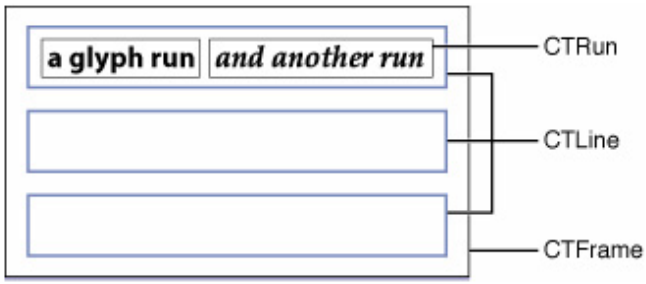
API接口文档。

https://developer.apple.com/library/mac/#documentation/Carbon/Reference/CoreText_Framework_Ref/_index.html

CoreText 框架中最常用的几个类：

- CTFont
- CTFontCollection
- CTFontDescriptor
- CTFrame
- CTFramesetter
- CTGlyphInfo
- CTLine
- CTParagraphStyle
- CTRun
- CTTextTab
- CTTypesetter

先来了解一下该框架的整体视窗组合图：



CTFrame 作为一个整体的画布(Canvas)，其中由行(CTLine)组成，而每行可以分为一个或多个小方块（CTRun）。

注意：你不需要自己创建CTRun，Core Text将根据NSAttributedString的属性来自动创建CTRun。每个CTRun对象对应不同的属性，正因此，你可以自由的控制字体、颜色、字间距等等信息。

通常处理步聚：

1.使用core text就是先有一个要显示的string，然后定义这个string每个部分的样式->attributedString - > 生成 CTFramesetter -> 得到CTFrame -> 绘制（CTFrameDraw）
其中可以更详细的设置换行方式，对齐方式，绘制区域的大小等。
2.绘制只是显示，点击事件就需要一个判断了。
CTFrame 包含了多个CTLine,并且可以得到各个line的其实位置与大小。判断点击处在不在某个line上。CTLine 又可以判断这个点(相对于ctline的坐标)处的文字范围。然后遍历这个string的所有NSTextCheckingResult，根据result的rang判断点击处在不在这个rang上，从而得到点击的链接与位置。

字体的基本知识：

字体(Font):是一系列字号、样式和磅值相同的字符(例如:10磅黑体Palatino)。现多被视为字样的同义词

字面(Face):是所有字号的磅值和格式的综合

字体集(Font family):是一组相关字体(例如:Franklin family包括Franklin Gothic、Fran-klinHeavy和Franklin Compressed)

磅值(Weight):用于描述字体粗度。典型的磅值,从最粗到最细,有极细、细、book、中等、半粗、粗、较粗、极粗

样式(Style):字形有三种形式:Roman type是直体;oblique type是斜体;utakuc type是斜体兼曲线(比Roman type更像书法体)。

x高度(X height):指小写字母的平均高度(以x为基准)。磅值相同的两字母,x高度越大的字母看起来比x高度小的字母要大

Cap高度(Cap height):与x高度相似。指大写字母的平均高度(以C为基准)

下行字母(Descender):例如在字母q中,基线以下的字母部分叫下伸部分

上行字母(Ascender):x高度以上的部分(比如字母b)叫做上伸部分

基线(Baseline):通常在x、v、b、m下的那条线

描边(Stroke):组成字符的线或曲线。可以加粗或改变字符形状

衬线(Serif):用来使字符更可视的一条水平线。如字母左上角和下部的水平线。

无衬线(Sans Serif):可以让排字员不使用衬线装饰。

方形字(Block):这种字体的笔画使字符看起来比无衬线字更显眼,但还不到常见的衬线字的程度。例如Lubalin Graph就是方形字,这种字看起来好像是木头块刻的一样

