1. 富文本所用技术: 2

2. 富文本简单使用: 3

3. AttributedString的21个属性 5

4. 常用属性举例说明及该属性的注意事项 6

5. 高级富文本类库RTLabel 13

# 1. 富文本所用技术:

类AttributedString和类NSMutableAttributedString

与NSString类似，在iOS中AttributedString也分为NSAttributedString和NSMutableAttributedString，不同的是，AttributedString对象多了一个Attribute的概念，一个AttributedString的对象包含很多的属性，每一个属性都有其对应的字符区域，在这里是使用NSRange来进行描述的。如

[aStr addAttribute: NSForegroundColorAttributeName value: [UIColor redColor] range: NSMakeRange(4, 3)];

设置字符串aStr从第4个字符开始的后3个字符的区域字体颜色为红色

# 2. 富文本简单使用:

// 1.拼接要显示的字符串

NSString \*str1 = @"haha";

NSString \*str2 = @"solong";

NSString \*str3 = @"图样图森破";

NSString \*originStr = [NSString stringWithFormat:@"%@%@%@", str1, str2,str3];

// 2.创建可变的AttributedString

NSMutableAttributedString\* aStr = [[NSMutableAttributedString alloc] initWithString:originStr];

// 以下方法每次只能设置一个属性,可用于设置整体的属性

// [aStr addAttribute: NSFontAttributeName value: [UIFont fontWithName: @"Zapfino" size: 15]

// range: NSMakeRange(0, originStr.length)];

// 3.1 设置str1的属性

NSDictionary \*dic = @{ // 创建一个字典存放要设置的属性及值

NSStrikethroughStyleAttributeName:@(NSUnderlineStyleSingle), // 删除线名称,默认无删除线(枚举类型是整型,要转成numble类型)

NSStrikethroughColorAttributeName: [UIColor blueColor], // 删除线颜色,默认与字体颜色一样

NSFontAttributeName: [UIFont systemFontOfSize:20], // 字体大小

NSForegroundColorAttributeName:[UIColor blackColor] // 字体颜色

};

[aStr addAttributes:dic range:[originStr rangeOfString:str1]]; // str1范围内的字符设为dic中的属性

// 3.2 设置str2的属性

dic = @{

NSUnderlineStyleAttributeName:@(NSUnderlineStyleDouble), // 下划线名称,默认无下划线

NSFontAttributeName: [UIFont systemFontOfSize:30],

NSForegroundColorAttributeName:[UIColor purpleColor]

};

[aStr addAttributes:dic range:[originStr rangeOfString:str2]];

// 3.3 设置str3的属性

dic = @{

NSFontAttributeName: [UIFont systemFontOfSize:40],

NSForegroundColorAttributeName:[UIColor orangeColor],

NSLinkAttributeName:[NSURL URLWithString:@"http://www.baidu.com"]

};

[aStr addAttributes:dic range:[originStr rangeOfString:str3]];

// 4. 创建字符串要显示到的控件

UITextView \*richText2 = [[UITextView alloc] initWithFrame:CGRectMake(10, 200, 300, 200)];

richText2.attributedText = aStr;

richText2.userInteractionEnabled = YES;

richText2.editable = NO; // 一定要关了编辑属性,否则超链接不起作用

richText2.delegate = self; // 实现代理方法,允许URL跳转,否则超链接不起作用.

//- (BOOL)textView:(UITextView \*)textView shouldInteractWithURL:(NSURL \*)url inRange:(NSRange)characterRange

// 5. 把显示控件增加到控制器中

[self.view addSubview:richText2];

注意:如果使用超链接功能,显示控件不能为UILabel

# 3. AttributedString的21个属性

NSFontAttributeName           设置字体属性，默认值：字体：Helvetica(Neue) 字号：12

NSForegroundColorAttributeNam    设置字体颜色，取值为 UIColor对象，默认值为黑色

NSBackgroundColorAttributeName   设置字体所在区域背景颜色，取值为 UIColor对象，默认值为nil, 透明色

NSLigatureAttributeName    设置连体属性，取值为NSNumber 对象(整数)，0 表示没有连体字符，1 表示使用默认的连体字符

NSKernAttributeName      设定字符间距，取值为 NSNumber 对象（整数），正值间距加宽，负值间距变窄

NSStrikethroughStyleAttributeName  设置删除线，取值为 NSNumber 对象（整数）

NSStrikethroughColorAttributeName  设置删除线颜色，取值为 UIColor 对象，默认值为黑色

NSUnderlineStyleAttributeName      设置下划线，取值为 NSNumber 对象（整数），枚举常量 NSUnderlineStyle中的值，与删除线类似

