YYText 1: 基础属性与富文本, 交互与高亮

1. YYLabel本身属性

自动布局限制高度和最大宽度

`preferredMaxLayoutWidth` `numberOfLines`

垂直文字对齐

`textVerticalAlignment`

换行模式

`lineBreakMode`, 会覆盖 `NSAttributedString` 的 `yy_linebreakMode`

点击交互

`YYTextAction textTapAction`

2. NSAttributedString+YYText 扩展

设置文本颜色

`yy_color`

设置字体

`yy_font`

设置背景色

`yy_backgroundColor`

以上都是系统本身拥有的能力,不过YY做了扩展,并且内部使用CoreText来做

3. YYTextAttribute

在系统也有能力上增加一些能力,点击交互,绑定,结尾token, 图文混排

点击交互和高亮

先构造 `YYTextHighlight` 后,下面这个方法来实现点击交互和高亮

`-(void)yy_setTextHighlight:(YYTextHighlight *)textHighlight range:(NSRange)range`,

`YYTextHighlight` 构造过程如下

代码块

```
    let highlight = YYTextHighlight()
    highlight.setColor(UIColor.brown) // 按下的时候颜色
    // 点击回调
    highlight.tapAction = { containerView, text, range, rect in
    }
    highlight.longPressAction = { containerView, text, range, rect in
    }
```

他为了方便,也提供了一些便利化的方法,只不过把创建 `YYTextHighlight` 的过程实现在内部而已

代码块

```
- (void)yy_setTextHighlightRange:(NSRange)range
                           color:(UIColor *)color
                 backgroundColor:(UIColor *)backgroundColor
                        userInfo:(NSDictionary *)userInfo
                       tapAction:(YYTextAction)tapAction
                 longPressAction:(YYTextAction)longPressAction

- (void)yy_setTextHighlightRange:(NSRange)range
                           color:(UIColor *)color
                 backgroundColor:(UIColor *)backgroundColor
                       tapAction:(YYTextAction)tapAction

 - (void)yy_setTextHighlightRange:(NSRange)range
                           color:(UIColor *)color
                 backgroundColor:(UIColor *)backgroundColor
                        userInfo:(NSDictionary *)userInfo
```

token

使用 `truncationToken` 来实现自定义结尾符号,由于本身是一个 `NSAttributeString` ,可以用占位字符串用一个图片作为结尾符号.如下

代码块

```
    let attachment = NSMutableAttributedString.yy_attachmentString(withContent: imageView, contentMode: .scaleAspectFit, attachmentSize: CGSize(width: 20, height: 20), alignTo: UIFont.systemFont(ofSize: 20), alignment: .center)
```

```
    label3.truncationToken = attachment
```

但是不能给truncationToken设置`YYTextHighlight`高亮属性实现点击响应

token添加交互无法响应的根本原因：源码逻辑忽略了 Token

当手指点击 YYLabel 时，会触发 touchesBegan ，它内部调用了 _getHighlightAtPoint: 来判断你是否点中了高亮区域。注意这里的_innerText不包括truncationToken,是独立的元素

代码块

```
// YYLabel.m 源码简化版
- (YYTextHighlight *)_getHighlightAtPoint:(CGPoint)point range:(NSRangePointer)range {
    // 1. 获取点击位置对应的正文索引
    YYTextRange *textRange = [self._innerLayout textRangeAtPoint:point];

    // 2. 注意这里：它只去 _innerText（你的 label.attributedText）里查找属性
    YYTextHighlight *highlight = [_innerText attribute:YYTextHighlightAttributeName atIndex:startIndex ...];
    return highlight;
}
```

`yylabel`中的`textLayout`中有一个属性`truncatedLine`,代表截断符号所在的行,如没有截断符那么该值为nil

通过以下代码实现点击截断符交互

swift

```
        label3.textTapAction = { containerView, text, range, rect in
              if let  truncationLine = (containerView as! YYLabel).textLayout?.truncatedLine {
                  let tokenRect = CGRect(x: truncationLine.left + truncationLine.width - 25,
                                         y: truncationLine.top,
                                         width: 25,
                                         height: truncationLine.height)
                  if tokenRect.intersects(rect) {
                      print("点击token")
                  }
              }
          }
```