手写体脚本(Calligraphic script):是一种仿效手写体的字体。例如Murray Hill或者Fraktur字体

艺术字(Decorative):像绘画般的字体

Pi符号(Pisymbol):非标准的字母数字字符的特殊符号。例如Wingdings和Mathematical Pi

连写(Ligature):是一系列连写字母如fi、fl、ffi或ffl。由于字些字母形状的原因经常被连写,故排字员已习惯将它们连写。

1 基准线 baseline

2 x 高(小写线) x-height

3 大写线 capital line

4 上缘线 ascender line

5 下缘线 descender line

上部 ascender area

下部 descender area

字符属性名称：

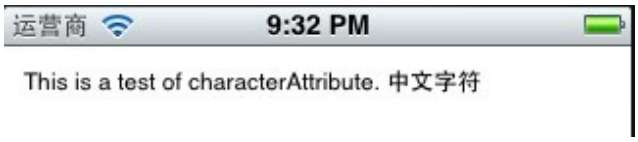
```
const CFStringRef kCTCharacterShapeAttributeName;

//字体形状属性　必须是CFNumberRef对象默认为0，非0则对应相应的字符形状定义，如1表示传统字符形状

const CFStringRef kCTFontAttributeName;

//字体属性　　必须是CTFont对象const CFStringRef kCTKernAttributeName;
```

```
//字符间隔属性 必须是CFNumberRef对象const CFStringRef kCTLigatureAttributeName;


//设置是否使用连字属性，设置为0，表示不使用连字属性。标准的英文连字有FI,FL.默认值为1，既是使用标准连字。也就是当搜索到f时候，会把fl当成一个文字。必须是CFNumberRef 默认为1,可取0,1,2const CFStringRef kCTForegroundColorAttributeName;

//字体颜色属性  必须是CGColor对象，默认为blackconst CFStringRef kCTForegroundColorFromContextAttributeName;

 //上下文的字体颜色属性 必须为CFBooleanRef 默认为False,const CFStringRef kCTParagraphStyleAttributeName;

//段落样式属性 必须是CTParagraphStyle对象 默认为NILconst CFStringRef kCTStrokeWidthAttributeName;

//笔画线条宽度 必须是CFNumberRef对象，默为0.0f，标准为3.0fconst CFStringRef kCTStrokeColorAttributeName;

//笔画的颜色属性 必须是CGColorRef 对象，默认为前景色const CFStringRef kCTSuperscriptAttributeName;

//设置字体的上下标属性 必须是CFNumberRef对象 默认为0,可为-1为下标,1为上标，需要字体支持才行。如排列组合的样式Cn1const CFStringRef kCTUnderlineColorAttributeName;


//字体下划线颜色属性 必须是CGColorRef对象，默认为前景色const CFStringRef kCTUnderlineStyleAttributeName;

//字体下划线样式属性 必须是CFNumberRef对象,默为kCTUnderlineStyleNone 可以通过CTUnderlineStypleModifiers 进行修改下划线风格const CFStringRef kCTVerticalFormsAttributeName;

//文字的字形方向属性 必须是CFBooleanRef 默认为false，false表示水平方向，true表示竖直方向const CFStringRef kCTGlyphInfoAttributeName;

//字体信息属性 必须是CTGlyphInfo对象const CFStringRef kCTRunDelegateAttributeName

//CTRun 委托属性 必须是CTRunDelegate对象
```

举例说明：

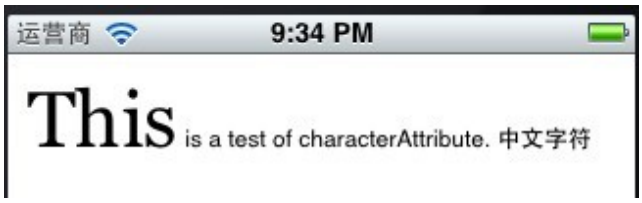
[cpp] view plain copy

```
01. NSMutableAttributedString *mabstring = [[NSMutableAttributedString alloc]initWithString:@"This is a test of characterAttribute. 中文字符"];
```



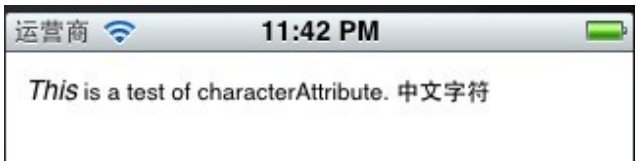
[cpp] view plain copy

```
01. //设置字体属性
02.  CTFontRef font = CTFontCreateWithName(CFSTR("Georgia"), 40, NULL);
03.  [mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];
```