NSUnderlineColorAttributeName      设置下划线颜色，取值为 UIColor 对象，默认值为黑色

NSStrokeWidthAttributeName  设置笔画宽度，取值为 NSNumber 对象（整数），负值填充效果，正值中空效果

NSStrokeColorAttributeName    填充部分颜色，不是字体颜色，取值为 UIColor 对象

NSShadowAttributeName      设置阴影属性，取值为 NSShadow 对象

NSTextEffectAttributeName    设置文本特殊效果，取值为 NSString 对象，目前只有图版印刷效果可用：

NSBaselineOffsetAttributeName      设置基线偏移值，取值为 NSNumber （float）,正值上偏，负值下偏

NSObliquenessAttributeName         设置字形倾斜度，取值为 NSNumber （float）,正值右倾，负值左倾

NSExpansionAttributeName           设置文本横向拉伸属性，取值为 NSNumber （float）,正值横向拉伸文本，负值横向压缩文本

NSWritingDirectionAttributeName    设置文字书写方向，从左向右书写或者从右向左书写

NSVerticalGlyphFormAttributeName   设置文字排版方向，取值为 NSNumber 对象(整数)，0 表示横排文本，1 表示竖排文本

NSLinkAttributeName                设置链接属性，点击后调用[浏览器](http://www.2cto.com/os/liulanqi/" \t "_blank)打开指定URL地址

NSAttachmentAttributeName          设置文本附件,取值为NSTextAttachment对象,常用于文字图片混排

NSParagraphStyleAttributeName      设置文本段落排版格式，取值为 NSParagraphStyle 对象

# 4. 常用属性举例说明及该属性的注意事项

1. NSFontAttributeName

//NSForegroundColorAttributeName 设置字体颜色，取值为 UIColor，默认为黑色

NSDictionary \*attrDict1 = @{ NSForegroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict2 = @{ NSForegroundColorAttributeName: [UIColor blueColor] };

NSDictionary \*attrDict3 = @{ NSForegroundColorAttributeName: [UIColor orangeColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

注意：

    NSForegroundColorAttributeName设置的颜色与UILabel的textColor属性设置的颜色在地位上是相等的，谁最后赋值，最终显示的就是谁的颜色。

2. NSBackgroundColorAttributeName

//NSForegroundColorAttributeName 设置字体颜色，取值为 UIColor，默认为黑色

NSDictionary \*attrDict1 = @{ NSForegroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict2 = @{ NSForegroundColorAttributeName: [UIColor blueColor] };

NSDictionary \*attrDict3 = @{ NSForegroundColorAttributeName: [UIColor orangeColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

//NSBackgroundColorAttributeName 设置字体所在区域背景的颜色，取值为UIColor，默认值为nil

NSDictionary \*attrDict4 = @{ NSBackgroundColorAttributeName: [UIColor orangeColor] };

NSDictionary \*attrDict5 = @{ NSBackgroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict6 = @{ NSBackgroundColorAttributeName: [UIColor cyanColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict4];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict5];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict6];

仔细观察会发现个问题，我并没有关闭 NSForegroundColorAttributeName 属性，但是在运行结果中，所有字体的颜色都变成了默认色——黑色，这说明 NSForegroundColorAttributeName 和 NSBackgroundColorAttributeName 的低位是相等的，跟前面介绍的 textColor 一样，哪个属性最后一次赋值，就会冲掉前面的效果，若是我们把属性代码顺序交换一下

//NSBackgroundColorAttributeName 设置字体所在区域背景的颜色，取值为UIColor，默认值为nil

NSDictionary \*attrDict4 = @{ NSBackgroundColorAttributeName: [UIColor orangeColor] };

NSDictionary \*attrDict5 = @{ NSBackgroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict6 = @{ NSBackgroundColorAttributeName: [UIColor cyanColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict4];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict5];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict6];

//NSForegroundColorAttributeName 设置字体颜色，取值为 UIColor，默认为黑色

NSDictionary \*attrDict1 = @{ NSForegroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict2 = @{ NSForegroundColorAttributeName: [UIColor blueColor] };

NSDictionary \*attrDict3 = @{ NSForegroundColorAttributeName: [UIColor orangeColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

但是textColor属性可以与 NSBackgroundColorAttributeName 属性叠加

\_label01.textColor = [UIColor greenColor];

