

点击量 6848



如何接手一个老旧的iOS项目

点击量 6364



让您的Xcode键盘如飞

点击量 5974



我是如何花了一年时间来学机器学习的

点击量 4905



WWDC 2017时间定6月5日，哪些没变，哪

点击量 4654



iOS 10.3 beta 3更新新增应用兼容性选项

点击量 4512

综合评论

厉害了 我的哥!

huangwenxia153 评论了 iOS 组件化

—— 路由设计思路分析...

看完我的菊花 微微一紧

xmg93 评论了 哈哈哈哈哈哈哈哈哈哈

哈哈《iOS界裁员：从入门到精通...

确实涨姿势了

coco_for_LJ 评论了 谈谈 iOS 中图片

的解压缩...

呵呵

的。这个时候需要的图像大小应该是 100 * pixelRatio 的大小。

react 提供了 **PixelRatio** 的获取方式

```
var image = getImage({
  width: 200 * PixelRatio.get(),
  height: 100 * PixelRatio.get()
});
<Image source={image} style={{width: 200, height: 100}} />
```

flex的布局

默认宽度

我们知道一个div如果不设置宽度，默认的会占用100%的宽度，为了验证100%这个问题，做三个实验

1. 根节点上方一个View，不设置宽度
2. 固定宽度的元素上设置一个View，不设置宽度
3. flex的元素上放一个View宽度，不设置宽度

```
<Text style={[styles.text, styles.header]}>
  根节点上放一个元素，不设置宽度
</Text>

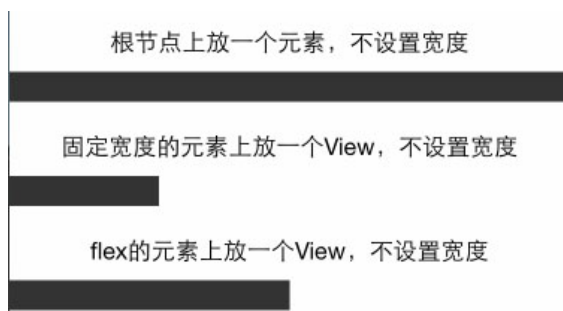
<View style={{height: 20, backgroundColor: '#333333'}} />

<Text style={[styles.text, styles.header]}>
  固定宽度的元素上放一个View，不设置宽度
</Text>

<View style={{width: 100}}>
  <View style={{height: 20, backgroundColor: '#333333'}} />
</View>

<Text style={[styles.text, styles.header]}>
  flex的元素上放一个View，不设置宽度
</Text>

<View style={{flexDirection: 'row'}}>
  <View style={{flex: 1}}>
    <View style={{height: 20, backgroundColor: '#333333'}} />
  </View>
  <View style={{flex: 1}}/>
</View>
```



结果可以看到flex的元素如果不设置宽度，都会百分之百的占满父容器。

水平垂直居中

css 里边经常会做的事情是去讲一个文本或者图片水平垂直居中，如果使用过css 的flexbox当然知道使用alignItems 和 justifyContent . 那用react-native也来做一下实验

superlee 评论了 新版诺基亚3310上手：很有范儿让一切回归简单...

马克

bobbybiao 评论了 我的iOS工程结构

还不如出个大屏幕的 老人机呢... 这屏幕 键盘太小了..

OS_x_Mac 评论了 新版诺基亚3310上手：很有范儿让一切回归简单...

mark

ss_Ye 评论了 iOS开发之Runtime常用示例总结...

这个概念确实难搞懂，还需要深入研究
www.5200pt.com

vlay 评论了 深入理解RunLoop

请问如果https协议的话该怎么请求数据呢

tonygo 评论了 用RxSwift仿写知乎日报...

mark

机智的小黑 评论了 【精品教程】Cocos2d-x v3.6制作射箭游...