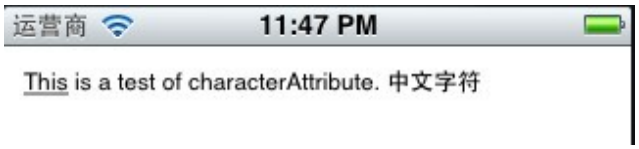
[cpp] view plain copy

```
01. //设置斜体字
02.     CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont
    italicSystemFontOfSize:20].fontName, 14, NULL);
03.     [mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];
```



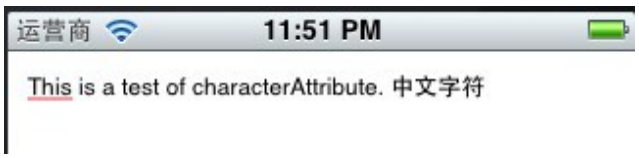
[cpp] view plain copy

```
01. //下划线
02.     [mabstring addAttribute:(id)kCTUnderlineStyleAttributeName value:(id)[NSNumber
    numberWithInt:kCTUnderlineStyleDouble] range:NSMakeRange(0, 4)];
```



[cpp] view plain copy

```
01. //下划线颜色
02.     [mabstring addAttribute:(id)kCTUnderlineColorAttributeName value:(id)[UIColor redColor].CGColor
    range:NSMakeRange(0, 4)];
```



[cpp] view plain copy

```
01. //设置字体简隔 eg:test
02. long number = 10;
03.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
04.     [mabstring addAttribute:(id)kCTKernAttributeName value:(id)num range:NSMakeRange(10, 4)];
```

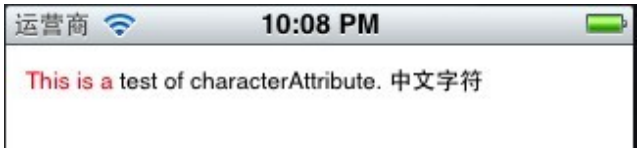
[cpp] view plain copy

```
01. //设置连字
02. long number = 1;
03.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
04.     [mabstring addAttribute:(id)kCTLigatureAttributeName value:(id)num range:NSMakeRange(0, [str
    length])];
```

连字还不会使用，未看到效果。

[cpp] view plain copy

```
01. //设置字体颜色
02.     [mabstring addAttribute:(id)kCTForegroundColorAttributeName value:(id)[UIColor redColor].CGColor
    range:NSMakeRange(0, 9)];
```



[cpp] view plain copy

```
01. //设置字体颜色为前影色
02.     CFBooleanRef flag = kCFBooleanTrue;
03.     [mabstring addAttribute:(id)kCTForegroundColorFromContextAttributeName value:(id)flag
    range:NSMakeRange(5, 10)];
```

无明显效果。

[cpp] view plain copy

```
01. //设置空心字
02. long number = 2;
03.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
04.     [mabstring addAttribute:(id)kCTStrokeWidthAttributeName value:(id)num range:NSMakeRange(0, [str
    length])];
```

[cpp] view plain copy

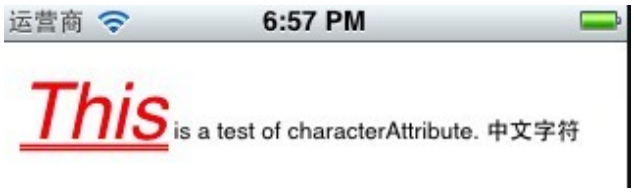
```
01. //设置空心字
02. long number = 2;
03.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
04.     [mabstring addAttribute:(id)kCTStrokeWidthAttributeName value:(id)num range:NSMakeRange(0, [str
    length])];
05.
06. //设置空心字颜色
07.     [mabstring addAttribute:(id)kCTStrokeColorAttributeName value:(id)[UIColor greenColor].CGColor
    range:NSMakeRange(0, [str length])];
```

在设置空心字颜色时，必须先将字体高为空心，否则设置颜色是没有效果的。

[cpp] view plain copy

```
01. //对同一段字体进行多属性设置
02. //红色
03.     NSMutableDictionary *attributes = [NSMutableDictionary dictionaryWithObject:(id)[UIColor
    redColor].CGColor forKey:(id)kCTForegroundColorAttributeName];
04. //斜体
05.     CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont
    italicSystemFontOfSize:20].fontName, 40, NULL);
06.     [attributes setObject:(id)font forKey:(id)kCTFontAttributeName];
07. //下划线
```

```
08.     [attributes setObject:(id)[NSNumber numberWithInt:kCTUnderlineStyleDouble] forKey:
    (id)kCTUnderlineStyleAttributeName];
09.
10.     [mabstring addAttributes:attributes range:NSMakeRange(0, 4)];
```