\_label02.textColor = [UIColor yellowColor];

\_label03.textColor = [UIColor blueColor];

//NSForegroundColorAttributeName 设置字体颜色，取值为 UIColor，默认为黑色

NSDictionary \*attrDict1 = @{ NSForegroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict2 = @{ NSForegroundColorAttributeName: [UIColor blueColor] };

NSDictionary \*attrDict3 = @{ NSForegroundColorAttributeName: [UIColor orangeColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

//NSBackgroundColorAttributeName 设置字体所在区域背景的颜色，取值为UIColor，默认值为nil

NSDictionary \*attrDict4 = @{ NSBackgroundColorAttributeName: [UIColor orangeColor] };

NSDictionary \*attrDict5 = @{ NSBackgroundColorAttributeName: [UIColor redColor] };

NSDictionary \*attrDict6 = @{ NSBackgroundColorAttributeName: [UIColor cyanColor] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict4];

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict5];

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict6];

虽然 textColor 在 NSFontAttributeName 之前赋值，但是由于 NSFontAttributeName 的属性效果被NSBackgroundColorAttributeName 属性冲掉了，所以最终显示了 textColor 的颜色。

3. NSLigatureAttributeName

//NSLigatureAttributeName 设置连体属性，取值为NSNumber 对象(整数)，0 表示没有连体字符，1 表示使用默认的连体字符，

//                        2 表示使用所有连体符号，默认值为 1（iOS 不支持 2）

NSString \*ligatureStr = @"flush";

NSDictionary \*attrDict1 = @{ NSLigatureAttributeName: [NSNumber numberWithInt: 0],

                             NSFontAttributeName: [UIFont fontWithName: @"futura" size: 30] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: ligatureStr attributes: attrDict1];

NSDictionary \*attrDict2 = @{ NSLigatureAttributeName: @(1),

                             NSFontAttributeName: [UIFont fontWithName: @"futura" size: 30]

                             };

\_label02.attributedText = [[NSAttributedString alloc] initWithString: ligatureStr attributes: attrDict2];

       由于要展示连体字符，所以将前面使用的带有中文的字符串换成 flush

       NSLigatureAttributeName的取值为NSNumber对象，所以不能直接将一个整数值赋给它，创建 NSNumber 对象的方法有很多，或者可以简写成 @(int)

注意观察字母f和l之间的变化。

感觉连写就是一个艺术字功能，当字符f和l组合使用组合符号（所谓的字形(glyph)）绘制时，看起来确实更加美观。但是并非所有的字符之间都有组合符号，事实上，只有某些字体中得某些字符的组合（如字符f和l，字符f和i等）才具有美观的组合符号。

4. NSKernAttributeName

//NSKernAttributeName 设定字符间距，取值为 NSNumber 对象（整数），正值间距加宽，负值间距变窄

  NSDictionary \*attrDict1 = @{ NSKernAttributeName: @(-3),

                               NSFontAttributeName: [UIFont systemFontOfSize: 20]

                               };

  \_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

  NSDictionary \*attrDict2 = @{ NSKernAttributeName: @(0),

                               NSFontAttributeName: [UIFont systemFontOfSize: 20]

                               };

  \_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

  NSDictionary \*attrDict3 = @{ NSKernAttributeName: @(10),

                               NSFontAttributeName: [UIFont systemFontOfSize: 20]

                               };

  \_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

5. NSStrikethroughStyleAttributeName

//NSStrikethroughStyleAttributeName 设置删除线，取值为 NSNumber 对象（整数），枚举常量 NSUnderlineStyle中的值

// NSUnderlineStyleNone   不设置删除线

// NSUnderlineStyleSingle 设置删除线为细单实线

// NSUnderlineStyleThick  设置删除线为粗单实线

// NSUnderlineStyleDouble 设置删除线为细双实线

NSDictionary \*attrDict1 = @{ NSStrikethroughStyleAttributeName: @(NSUnderlineStyleSingle),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

NSDictionary \*attrDict2 = @{ NSStrikethroughStyleAttributeName: @(NSUnderlineStyleThick),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

NSDictionary \*attrDict3 = @{ NSStrikethroughStyleAttributeName: @(NSUnderlineStyleDouble),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