相关帖子

我这是要失业的节奏？新需求跟个手游差不多！

我的企业证书账号,听说现在很值钱？

鸡急急急急鸡，关于ios9之后的通讯删除联系人的问题，求大神

想接个项目,请教一下价格问题！

[AVAudioSession
setActive:withOptions:error]:
Deactivating an audio session

在线问诊APP开发主要功能

急急急！安装 Ruby 环境 失败

一站式新闻抓取工具免费内容分发平台
鲜闻

如何在一个view中添加第二个
scrollview

```
<Text style={[styles.text, styles.header]}>
  水平居中
</Text>

<View style={{height: 100, backgroundColor: '#333333', alignItems:
'center'}}>
  <View style={{backgroundColor: '#fefefe', width: 30, height: 30,
borderRadius: 15}}/>
</View>

<Text style={[styles.text, styles.header]}>
  垂直居中
</Text>
<View style={{height: 100, backgroundColor: '#333333',
justifyContent: 'center'}}>
  <View style={{backgroundColor: '#fefefe', width: 30, height: 30,
borderRadius: 15}}/>
</View>

<Text style={[styles.text, styles.header]}>
  水平垂直居中
</Text>
<View style={{height: 100, backgroundColor: '#333333', alignItems:
'center', justifyContent: 'center'}}>
  <View style={{backgroundColor: '#fefefe', width: 30, height: 30,
borderRadius: 15}}/>
</View>
```

水平居中



垂直居中



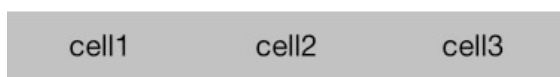
水平垂直居中



网格布局

网格布局实验， 网格布局能够满足绝大多数的日常开发需求，所以只要满足网格布局的spec，那么就可以证明react的flex布局能够满足正常开发需求

等分的网格



微博



CocoaChina

加关注

【iOS开发之Runtime常用示例总结】
<http://t.cn/RiVYgrZ> 本篇博客所聊的Runtime的内容大概有：动态获取类名、动态获取类的成员变量、动态获取类的属性列表、动态获取类的方法列表、动态获取类所遵循的协议列表、动态添加新的方法、类的实例方法实现的交换、动态属性关联、消息发送与消息转发机制等。



今天 09:33

转发 | 评论

【程序员「奇葩」说】<http://t.cn/Ri5eYPD> 「奇葩说」里的达人都是能说会



```
<View style={styles.flexContainer}>
  <View style={styles.cell}>
    <Text style={styles.welcome}>
      cell1
    </Text>
  </View>
  <View style={styles.cell}>
    <Text style={styles.welcome}>
      cell2
    </Text>
  </View>
  <View style={styles.cell}>
    <Text style={styles.welcome}>
      cell3
    </Text>
  </View>
</View>

styles = {
  flexContainer: {
    // 容器需要添加direction才能变成让子元素flex
    flexDirection: 'row'
  },
  cell: {
    flex: 1,
    height: 50,
    backgroundColor: '#aaaaaa'
  },
  welcome: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10
  },
}
```

左边固定，右边固定，中间flex的布局



```
<View style={styles.flexContainer}>
  <View style={styles.cellfixed}>
    <Text style={styles.welcome}>
      fixed
    </Text>
  </View>
  <View style={styles.cell}>
    <Text style={styles.welcome}>
      flex
    </Text>
  </View>
  <View style={styles.cellfixed}>
    <Text style={styles.welcome}>
      fixed
    </Text>
  </View>
</View>

styles = {
  flexContainer: {
    // 容器需要添加direction才能变成让子元素flex
    flexDirection: 'row'
  },
  cell: {
    flex: 1,
    height: 50,
    backgroundColor: '#aaaaaa'
  },
  welcome: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10
  },
  cellfixed: {
    height: 50,
    width: 80,
    backgroundColor: '#fefefe'
  },
}
```