最后是draw了。

[cpp] view plain copy

```
01. -(void)characterAttribute
02. {
03.     NSString *str = @"This is a test of characterAttribute. 中文字符";
04.     NSMutableAttributedString *mabstring = [[NSMutableAttributedString alloc]initWithString:str];
05.
06.     [mabstring beginEditing];
07. /*
08.     long number = 1;
09.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
10.     [mabstring addAttribute:(id)kCTCharacterShapeAttributeName value:(id)num range:NSMakeRange(0,
    4)];
11.     */
12. /*
13.     //设置字体属性
14.     CTFontRef font = CTFontCreateWithName(CFSTR("Georgia"), 40, NULL);
15.     [mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];
16.     */
17. /*
18.     //设置字体简隔 eg:test
19.     long number = 10;
20.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
21.     [mabstring addAttribute:(id)kCTKernAttributeName value:(id)num range:NSMakeRange(10, 4)];
22.     */
23.
24. /*
25.     long number = 1;
26.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
27.     [mabstring addAttribute:(id)kCTLigatureAttributeName value:(id)num range:NSMakeRange(0, [str
    length])];
28.      */
29. /*
30.     //设置字体颜色
31.     [mabstring addAttribute:(id)kCTForegroundColorAttributeName value:(id)[UIColor redColor].CGColor
    range:NSMakeRange(0, 9)];
32.      */
33. /*
34.     //设置字体颜色为前影色
35.     CFBooleanRef flag = kCFBooleanTrue;
36.     [mabstring addAttribute:(id)kCTForegroundColorFromContextAttributeName value:(id)flag
    range:NSMakeRange(5, 10)];
37.      */
38.
39. /*
40.     //设置空心字
41.     long number = 2;
42.     CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
43.     [mabstring addAttribute:(id)kCTStrokeWidthAttributeName value:(id)num range:NSMakeRange(0, [str
```