注意：

虽然使用了枚举常量，但是枚举常量的本质仍为整数，所以同样必须先转化为 NSNumber 才能使用

删除线和下划线使用相同的枚举常量作为其属性值,目前iOS中只有上面列出的4中效果，虽然我们能够在头文件中发现其他更多的取值，但是使用后没有任何效果,可以看出，中文和英文删除线的位置有所不同,另外，删除线属性取值除了上面的4种外，其实还可以取其他整数值，有兴趣的可以自行试验，取值为 0 - 7时，效果为单实线，随着值得增加，单实线逐渐变粗，取值为 9 - 15时，效果为双实线，取值越大，双实线越粗。

NSDictionary \*attrDict1 = @{ NSStrikethroughStyleAttributeName: @(1),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

NSDictionary \*attrDict2 = @{ NSStrikethroughStyleAttributeName: @(3),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

NSDictionary \*attrDict3 = @{ NSStrikethroughStyleAttributeName: @(7),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

6. NSStrikethroughColorAttributeName

//NSStrikethroughColorAttributeName 设置删除线颜色，取值为 UIColor 对象，默认值为黑色

  NSDictionary \*attrDict1 = @{ NSStrikethroughColorAttributeName: [UIColor blueColor],

                               NSStrikethroughStyleAttributeName: @(1),

                               NSFontAttributeName: [UIFont systemFontOfSize:20] };

  \_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

  NSDictionary \*attrDict2 = @{ NSStrikethroughColorAttributeName: [UIColor orangeColor],

                               NSStrikethroughStyleAttributeName: @(3),

                               NSFontAttributeName: [UIFont systemFontOfSize:20] };

  \_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

  NSDictionary \*attrDict3 = @{ NSStrikethroughColorAttributeName: [UIColor greenColor],

                               NSStrikethroughStyleAttributeName: @(7),

                               NSFontAttributeName: [UIFont systemFontOfSize:20] };

  \_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

7. NSUnderlineStyleAttributeName

    下划线除了线条位置和删除线不同外，其他的都可以完全参照删除线设置。

//NSUnderlineStyleAttributeName 设置下划线，取值为 NSNumber 对象（整数），枚举常量 NSUnderlineStyle中的值，与删除线类似

NSDictionary \*attrDict1 = @{ NSUnderlineStyleAttributeName: @(NSUnderlineStyleSingle),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

NSDictionary \*attrDict2 = @{ NSUnderlineStyleAttributeName: @(NSUnderlineStyleThick),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

NSDictionary \*attrDict3 = @{ NSUnderlineStyleAttributeName: @(NSUnderlineStyleDouble),

                             NSFontAttributeName: [UIFont systemFontOfSize:20] };

\_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

8. NSUnderlineColorAttributeName

    可以完全参照下划线颜色设置

//NSUnderlineColorAttributeName 设置下划线颜色，取值为 UIColor 对象，默认值为黑色

   NSDictionary \*attrDict1 = @{ NSUnderlineColorAttributeName: [UIColor blueColor],

                                NSUnderlineStyleAttributeName: @(NSUnderlineStyleSingle),

                                NSFontAttributeName: [UIFont systemFontOfSize:20] };

   \_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

   NSDictionary \*attrDict2 = @{ NSUnderlineColorAttributeName: [UIColor orangeColor],

                                NSUnderlineStyleAttributeName: @(NSUnderlineStyleThick),

                                NSFontAttributeName: [UIFont systemFontOfSize:20] };

   \_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

   NSDictionary \*attrDict3 = @{ NSUnderlineColorAttributeName: [UIColor greenColor],

                                NSUnderlineStyleAttributeName: @(NSUnderlineStyleDouble),

                                NSFontAttributeName: [UIFont systemFontOfSize:20] };

   \_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

9. NSStrokeWidthAttributeName

//NSStrokeWidthAttributeName 设置笔画宽度，取值为 NSNumber 对象（整数），负值填充效果，正值中空效果

  NSDictionary \*attrDict1 = @{ NSStrokeWidthAttributeName: @(-3),

                               NSFontAttributeName: [UIFont systemFontOfSize:30] };

  \_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

  NSDictionary \*attrDict2 = @{ NSStrokeWidthAttributeName: @(0),

                               NSFontAttributeName: [UIFont systemFontOfSize:30] };

  \_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

  NSDictionary \*attrDict3 = @{ NSStrokeWidthAttributeName: @(3),

                               NSFontAttributeName: [UIFont systemFontOfSize:30] };

  \_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

10. NSStrokeColorAttributeName

//NSStrokeColorAttributeName 填充部分颜色，不是字体颜色，取值为 UIColor 对象

   NSDictionary \*attrDict1 = @{ NSStrokeWidthAttributeName: @(-3),

                                NSStrokeColorAttributeName: [UIColor orangeColor],

                                NSFontAttributeName: [UIFont systemFontOfSize:30] };