嵌套的网格

通常网格不是一层的，布局容器都是一层套一层的，所以必须验证在real world下面的网格布局

```
<Text style={[styles.text, styles.header]}>
  嵌套的网格
</Text>
<View style={{flexDirection: 'row', height: 200,
  backgroundColor: '#fefefe', padding: 20}}>
  <View style={{flex: 1, flexDirection: 'column', padding: 15,
    backgroundColor: '#eeeeee'}}>
    <View style={{flex: 1, backgroundColor: '#bbaaaa'}}>
    </View>
    <View style={{flex: 1, backgroundColor: '#aabbba'}}>
    </View>
  </View>
  <View style={{flex: 1, padding: 15, flexDirection: 'row',
    backgroundColor: '#eeeeee'}}>
    <View style={{flex: 1, backgroundColor: '#aaaabb'}}>
      <View style={{flex: 1, flexDirection: 'row',
        backgroundColor: '#eeeeee'}}>
        <View style={{flex: 1, backgroundColor: '#eebbaa'}}>
        </View>
        <View style={{flex: 1, backgroundColor: '#bbccee'}}>
        </View>
      </View>
      <View style={{flex: 1, backgroundColor: '#eebbdd'}}>
      </View>
    </View>
    <View style={{flex: 1, backgroundColor: '#aaccba'}}>
    <ScrollView style={{flex: 1, backgroundColor: '#bbccdd',
      padding: 5}}>
      <View style={{flexDirection: 'row', height: 50,
        backgroundColor: '#fefefe'}}>
        <View style={{flex: 1, flexDirection: 'column',
          backgroundColor: '#eeeeee'}}>
          <View style={{flex: 1, backgroundColor: '#bbaaaa'}}>
          </View>
          <View style={{flex: 1, backgroundColor: '#aabbba'}}>
          </View>
        </View>
        <View style={{flex: 1, flexDirection: 'row',
          backgroundColor: '#eeeeee'}}>
          <View style={{flex: 1, backgroundColor: '#aaaabb'}}>
          </View>
        </View>
      </View>
    </View>
  </View>
</View>
```

```

    <View style={{flex: 1, flexDirection: 'row',
      backgroundColor: '#eeeeee'}}>
      <View style={{flex: 1,
        backgroundColor: '#eebbaa'}}>
      </View>
      <View style={{flex: 1,
        backgroundColor: '#bbccee'}}>
      </View>
    </View>
    <View style={{flex: 1,
      backgroundColor: '#eebbdd'}}>
    </View>
  </View>
  <View style={{flex: 1, backgroundColor: '#aaccba'}}>
  </View>
</View>
<Text style={[styles.text, styles.header, {color:
  'ffffff', fontSize: 12}]}>
  {(function(){
    var str = '';
    var n = 100;
    while(n--){
      str += '嵌套的网格' + '\n';
    }
    return str;
  })()}
</Text>
</ScrollView>
</View>
</View>
</View>
```

嵌套的网格



好在没被我玩儿坏，可以看到上图的嵌套关系也是足够的复杂的，（我还加了一个ScrollView，然后再嵌套整个结构）嵌套多层的布局是没有问题的。

图片布局

首先我们得知道图片有一个stretchMode. 通过Image.resizeMode访问

找出有哪些mode

```
var keys = Object.keys(Image.resizeMode).join(' ');
```

尝试使用这些mode

```
<Text style={styles.welcome}> 100px height </Text>
<Image style={{height: 100}} source={{uri: 'http://gtms03.alicdn.
com/tps/i3/TB1Kcs5GXXXXbMXVXXutsrNFFX-608-370.png'}} />
```



100px 高度，可以看到图片适应100高度和全屏宽度，背景居中适应未拉伸但是被截断也就是cover。

```
<Text style={styles.welcome}> 100px height with resizeMode contain
</Text>
<View style={{flex: 1, backgroundColor: '#fe0000'}}>
  <Image style={{flex: 1, height: 100, resizeMode: Image.resizeMode.
    contain}} source={{uri: 'http://gtms03.alicdn.
    com/tps/i3/TB1Kcs5GXXXXbMXVXXutsrNFFX-608-370.png'}} />
</View>
```



contain 模式容器完全容纳图片，图片自适应宽高


```
<Text style={styles.welcome}> 100px height with resizeMode cover </Text>
<View style={{flex: 1, backgroundColor: '#fe0000'}}>
  <Image style={{flex: 1, height: 100, resizeMode: Image.resizeMode.
    cover}} source={{uri: 'http://gtms03.alicdn.
    com/tps/i3/TB1Kcs5GXXXXbMXVXXutsrNFX-608-370.png'}} />
</View>
```

100px height with resizeMode cover



cover模式同100px高度模式

```
<Text style={styles.welcome}> 100px height with resizeMode stretch
</Text>
<View style={{flex: 1, backgroundColor: '#fe0000'}}>
  <Image style={{flex: 1, height: 100, resizeMode: Image.resizeMode.
    stretch}} source={{uri: 'http://gtms03.alicdn.
    com/tps/i3/TB1Kcs5GXXXXbMXVXXutsrNFX-608-370.png'}} />
</View>
```

100px height with resizeMode stretch



stretch模式图片被拉伸适应屏幕

```
<Text style={styles.welcome}> set height to image container </Text>
<View style={{flex: 1, backgroundColor: '#fe0000', height: 100}}>
  <Image style={{flex: 1}} source={{uri: 'http://gtms03.alicdn.
    com/tps/i3/TB1Kcs5GXXXXbMXVXXutsrNFX-608-370.png'}} />
</View>
```

set height to image container



随便试验了一下，发现高度设置到父容器，图片flex的时候也会等同于cover模式

绝对定位和相对定位

```
<View style={{flex: 1, height: 100, backgroundColor: '#333333'}}>
  <View style={[styles.circle, {position: 'absolute', top: 50, left:
    180}]}>
  </View>
</View>
styles = {
  circle: {
    backgroundColor: '#fe0000',
    borderRadius: 10,
    width: 20,
    height: 20
  }
}
```



和css的标准不同的是，元素容器不用设置position: 'absolute|relative'.

```
<View style={{flex: 1, height: 100, backgroundColor: '#333333'}}>
  <View style={[styles.circle,
    {position: 'relative', top: 50, left: 50, marginLeft: 50}]}>
  </View>
</View>
```



相对定位的可以看到很容易的配合margin做到了。（我还担心不能配合margin，所以测试了一下：-：）

padding和margin

我们知道在css中区分inline元素和block元素，既然react-native实现了一个超级小的css subset。那我们就来实验一下padding和margin在inline和非inline元素上的padding和margin的使用情况。

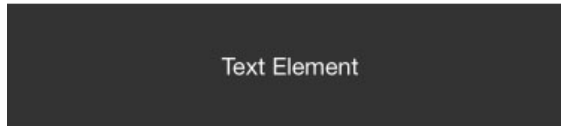
padding

```
<Text style={[styles.text, styles.header]}>
  在正常的View上设置padding
</Text>

<View style={{padding: 30, backgroundColor: '#333333'}}>
  <Text style={[styles.text, {color: '#fefefe'}]}> Text Element</Text>
</View>

<Text style={[styles.text, styles.header]}>
  在文本元素上设置padding
</Text>
<View style={{padding: 0, backgroundColor: '#333333'}}>
  <Text style={[styles.text, {backgroundColor: '#fe0000', padding: 30}]}>
  >
    text 元素上设置paddinga
  </Text>
</View>
```


在正常的View上设置padding



在文本元素上设置padding



在View上设置padding很顺利，没有任何问题，但是如果在inline元素上设置padding，发现会出现上面的错误，paddingTop和paddingBottom都被挤成marginBottom了。按理说，不应该对Text做padding处理，但是确实有这样的问题存在，所以可以将这个问题mark一下。

margin

```
<Text style={[styles.text, styles.header]}>
  在正常的View上设置margin
</Text>

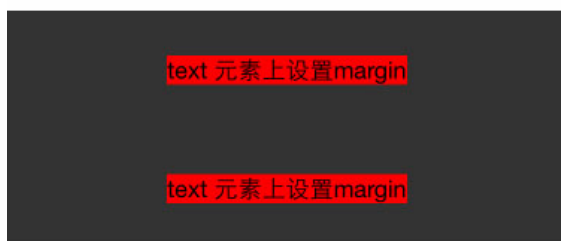
<View style={{backgroundColor: '#333333'}}>
  <View style={{backgroundColor: '#fefefe', width: 30, height: 30, margin: 30}}/>
</View>

<Text style={[styles.text, styles.header]}>
  在文本元素上设置margin
</Text>
<View style={{backgroundColor: '#333333'}}>
  <Text style={[styles.text, {backgroundColor: '#fe0000', margin: 30}]}>
    text 元素上设置margin
  </Text>
  <Text style={[styles.text, {backgroundColor: '#fe0000', margin: 30}]}>
    text 元素上设置margin
  </Text>
</View>
```

