1,CoreText在系统框架中的地位。

Core Graphics

Core Text

TextKit

UILabel

WebKit

UITextField

UITextView

UIWebView

对于排版，基于CoreText和基于UIWebView相比，具有以下不同之处：

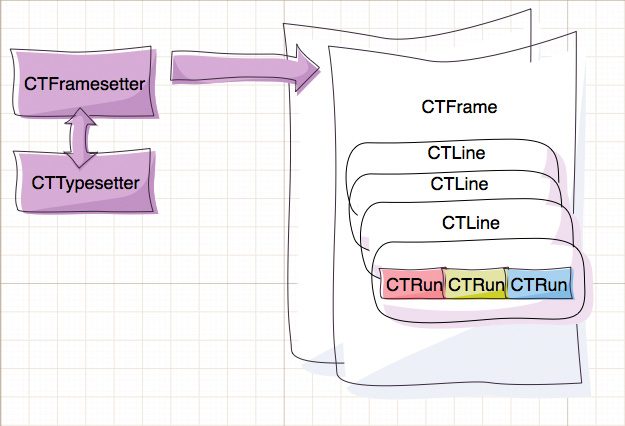
优点:

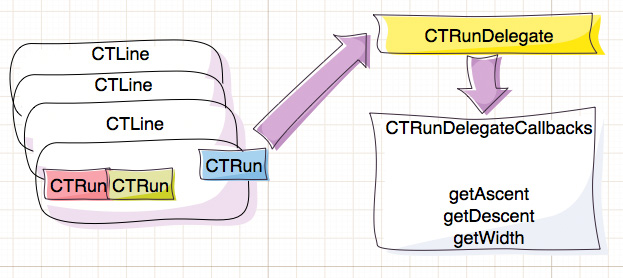
1. CoreText占用的内存更少，渲染速度更快，UIWebView占用的内存多，渲染速度慢.
2. CoreText在渲染界面前就可以精确的获得显示内容的高度(只要有了CTFrame即可),而UIWebView只有渲染出内容后，才能获得内容的高度(而且还需要用JavaScript代码来获取).
3. CoreText的CTFrame可以在后台线程渲染,UIWebView的内容只能在主线程渲染.
4. 基于CoreText可以做更好的原生交互效果,交互效果可以更细腻。而UIWebView的交互都是用JavaScript来实现的,在交互效果上会有一些卡顿情况存在.

缺点:

1. CoreText渲染出来的内容不能像UIWebView那样方便的支持内容的复制.
2. 基于CoreText来排版需要自己处理很多复杂逻辑，例如需要自己处理图片与文字混排相关的逻辑,也需要自己实现链接点击操作的支持.

CoreText对象模型如下：





CoreText 框架中最常用的几个类：

CTFont

CTFontCollection

CTFontDescriptor

CTFrame

CTFramesetter

CTGlyphInfo

CTLine

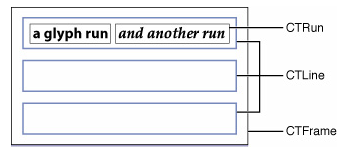
CTParagraphStyle

CTRun

CTTextTab

CTTypesetter

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。

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。

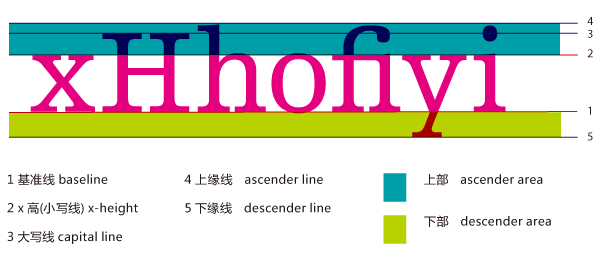
方形字(Block):这种字体的笔画使字符看起来比无衬线字更显眼,但还不到常见的衬线字的程度。例如Lubalin Graph就是方形字,这种字看起来好像是木头块刻的一样

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

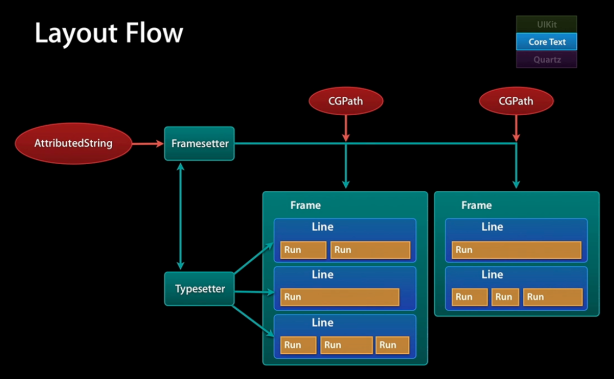
艺术字(Decorative):像绘画般的字体

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。







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0,1,2

const CFStringRef kCTForegroundColorAttributeName;

//字体颜色属性 必须是CGColor对象，默认为black

const 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NIL

const CFStringRef kCTStrokeWidthAttributeName;

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

const 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Cn1

const 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

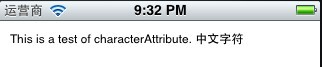
const CFStringRef kCTGlyphInfoAttributeName;

//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

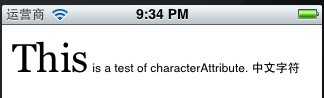
NSMutableAttributedString \*mabstring = [[NSMutableAttributedString alloc]initWithString:@"This is a test of characterAttribute. 中文字符"];



//设置字体属性

CTFontRef font = CTFontCreateWithName(CFSTR("Georgia"), 40, NULL);

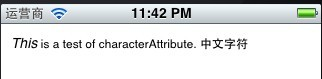
[mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];