绑定

图文混排

原理: **用一个占位符字符串生成属性字符串,添加 YYTextAttachment对象属性,为这个占位字符串设置尺寸**.这三步实际上YYText都已经封住好了,不需要自己调用,而是对外暴露了三个方法. 我们只需要生成attachment的string.添加到原有的字符串后面就好了

attachment

```
+ (NSMutableAttributedString *)yy_attachmentStringWithContent:(id)content
                                                  contentMode:
(UIViewContentMode)contentMode
                                                attachmentSize:
(CGSize)attachmentSize
                                                  alignToFont:(UIFont *)font
                                                    alignment:
(YYTextVerticalAlignment)alignment

 + (NSMutableAttributedString *)yy_attachmentStringWithContent:(id)content
                                                  contentMode:
(UIViewContentMode)contentMode
                                                        width:(CGFloat)width
                                                       ascent:(CGFloat)ascent
                                                      descent:(CGFloat)descent

+ (NSMutableAttributedString *)yy_attachmentStringWithEmojiImage:(UIImage
*)image
                                                         fontSize:
(CGFloat)fontSize
```

`NSMutableAttributedString` 的 `yy_attachmentString` 系列方法（实际上是 `NSAttributedString+YYText` 分类中的方法）主要用于生成包含“附件”（Attachment，通常是图片或UIView）的富文本字符串。

这三个方法分别是：

1. `yy_attachmentStringWithContent:contentMode:attachmentSize:alignToFont:alignment:`

- **场景**：**最通用、最强大**的图文混排。
- **用途**：当你需要把一个 `UIImage`、`UIView` 或者是 `CALayer` 插入到文字中间，并且要求它与周围的文字（Font）进行**完美的对齐**（比如居中对齐、底部对齐）时使用。
- **核心参数**：
  - `content`：可以是 `UIImage` / `UIView` / `CALayer`。

- `alignToFont`：**关键！** 你传入周围文字所用的 `UIFont`，YYText 会自动计算基线（Baseline），确保图片和文字在垂直方向上完美居中对齐，而不需要你手动去算 `ascent` 和 `descent`。

- **例子**：在文字末尾加一个小图标（🔥），让图标和文字垂直居中。

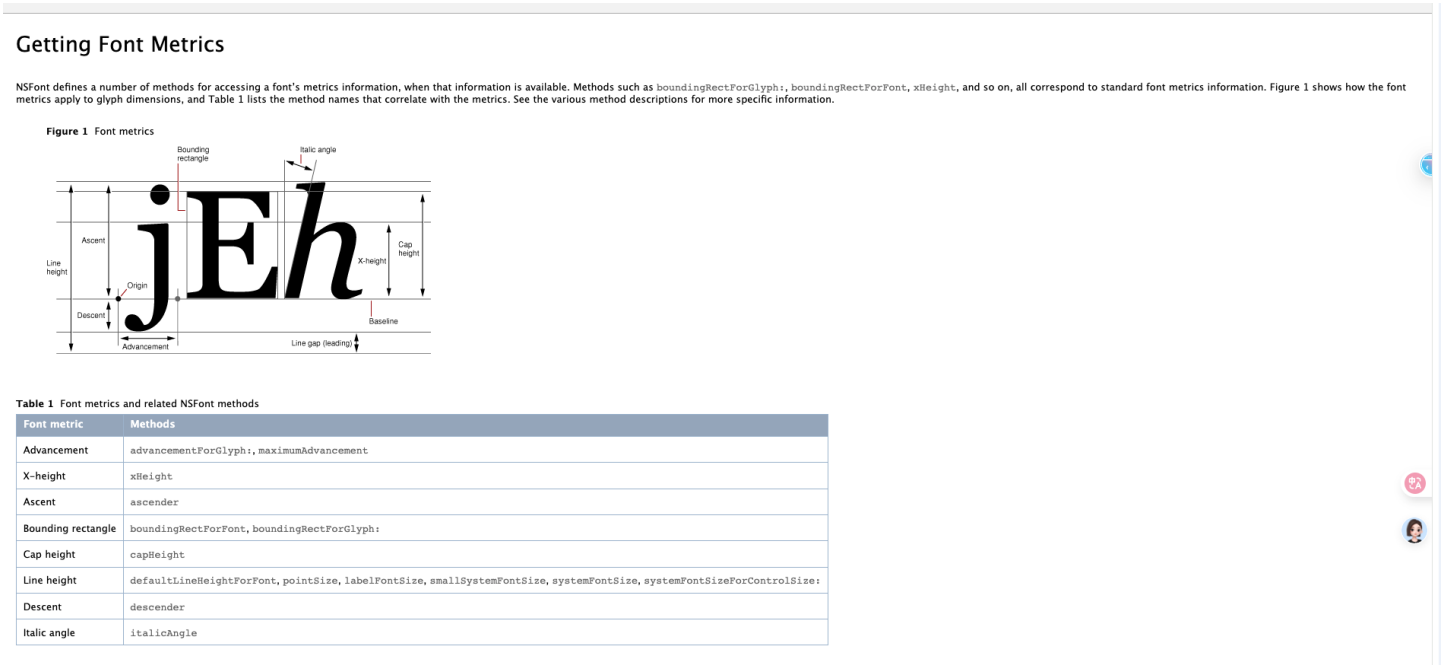
attachmentSize 决定了attachment的尺寸, UIViewContentMode决定了图片的填充模式, alignToFont 代表附件前的文字尺寸,alignment代表在font下的垂直对齐方式

2. `yy_attachmentStringWithContent:contentMode:width:ascent:descent:`

- **场景**：**手动控制排版尺寸**。
- **用途**：当你不想依赖字体自动计算，而是想**精确控制**这个附件占据多宽、多高、基线在哪时使用。
- **核心参数**：
  - `ascent` (上行高度) / `descent` (下行高度)。
- **原理**：这其实就是刚才你问的 `YYTextRunDelegate` 的封装版。它直接让你填这三个数。
- **例子**：你需要插入一个非常大的图片，且它的位置非常特殊，自动对齐无法满足需求时。

> Core Text Programming Guide

- https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/FontHandling/Tasks/GettingFontMetrics.html



Getting Font Metrics

NSFont defines a number of methods for accessing a font's metrics information, when that information is available. Methods such as `boundingRectForGlyph:`, `boundingRectForFont`, `xHeight`, and so on, all correspond to standard font metrics information. Figure 1 shows how the font metrics apply to glyph dimensions, and Table 1 lists the method names that correlate with the metrics. See the various method descriptions for more specific information.

Figure 1 Font metrics

Table 1 Font metrics and related NSFont methods

Font metric	Methods
Advancement	`advancementForGlyph:`, `maximumAdvancement`
X-height	`xHeight`
Ascent	`ascender`
Bounding rectangle	`boundingRectForFont`, `boundingRectForGlyph:`
Cap height	`capHeight`
Line height	`defaultLineHeightForFont`, `pointSize`, `labelFontSize`, `smallSystemFontSize`, `systemFontSize`, `systemFontSizeForControlSize:`
Descent	`descender`
Italic angle	`italicAngle`

- **场景**：**Emoji 表情 / 小微标**。
- **用途**：这是一个简便方法，专门用来生成像 Emoji 那样的正方形小图片。
- **特点**：

    - **它假设附件是正方形的。**
    - **它会自动让图片大小等于 `fontSize`（字体大小）。**
- **例子**：聊天气泡里把文本 `[微笑]` 替换成一张 16x16 的微笑图片。

总结选择指南

- **想省事，要文字图片对齐** -> 用 **方法 1**（传 Font 进去，让它自己算）。
- **要精细控制尺寸** -> 用 **方法 2**（自己传 ascent/descent）。
- **只是插个小表情** -> 用 **方法 3**。