   \_label01.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict1];

   NSDictionary \*attrDict2 = @{ NSStrokeWidthAttributeName: @(0),

                                NSStrokeColorAttributeName: [UIColor blueColor],

                                NSFontAttributeName: [UIFont systemFontOfSize:30] };

   \_label02.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict2];

   NSDictionary \*attrDict3 = @{ NSStrokeWidthAttributeName: @(3),

                                NSStrokeColorAttributeName: [UIColor greenColor],

                                NSFontAttributeName: [UIFont systemFontOfSize:30] };

   \_label03.attributedText = [[NSAttributedString alloc] initWithString: originStr attributes: attrDict3];

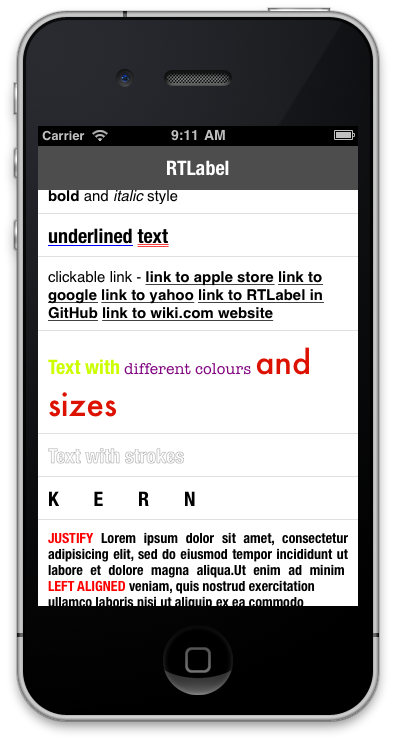
# 5. 高级富文本类库RTLabel

(摘自: <http://blog.csdn.net/duxinfeng2010/article/details/9004749>)

适合做过网页开发,懂html的同学研究

# RTLabel

基于富文本的格式，适用于iOS，类似HTML的标记。



RTLabel 基于UILabel类的拓展,能够支持Html标记的富文本显示，它是基于Core Text,因此也支持Core Text上的一些东西。

# 特点

##### 粗体和斜体

##### 颜色和大小

##### 下划线

##### 缩进

##### 字距

##### 行距

##### 超链接

# 

# 用法

1）将RTLabel.h 和 RTLabel.m 拖拽道你的工程中，导入Core Text.framework。添加头文件

**[cpp]** [view plaincopy](http://blog.csdn.net/duxinfeng2010/article/details/9004749)

1. #import "RTLabel.h"

2)创建一个RTLabel

**[cpp]** [view plaincopy](http://blog.csdn.net/duxinfeng2010/article/details/9004749)

1. NSString \*sample\_text = @"<b>bold</b>,<i>italic</i> and <u>underlined</u> text, and <font face='HelveticaNeue-CondensedBold' size=20 color='#CCFF00'>text with custom font and color</font>";
3. RTLabel \*label = [[RTLabel alloc] initWithFrame:...];
4. [self addSubview:label];
5. [label setText:sample\_text];

3)支持以下标签

**[html]** [view plaincopy](http://blog.csdn.net/duxinfeng2010/article/details/9004749)

1. **<b>**Bold**</b>**
2. **<i>**Italic**</i>**
3. **<bi>**Bold & Italic**</bi>**
4. **<u>**underline**</u>**, **<u** color=red**>**underline with color**</u>**
5. **<a** href='http://..'**>**link**</a>**
6. **<uu>**double underline**</uu>** , **<uu** color='#ccff00'**>**double underline with color**</uu>**
7. **<font** face='HelveticaNeue-CondensedBold' size=20 color='#CCFF00'**>**custom font**</font>**
8. **<font** face='HelveticaNeue-CondensedBold' size=20 color='#CCFF00' stroke=1**>**custom font with strokes**</font>**
9. **<font** face='HelveticaNeue-CondensedBold' size=20 color='#CCFF00' kern=35**>**custom font with kerning**</font>**
10. **<p** align=justify**>**alignment**</p>**
11. **<p** indent=20**>**indentation**</p>**

# 最低要求

* ARC -RTLabel使用了ARC,如果你的工程没有使用ARC,在编译时 compiler flag 上 添加 '-fobjc-arc'(如下图显示)
* XCode 4.4 或以上新版本。