//设置斜体字

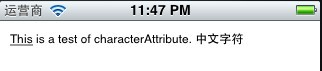
CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont italicSystemFontOfSize:20].fontName, 14, NULL);

[mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];



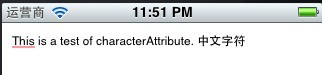
//下划线

[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)];

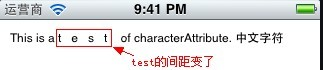


//设置字体简隔 eg:test

long number = 10;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTKernAttributeName value:(id)num range:NSMakeRange(10, 4)];



//设置连字

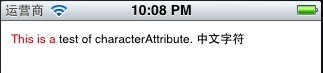
long number = 1;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTLigatureAttributeName value:(id)num range:NSMakeRange(0, [str length])];

//设置字体颜色

[mabstring addAttribute:(id)kCTForegroundColorAttributeName value:(id)[UIColor redColor].CGColor range:NSMakeRange(0, 9)];



//设置字体颜色为前影色

CFBooleanRef flag = kCFBooleanTrue;

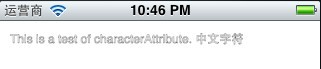
[mabstring addAttribute:(id)kCTForegroundColorFromContextAttributeName value:(id)flag range:NSMakeRange(5, 10)];

//设置空心字

long number = 2;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTStrokeWidthAttributeName value:(id)num range:NSMakeRange(0, [str length])];



//设置空心字

long number = 2;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[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])];



//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

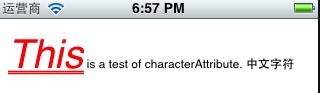
CTFontRef font = CTFontCreateWithName((CFStringRef)[UIFont italicSystemFontOfSize:20].fontName, 40, NULL);

[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)];



-(void)characterAttribute

{

NSString \*str = @"This is a test of characterAttribute. 中文字符";

NSMutableAttributedString \*mabstring = [[NSMutableAttributedString alloc]initWithString:str];

[mabstring beginEditing];

/\*

long number = 1;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTCharacterShapeAttributeName value:(id)num range:NSMakeRange(0, 4)];

\*/

/\*

//设置字体属性

CTFontRef font = CTFontCreateWithName(CFSTR("Georgia"), 40, NULL);

[mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];

\*/

/\*

//设置字体简隔 eg:test

long number = 10;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTKernAttributeName value:(id)num range:NSMakeRange(10, 4)];

\*/

/\*

long number = 1;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTLigatureAttributeName value:(id)num range:NSMakeRange(0, [str length])];

\*/

/\*

//设置字体颜色

[mabstring addAttribute:(id)kCTForegroundColorAttributeName value:(id)[UIColor redColor].CGColor range:NSMakeRange(0, 9)];

\*/

/\*

//设置字体颜色为前影色

CFBooleanRef flag = kCFBooleanTrue;

[mabstring addAttribute:(id)kCTForegroundColorFromContextAttributeName value:(id)flag range:NSMakeRange(5, 10)];

\*/

/\*

//设置空心字

long number = 2;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[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));

CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), Path, NULL);

//获取当前(View)上下文以便于之后的绘画，这个是一个离屏。

CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetTextMatrix(context , CGAffineTransformIdentity);

//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

//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。

CGContextSaveGState(context);

//x，y轴方向移动

CGContextTranslateCTM(context , 0 ,self.bounds.size.height);

//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

CGContextScaleCTM(context, 1.0 ,-1.0);

CTFrameDraw(frame,context);

CGPathRelease(Path);

CFRelease(framesetter);

}

-(void)characterAttribute

{

NSString \*str = @"This is a test of characterAttribute. 中文字符";

NSMutableAttributedString \*mabstring = [[NSMutableAttributedString alloc]initWithString:str];

[mabstring beginEditing];

/\*

long number = 1;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTCharacterShapeAttributeName value:(id)num range:NSMakeRange(0, 4)];

\*/

/\*

//设置字体属性

CTFontRef font = CTFontCreateWithName(CFSTR("Georgia"), 40, NULL);

[mabstring addAttribute:(id)kCTFontAttributeName value:(id)font range:NSMakeRange(0, 4)];

\*/

/\*

//设置字体简隔 eg:test

long number = 10;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTKernAttributeName value:(id)num range:NSMakeRange(10, 4)];

\*/

/\*

long number = 1;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[mabstring addAttribute:(id)kCTLigatureAttributeName value:(id)num range:NSMakeRange(0, [str length])];

\*/

/\*

//设置字体颜色

[mabstring addAttribute:(id)kCTForegroundColorAttributeName value:(id)[UIColor redColor].CGColor range:NSMakeRange(0, 9)];

\*/

/\*

//设置字体颜色为前影色

CFBooleanRef flag = kCFBooleanTrue;

[mabstring addAttribute:(id)kCTForegroundColorFromContextAttributeName value:(id)flag range:NSMakeRange(5, 10)];

\*/

/\*

//设置空心字

long number = 2;

CFNumberRef num = CFNumberCreate(kCFAllocatorDefault,kCFNumberSInt8Type,&number);

[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));

CTFrameRef frame = CTFramesetterCreateFrame(framesetter, CFRangeMake(0, 0), Path, NULL);

//获取当前(View)上下文以便于之后的绘画，这个是一个离屏。

CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetTextMatrix(context , CGAffineTransformIdentity);

//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

//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。

CGContextSaveGState(context);

//x，y轴方向移动

CGContextTranslateCTM(context , 0 ,self.bounds.size.height);

//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

CGContextScaleCTM(context, 1.0 ,-1.0);

CTFrameDraw(frame,context);

CGPathRelease(Path);

CFRelease(framesetter);

}

- (void)drawRect:(CGRect)rect

{

[self characterAttribute];

}