在正常的View上设置margin



在文本元素上设置margin



我们知道，对于inline元素，设置margin-left和margin-right有效，top和bottom按理是不会生效的，但是上图的结果可以看到，实际是生效了的。所以现在给我的感觉是Text元素更应该理解为一个不能设置padding的block。

算了不要猜了，我们看看[官方文档怎么说Text](#)

```
<Text>
  <Text>First part and </Text>
  <Text>second part</Text>
</Text>
// Text container: all the text flows as if it was one
// |First part |
// |and second |
// |part      |

<View>
  <Text>First part and </Text>
  <Text>second part</Text>
</View>
// View container: each text is its own block
// |First part |
// |and       |
// |second part|
```

也就是如果Text元素在Text里边，可以考虑为inline，如果单独在View里边，那就是Block。

下面会专门研究一下文本相关的布局

文本元素

首先我们得考虑对于Text元素我们希望有哪些功能或者想验证哪些功能：

1. 文字是否能自动换行？
2. overflow ellipse?
3. 是否能对部分文字设置样式，类似span等标签

先看看文字有哪些支持的**style**属性

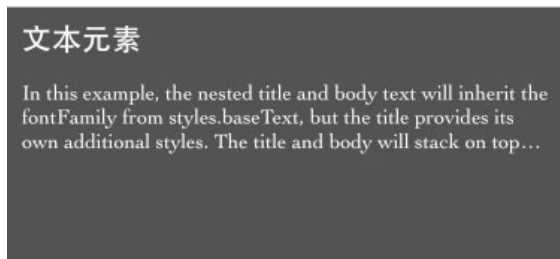
```
/*=====TEXT=====*/
Attributes.style = {
  color string
  containerBackgroundColor string
  fontFamily string
  fontSize number
  fontStyle enum('normal', 'italic')
  fontWeight enum("normal", 'bold', '100', '200', '300', '400', '500',
    '600', '700', '800', '900')
  lineHeight number
  textAlign enum("auto", 'left', 'right', 'center')
  writingDirection enum("auto", 'ltr', 'rtl')
}
```

实验1, 2, 3

```
<Text style={{styles.text, styles.header}}>
  文本元素
</Text>

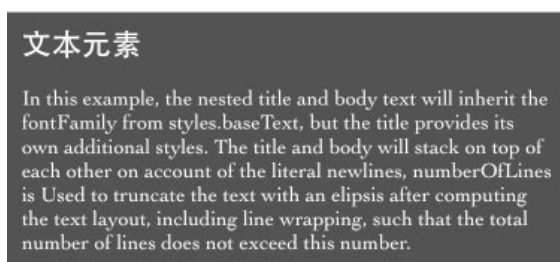
<View style={{backgroundColor: '#333333', padding: 10}}>
  <Text style={styles.baseText} numberOfLines={5}>
    <Text style={styles.titleText} onPress={this.onPressTitle}>
      文本元素{'\n'}
    </Text>
    <Text>
      {'\n'}In this example, the nested title and body text will
      inherit the fontFamily from styles.baseText, but the title
      provides its own additional styles. The title and body will
      stack on top of each other on account of the literal newlines,
      numberOfLines is Used to truncate the text with an elipsis
      after computing the text layout, including line wrapping, such
      that the total number of lines does not exceed this number.
    </Text>
  </Text>
</View>
styles = {
  baseText: {
    fontFamily: 'Cochin',
    color: 'white'
  },
  titleText: {
    fontSize: 20,
    fontWeight: 'bold',
  }
}
```

文本元素



从结果来看1, 2, 3得到验证。但是不知道各位有没有发现问题，为什么底部空出了这么多空间，没有设置高度啊。我去掉numberOfLines={5} 这行代码，效果如下：

文本元素



所以实际上，那段空间是文本撑开的，但是文本被numberOfLines={5} 截取了，但是剩余的空间还在。我猜这应该是个bug。

其实官方文档里边把numberOfLines={5}这句放到的是长文本的Text元素上的，也就是子Text上的。实际结果是不生效。这应该又是一个bug。

Text元素的子Text元素的具体实现是怎样的，感觉这会有很多bug，看官方文

```
<Text style={{fontWeight: 'bold'}}>
  I am bold
  <Text style={{color: 'red'}}>
    and red
  </Text>
</Text>
```

Behind the scenes, this is going to be converted to a flat

NSAttributedString that contains the following information

```
"I am bold and red"
0-9: bold
9-17: bold, red
```

好吧，那对于numberOfLines={5} 放在子Text元素上的那种bug倒是可以解释了。

Text的样式继承

实际上React-native里边是没有样式继承这种说法的，但是对于Text元素里边的Text元素，上面的例子可以看出存在继承。那既然有继承，问题就来了！

到底是继承的最外层的Text的值呢，还是继承父亲Text的值呢？

```
<Text style={[styles.text, styles.header]}>
  文本样式继承
</Text>

<View style={{backgroundColor: '#333333', padding: 10}}>
  <Text style={{color: 'white'}}>
    <Text style={{color: 'red'}} onPress={this.onPressTitle}>
      文本元素{'\n'}
    <Text>我是white还是red呢? {'\n'} </Text>
    </Text>
    <Text>我应该是white的</Text>
  </Text>
</View>
```

文本样式继承

```
文本元素
我是white还是red呢?
我应该是white的
```

结果可见是直接继承父亲Text的。

总结

1. react 宽度基于pt为单位，可以通过Dimensions 来获取宽高，PixelRatio 获取密度，如果想使用百分比，可以通过获取屏幕宽度手动计算。
2. 基于flex的布局
 1. view默认宽度为100%
 2. 水平居中用alignItems, 垂直居中用justifyContent
 3. 基于flex能够实现现有的网格系统需求，且网格能够各种嵌套无bug
3. 图片布局
 1. 通过Image.resizeMode来适配图片布局，包括contain, cover, stretch
 2. 默认不设置模式等于cover模式

3. contain模式自适应宽高，给出高度值即可

4. cover铺满容器，但是会做截取

5. stretch铺满容器，拉伸

4. 定位

1. 定位相对于父元素，父元素不用设置position也行

2. padding 设置在Text元素上时会存在bug。所有padding变成了marginBottom

5. 文本元素

1. 文字必须放在Text元素里边

2. Text元素可以相互嵌套，且存在样式继承关系

3. numberOfLines 需要放在最外层的Text元素上，且虽然截取了文字但是还是会占用空间



微信扫一扫

订阅每日移动开发及APP推广热点资讯

公众号：CocoaChina

我要投稿

收藏文章

分享到：

29

上一篇：源码篇：MBProgressHUD

下一篇：源码推荐(4.20):图片流动显示 点击可放大，扫描二维码 仿微信效果 带有扫描条

相关资讯

最好用的 iOS 快速布局UI库

iOS自动布局框架-Masonry详解

关于如何写UI及屏幕适配的一些技巧（上）

Xcode 8 Auto Layout新手体验

手把手教你Masonry的理解

源码推荐：仿《百思不得姐》项目 简易的瀑布流布局

布局万花筒：UICollectionView

UICollectionView详解之自定义布局

UIButton图文布局

5分钟 搞定UIButton的文本与图片的布局

广告

我来说两句



您还没有登录！请 [登录](#) 或 [注册](#)


发表评论

所有评论 (0)

[关于我们](#) [商务合作](#) [联系我们](#) [合作伙伴](#)

北京触控科技有限公司版权所有

©2016 Chukong Technologies, Inc.

京ICP备 11006519号 京ICP证 100954号 京公网安备11010502020289  京网文[2012]0426-138号