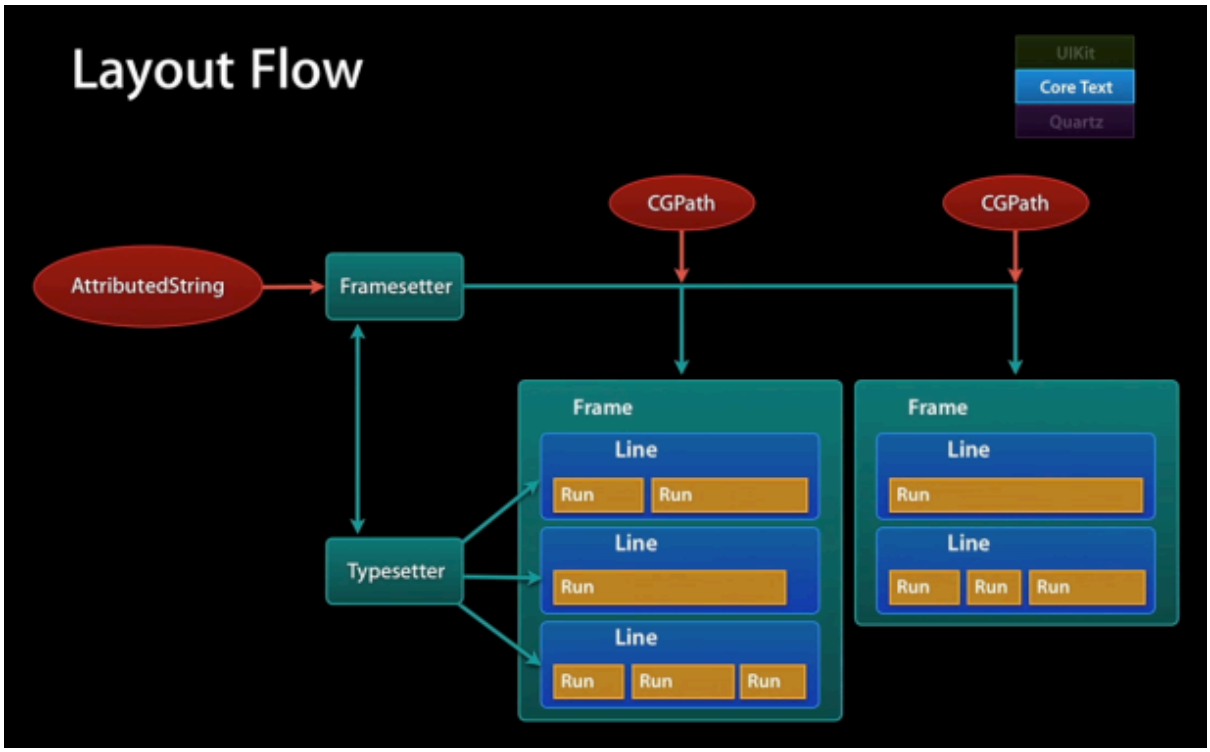
```
length])];

    //设置空心字颜色
    [mabstring addAttribute:(id)kCTStrokeColorAttributeName value:(id)[UIColor greenColor].CGColor range:NSMakeRange(0, [str length])];
     */

/*
    long number = 1;
    CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);
    [mabstring addAttribute:(id)kCTSuperscriptAttributeName value:(id)num range:NSMakeRange(3, 1)];
    */

/*
    //设置斜体字
    CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont italicSystemFontOfSize:20].fontName, 14, NULL);
    [mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];
    */

/*
    //下划线
    [mabstring addAttribute:(id)kCTUnderlineStyleAttributeName value:(id)[NSNumber numberWithInt:kCTUnderlineStyleDouble] range:NSMakeRange(0, 4)];
    //下划线颜色
    [mabstring addAttribute:(id)kCTUnderlineColorAttributeName value:(id)[UIColor redColor].CGColor range:NSMakeRange(0, 4)];
     */



//对同一段字体进行多属性设置
//红色
    NSMutableDictionary *attributes = [NSMutableDictionary dictionaryWithObject:(id)[UIColor redColor].CGColor forKey:(id)kCTForegroundColorAttributeName];
//斜体
    CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont italicSystemFontOfSize:20].fontName, 40, NULL);
    [attributes setObject:(id)font forKey:(id)kCTFontAttributeName];
//下划线
    [attributes setObject:(id)[NSNumber numberWithInt:kCTUnderlineStyleDouble] forKey:(id)kCTUnderlineStyleAttributeName];

    [mabstring addAttributes:attributes range:NSMakeRange(0, 4)];



    NSRange kk = NSMakeRange(0, 4);

    NSDictionary * dc = [mabstring attributesAtIndex:0 effectiveRange:&kk];

    [mabstring endEditing];

    NSLog(@"value = %@",dc);



    CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString((CFAttributedStringRef)mabstring);

    CGMutablePathRef Path = CGPathCreateMutable();

    CGPathAddRect(Path, NULL ,CGRectMake(10 , 0 ,self.bounds.size.width-10 , self.bounds.size.height-10));
```

```
98.
99.     CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), Path, NULL);
100.
101. //获取当前(View)上下文以便于之后的绘画，这个是一个离屏。
102.     CGContextRef context = UIGraphicsGetCurrentContext();
103.
104.     CGContextSetTextMatrix(context , CGAffineTransformIdentity);
105.
106. //压栈，压入图形状态栈中.每个图形上下文维护一个图形状态栈，并不是所有的当前绘画环境的图形状态
       的元素都被保存。图形状态中不考虑当前路径，所以不保存
107. //保存现在得上下文图形状态。不管后续对context上绘制什么都不会影响真正得屏幕。
108.     CGContextSaveGState(context);
109.
110. //x，y轴方向移动
111.     CGContextTranslateCTM(context , 0 ,self.bounds.size.height);
112.
113. //缩放x，y轴方向缩放，－1.0为反向1.0倍,坐标系转换,沿x轴翻转180度
114.     CGContextScaleCTM(context, 1.0 ,-1.0);
115.
116.     CTFrameDraw(frame,context);
117.
118.     CGPathRelease(Path);
119.     CFRelease(framesetter);
120. }
```

[cpp] view plain copy

```
01. - (void)drawRect:(CGRect)rect
02. {
03.     [self characterAttribute];
04. }
```

CORETEXT框架图



另对于Context的了解可以参考:http://www.padovo.com/blog/2013/01/31/study-coretext/