

- [Youtube](#)
- [Twitter](#)
- [Email](#)
- [RSS](#)

Jump

- [Home](#)
- [Blog](#)
- [Jobs](#)
- [Our Apps](#)
- [Advertise](#)

Tappable UILabel Hyperlinks

Mar 04, 2015

In the previous blog post of this series, I have demonstrated how to [modify the way UILabel draws hyperlinks](#). In this article I am showing how to implement function to make the hyperlinks tappable.

Ad

We previously implemented a custom stack of **NSLayoutManager**, **NSTextContainer** and **NSTextStorage**. This was necessary because – unfortunately – **UILabel** does not give us access to its internal instances of these. Having our own layout manager in place is the key ingredient to making the hyperlinks tappable.

When the user brings his finger down on our customised UILabel we need to first determine which character index of the attributed string he landed on. Then we can look up the **NSLinkAttributeName** for this index. Finally we can change the appearance of this link based on the touch.

As in the previous article, I thank [Matthew Styles](#) for working this out for this [KILabel](#) implementation. I re-implemented my solution from scratch, making many different – I like think more elegant – design choices, so please bear with me and don't rush off cloning the KILabel project.

Determining String Index of Touch

The layout manager instance we created provides all the necessary methods to determine the character index into the attributed string.

```
- (NSUInteger)_stringIndexAtLocation:(CGPoint)location
{
    // Do nothing if we have no text
    if (self.textStorage.string.length == 0)
    {
        return NSNotFound;
    }

    // Work out the offset of the text in the view
    CGPoint textOffset;
    NSRange glyphRange = [self.layoutManager
                           glyphRangeForTextContainer:self.textContainer];
    textOffset = [self textOffsetForGlyphRange:glyphRange];
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept

```

NSUInteger glyphIndex = [self.layoutManager glyphIndexForPoint:location
                           inTextContainer:self.textContainer];

// If the touch is in white space after the last glyph on the line we don't
// count it as a hit on the text
NSRange lineRange;
CGRect lineRect = [self.layoutManager lineFragmentUsedRectForGlyphAtIndex:glyphIndex
                           effectiveRange:&lineRange];

if (!CGRectContainsPoint(lineRect, location))
{
    return NSNotFound;
}

return [self.layoutManager characterIndexForGlyphAtIndex:glyphIndex];
}

```

In order to retrieve the NSURL for a given character index, I implemented a category method for NSAttributedString. Ok, it's a one-liner, but a method of the same signature exists for OS X:

```

- (NSURL *)URLAtIndex:(NSUInteger)location
    effectiveRange:(NSRangePointer)effectiveRange
{
    return [self attribute:NSLinkAttributeName atIndex:location
        effectiveRange:effectiveRange];
}

```

To show a gray background as long as the link is touched we simply set a background color for the effective range. When the finger is lifted we reset it. The following helper method – the setter for the selectedRange property – takes care of setting and removing said background color attribute.

```

- (void)setSelectedRange:(NSRange)range
{
    // Remove the current selection if the selection is changing
    if (self.selectedRange.length && !NSEqualRanges(self.selectedRange, range))
    {
        [self.textStorage removeAttribute:NSBackgroundColorAttributeName
                           range:self.selectedRange];
    }

    // Apply the new selection to the text
    if (range.length)
    {
        [self.textStorage addAttribute:NSBackgroundColorAttributeName
                               value:self.selectedLinkBackgroundColor
                               range:range];
    }
}

```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept

```
[self setNeedsDisplay];
}
```

This is all it takes in terms of display because the drawing already takes care of displaying the boxes. Note the additional property `selectedLinkBackgroundColor` which allows customization.

Handling Gestures

The action to be executed on a user's tapping on a hyperlink goes into a `linkTapHandler` property. This block property has one parameter for passing the tapped URL. The `_commonSetup` method gets called from both `initWithFrame:` and `initWithCoder:` to set up the gesture recogniser and default tap handler.

```
- (void)_commonSetup
{
    // Make sure user interaction is enabled so we can accept touches
    self.userInteractionEnabled = YES;

    // Default background colour looks good on a white background
    self.selectedLinkBackgroundColor = [UIColor colorWithWhite:0.95 alpha:1.0];

    // Attach a default detection handler to help with debugging
    self.linkTapHandler = ^(NSURL *URL) {
        NSLog(@"Default handler for %@", URL);
    };

    UITapGestureRecognizer *touch = [[UITapGestureRecognizer alloc] initWithTarget:self
                                         action:@selector(_handleTouch:)];
    touch.delegate = self;
    [self addGestureRecognizer:touch];
}
```

The `UITapGestureRecognizer` is a very simple recogniser which simply wraps the `touchesBegan`, `-Moved`, `-Ended` and `-Cancelled`.

@implementation `UITapGestureRecognizer`

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
    self.state = UIGestureRecognizerStateBegan;
}

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event
{
    self.state = UIGestureRecognizerStateFailed;
}

- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
    self.state = UIGestureRecognizerStateEnded;
}
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept

```

self.state = UIGestureRecognizerStateCancelled;
}

```

```
@end
```

The gesture recognizer handling method ties the methods together which we have previously implemented.

```

- (void)_handleTouch:(TouchGestureRecognizer *)gesture
{
    // Get the info for the touched link if there is one
    CGPoint touchLocation = [gesture locationInView:self];
    NSInteger index = [self _stringIndexAtLocation:touchLocation];

    NSRange effectiveRange;
    NSURL *touchedURL = nil;

    if (index != NSNotFound)
    {
        touchedURL = [self.attributedText URLAtIndex:index effectiveRange:&effectiveRange];
    }

    switch (gesture.state)
    {
        case UIGestureRecognizerStateBegan:
        {
            if (touchedURL)
            {
                self.selectedRange = effectiveRange;
            }
            else
            {
                // no URL, cancel gesture
                gesture.enabled = NO;
                gesture.enabled = YES;
            }

            break;
        }

        case UIGestureRecognizerStateEnded:
        {
            if (touchedURL &&& _linkTapHandler)
            {
                _linkTapHandler(touchedURL);
            }
        }

        default:
        {

```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept

```

    }
}
}

```

We want to consider the gesture as failed if the touch does not start on a hyperlink. This is important when using our UILabel implementation in a table view. We even want to go one step further. Our touch gesture recognizer should always recognize at the same time as other gesture recognizers.

To prevent any adverse blocking effects on scroll gesture recognizers, we want only touches be delivered to the touch gesture recognizer if they are on top of hyperlinks. The code for determining this is quite similar to the gesture handling method.

```

- (BOOL)gestureRecognizer:(UIGestureRecognizer *)gestureRecognizer
  shouldRecognizeSimultaneouslyWithGestureRecognizer:(UIGestureRecognizer *)otherGestureRecognizer
{
    return YES;
}

- (BOOL)gestureRecognizer:(UIGestureRecognizer *)gestureRecognizer
  shouldReceiveTouch:(UITouch *)touch
{
    CGPoint touchLocation = [touch locationInView:self];

    NSInteger index = [self _stringIndexAtLocation:touchLocation];

    NSRange effectiveRange;
    NSURL *touchedURL = nil;

    if (index != NSNotFound)
    {
        touchedURL = [self.attributedText URLAtIndex:index effectiveRange:&effectiveRange];
    }

    if (touchedURL)
    {
        return YES;
    }

    return NO;
}

```

With these additional modifications in place our label can also be used in table views without interfering with vertical scrolling. Also horizontal swipes (for editing) are unimpeded as well.

Conclusion

This is all the code that is necessary to get tap handling working for hyperlinks. It's quite a few lines of code, but they should not be too hard to wrap your head around.

It would all be much easier if Apple allowed us access to the TextKit stack of UILabel, specifically letting us replace the internal layout

mi Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
ba To find out more, including how to control cookies, see here: [Cookie Policy](#).

Close and accept

At the moment there is no public sample project containing the code mentioned, because I feel it is somewhat specific to my needs **for the prod.ly app**. Let's make a deal: I'll trade you the working source code I have for purchasing a copy of **my new book**. I am also looking for my next iOS and/or Mac project to participate in. Either way, you can contact me at **oliver@cocoanetics.com**.

Sharing:

Like this:

Loading...

Related

Customizing UILabel Hyperlinks

March 3, 2015

In "Recipes"

Draggable Buttons and Labels

November 25, 2010

In "Recipes"


Cells with Switch

July 21, 2010

In "Recipes"

Categories: [Recipes](#)[← Customizing UILabel Hyperlinks](#)[Proud Day →](#)


2 Comments »

1.  *Jim*
[June 3, 2015 | 4:03 am](#)

Safe to assume that `_textOffsetForGlyphRange:` is very similar to `KILabel's calcGlyphsPositionInView?`

Thanks

Jim

2.  *Jim*
[June 3, 2015 | 2:31 pm](#)

Ah.. never mind, it's in the previous post.

Jim

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept