

ios® Developer Notes for Professionals

Chapter 4: attributedText in UILabel

The current styled text that is displayed by the label.
You can add HTML text in UILabel using attributedText property or customized single UILabel property

Section 4.1: HTML text in UILabel

NSString * attributedString = @"html-body Example bold text in HTML.
 ";
NSAttributedString * attributedString = [[NSAttributedString alloc] initWithHTML:attributedString encoding:NSUTF8StringEncoding documentAttributes:nil error:nil];
UILabel * yourLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 0, 100, 100)];
yourLabel.attributedString = attributedString;

Section 4.2: Set different property to text in UILabel

The first step you need to perform is to create a NSMutableAttributedString object. NSMutableAttributedString instead of NSAttributedString because it enables us

```
NSMutableAttributedString * attributedString = [[NSMutableAttributedString alloc] initWithString:@"Example attributed text"];  
  
// Finding the range of text.  
NSRange rangeHello = [attributedString rangeOfString:@"Hello"];  
NSRange rangeWorld = [attributedString rangeOfString:@"World"];  
  
// Add font style for Hello  
[attributedString setAttributes:@{NSFontAttributeName: @("Helvetica")}, range: rangeHello];  
  
// Add text color for Hello  
[attributedString setAttributes:@{NSForegroundColorAttributeName: [UIColor blueColor]}, range: rangeHello];  
  
// Add font style for World  
[attributedString setAttributes:@{NSFontAttributeName: @("Chalkboard")}, range: rangeWorld];  
  
// Add text color for World  
[attributedString setAttributes:@{NSForegroundColorAttributeName: [UIColor greenColor]}, range: rangeWorld];  
  
// Set it to UILabel as NSAttributedString  
UILabel * yourLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 0, 100, 100)];  
yourLabel.attributedString = attributedString;  
[self.view addSubview:yourLabel];
```

GoalKicker.com - iPhone Developer's Notes for Professionals

Chapter 13: Cut a UIImage into a circle Section 13.1: Cut a image into a circle - Objective C

The code in the viewDidLoad or loadView should look something like this

```
-(void)viewDidLoad {  
    [super viewDidLoad];  
    UIImageView *imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0, 0, 320, 320)];  
    [self.view addSubview:imageView];  
    UIImage *image = [UIImage imageNamed:@"Himal-Photos-Images-Travel-Tourist-Images-Pictures-800x800.jpg"];  
    imageView.image = [self.circularScaleAndCropImage:[UIImage imageNamed:@"Himal-Photos-Images-Travel-Tourist-Images-Pictures-800x800.jpg"] frame:CGRectMake(0, 0, 320, 320)];  
}
```

Finally the function which does the heavy lifting (circularScaleAndCropImage) is as defined below

```
-(UIImage *)circularScaleAndCropImage:(UIImage *)image frame:(CGRect)frame {  
    // This function returns a new image, based on image, that has been:  
    // - scaled to fit in (CGRect) rect.  
    // - and cropped within a circle of radius: rect.size.width/2  
  
    // Create the bitmap graphics context  
    UIGraphicsBeginImageContextWithOptions(frame.size.width, frame.size.height, NO, 0.0);  
    CGContextRef context = UIGraphicsGetCurrentContext();  
  
    // Get the width and height  
    CGFloat imageWidth = image.size.width;  
    CGFloat imageHeight = image.size.height;  
    CGFloat rectWidth = frame.size.width;  
    CGFloat rectHeight = frame.size.height;  
  
    // Calculate the scale factor  
    CGFloat scaleFactor = rectWidth/imageWidth;  
    CGFloat scaleFactor = rectHeight/imageHeight;  
  
    // Calculate the centre of the circle  
    CGPoint imageCentre = CGPointMake(0, 0);  
    CGPoint rectCentre = CGPointMake(rectWidth/2, rectHeight/2);  
  
    // Create and clip to a circular path  
    // (This could be replaced with any closed path if you want a different shaped clip)  
    CGContextSaveGState(context);  
    CGContextTranslateCTM(context, imageCentre.x, imageCentre.y);  
    CGContextScaleCTM(context, scaleFactor, scaleFactor);  
    CGContextClipToMask(context, imageWidth, imageHeight, image);  
    CGContextTranslateCTM(context, rectCentre.x, rectCentre.y);  
    CGContextScaleCTM(context, scaleFactor, scaleFactor);  
  
    // Draw the IMAGE  
    CGContextDrawImage(context, CGRectMake(0, 0, imageWidth, imageHeight));  
    CGContextRestoreGState(context);  
    UIImage *img = UIGraphicsGetImageFromCurrentImageContext();  
    UIGraphicsEndImageContext();  
    return img;  
}
```

GoalKicker.com - iPhone Developer's Notes for Professionals

Chapter 107: Push Notifications

Parameter
userInfo: A dictionary that contains remote notification info, potentially including a badge number for the app icon, alert sound, alert message, a notification identifier, and custom data.

Section 107.1: Registering device for Push Notifications

To register your device for push notifications, add the following code to your AppDelegate file in AppDelegate.m

```
Swift  
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSDictionary: AnyObject]? = nil) -> Bool {  
    // Override point for customization after application launch.  
    if (UIApplication.sharedApplication().systemVersion.compare(Version(10, 0)) >= NSOrderedAscending(1)) {  
        // Register for Push Notifications, if running iOS 10.0 +  
        let types: UNNotificationTypes = [.Alert, .Sound, .Banner];  
        let settings: UNNotificationSettings = UNNotificationSettings(forTypes: types, categories: nil);  
        application.registerForRemoteNotifications(withSettings: settings);  
    } else {  
        // Register for Push Notifications before iOS 10  
        application.registerForRemoteNotifications();  
    }  
    var center = UNUserNotificationCenter(options: [UNAuthorizationOptionsAlert | UNAuthorizationOptionsBadge | UNAuthorizationOptionsSound]);  
    center.requestAuthorizationWithOptions([UNAuthorizationOptionsAlert | UNAuthorizationOptionsBadge | UNAuthorizationOptionsSound]) { (granted: Bool, error: NSError?) -> Void  
    }  
    if (error != nil) {  
        // Application should be able to receive push notifications.  
        // Required to get the app to do anything at all about push notifications  
        print("Push registration failed")  
        print("Error: \(error.localizedDescription) | \(error.localizedDescription)")  
        print("SUCCESS: \(error.localizedDescription) | \(error.localizedDescription)")  
    }  
    return true  
}
```

Objective-C
- (void)registerForRemoteNotifications {
 if ([self respondsToSelector:@selector(registerForRemoteNotifications)]) {
 [self registerForRemoteNotifications];
 }
}

GoalKicker.com - iPhone Developer's Notes for Professionals

800+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with iOS Development	2
Section 1.1: Creating a default Single View Application	2
Section 1.2: Hello World	6
Section 1.3: Xcode Interface	11
Section 1.4: Create your first program in Swift 3	17
Chapter 2: UILabel	22
Section 2.1: Create a UILabel	22
Section 2.2: Number of Lines	24
Section 2.3: Set Font	25
Section 2.4: Text Color	26
Section 2.5: Background Color	27
Section 2.6: Size to fit	27
Section 2.7: Text alignment	30
Section 2.8: Calculate Content Bounds (for i.e. dynamic cell heights)	30
Section 2.9: Label Attributed Text	32
Section 2.10: Clickable Label	38
Section 2.11: Variable height using constraints	39
Section 2.12: LineBreakMode	39
Section 2.13: Add shadows to text	41
Section 2.14: Changing Text in an Existing Label	41
Section 2.15: Auto-size label to fit text	42
Section 2.16: Get UILabel's size strictly based on its text and font	43
Section 2.17: Highlighted and Highlighted Text Color	44
Section 2.18: Justify Text	44
Section 2.19: Dynamic label frame from unknown text length	45
Chapter 3: UILabel text underlining	47
Section 3.1: Underlining a text in a UILabel using Objective C	47
Section 3.2: Underlining a text in UILabel using Swift	47
Chapter 4: attributedText in UILabel	48
Section 4.1: HTML text in UILabel	48
Section 4.2: Set different property to text in single UILabel	48
Chapter 5: UIButton	50
Section 5.1: Creating a UIButton	50
Section 5.2: Attaching a Method to a Button	50
Section 5.3: Setting Font	51
Section 5.4: Set Image	51
Section 5.5: Get UIButton's size strictly based on its text and font	51
Section 5.6: Disabling a UIButton	52
Section 5.7: Set title	52
Section 5.8: Set title color	52
Section 5.9: Horizontally aligning contents	53
Section 5.10: Getting the title label	53
Section 5.11: Adding an action to an UIButton via Code (programmatically)	54
Chapter 6: UIDatePicker	55
Section 6.1: Create a Date Picker	55
Section 6.2: Setting Minimum-Maximum Date	55

Section 6.3: Modes	55
Section 6.4: Setting minute interval	55
Section 6.5: Count Down Duration	56
Chapter 7: ULocalNotification	57
Section 7.1: Scheduling a local notification	57
Section 7.2: Presenting a local notification immediately	57
Section 7.3: Managing local notifications using UUID	58
Section 7.4: Registering for local notifications	59
Section 7.5: what's new in ULocalNotification with iOS10	60
Section 7.6: Responding to received local notification	62
Section 7.7: Register and Schedule Local Notification in Swift 3.0 (iOS 10)	62
Chapter 8: UIImage	64
Section 8.1: Creating UIImage	64
Section 8.2: Comparing Images	65
Section 8.3: Gradient Image with Colors	66
Section 8.4: Convert UIImage to/from base64 encoding	66
Section 8.5: Take a Snapshot of a UIView	67
Section 8.6: Change UIImage Color	67
Section 8.7: Apply UIColor to UIImage	67
Section 8.8: Creating and Initializing Image Objects with file contents	68
Section 8.9: Resizable image with caps	68
Section 8.10: Gradient Background Layer for Bounds	69
Chapter 9: Convert NSAttributedString to UIImage	70
Section 9.1: NSAttributedString to UIImage Conversion	70
Chapter 10: UIImagePickerController	71
Section 10.1: Generic usage of UIImagePickerController	71
Chapter 11: UIImageView	73
Section 11.1: UIImage masked with Label	73
Section 11.2: Making an image into a circle or rounded	73
Section 11.3: How the Mode property affects an image	74
Section 11.4: Animating a UIImageView	80
Section 11.5: Create a UIImageView	81
Section 11.6: Change color of an image	82
Section 11.7: Assigning an image to a UIImageView	82
Chapter 12: Resizing UIImage	83
Section 12.1: Resize any image by size & quality	83
Chapter 13: Cut a UIImage into a circle	84
Section 13.1: Cut a image into a circle - Objective C	84
Section 13.2: SWIFT 3 Example	85
Chapter 14: UITableView	87
Section 14.1: Self-Sizing Cells	87
Section 14.2: Custom Cells	87
Section 14.3: Separator Lines	90
Section 14.4: Delegate and Datasource	92
Section 14.5: Creating a UITableView	98
Section 14.6: Swipe to Delete Rows	102
Section 14.7: Expanding & Collapsing UITableViewCells	105
Chapter 15: UITableViewController	108
Section 15.1: TableView with dynamic properties with tableViewCellStyle basic	108

Section 15.2: TableView with Custom Cell	109
Chapter 16: UIRefreshControl TableView	111
Section 16.1: Set up refreshControl on tableView:	111
Section 16.2: Objective-C Example	111
Chapter 17: UITableViewCell	113
Section 17.1: Xib file of UITableViewCell	113
Chapter 18: Custom methods of selection of UITableViewCells	114
Section 18.1: Distinction between single and double selection on row	114
Chapter 19: Custom methods of selection of UITableViewCells	115
Section 19.1: Distinction between single and double selection on row	115
Chapter 20: UIView	116
Section 20.1: Make the view rounded	116
Section 20.2: Using IBInspectable and IBDesignable	118
Section 20.3: Taking a snapshot	121
Section 20.4: Create a UIView	121
Section 20.5: Shake a View	121
Section 20.6: Utilizing Intrinsic Content Size	122
Section 20.7: Programmatically manage UIView insertion and deletion into and from another UIView	124
Section 20.8: Create UIView using Autolayout	125
Section 20.9: Animating a UIView	127
Section 20.10: UIView extension for size and frame attributes	127
Chapter 21: Snapshot of UIView	129
Section 21.1: Getting the Snapshot	129
Section 21.2: Snapshot with subview with other markup and text	129
Chapter 22: UIAlertController	131
Section 22.1: AlertViews with UIAlertController	131
Section 22.2: Action Sheets with UIAlertController	132
Section 22.3: Adding Text Field in UIAlertController like a prompt Box	135
Section 22.4: Highlighting an action button	135
Section 22.5: Displaying and handling alerts	136
Chapter 23: UIColor	141
Section 23.1: Creating a UIColor	141
Section 23.2: Creating a UIColor from hexadecimal number or string	142
Section 23.3: Color with Alpha component	144
Section 23.4: Undocumented Methods	145
Section 23.5: UIColor from an image pattern	146
Section 23.6: Lighter and Darker Shade of a given UIColor	147
Section 23.7: Make user defined attributes apply the CGColor datatype	148
Chapter 24: UITextView	149
Section 24.1: Set attributed text	149
Section 24.2: Change font	149
Section 24.3: Auto Detect Links, Addresses, Dates, and more	149
Section 24.4: Change text	150
Section 24.5: Change text alignment	150
Section 24.6: UITextViewDelegate methods	150
Section 24.7: Change text color	151
Section 24.8: Remove extra paddings to fit to a precisely measured text	151
Section 24.9: Getting and Setting the Cursor Position	151

Section 24.10: UITextView with HTML text	153
Section 24.11: Check to see if empty or nil	153
Chapter 25: UITextField Delegate	154
Section 25.1: Actions when a user has started/ended interacting with a textfield	154
Section 25.2: UITextField - Restrict textfield to certain characters	155
Chapter 26: UINavigationController	156
Section 26.1: Embed a view controller in a navigation controller programmatically	156
Section 26.2: Popping in a Navigation Controller	156
Section 26.3: Purpose	156
Section 26.4: Pushing a view controller onto the navigation stack	157
Section 26.5: Creating a UINavigationController	157
Chapter 27: UIGestureRecognizer	158
Section 27.1: UITapGestureRecognizer	158
Section 27.2: UITapGestureRecognizer (Double Tap)	159
Section 27.3: Adding a Gesture recognizer in the Interface Builder	159
Section 27.4: UILongPressGestureRecognizer	160
Section 27.5: UISwipeGestureRecognizer	161
Section 27.6: UIPinchGestureRecognizer	162
Section 27.7: UIRotationGestureRecognizer	163
Chapter 28: UIBarButtonItem	164
Section 28.1: Creating a UIBarButtonItem in the Interface Builder	164
Section 28.2: Creating a UIBarButtonItem	167
Section 28.3: Bar Button Item Original Image with no Tint Color	167
Chapter 29: UIScrollView	168
Section 29.1: Scrolling content with Auto Layout enabled	168
Section 29.2: Create a UIScrollView	171
Section 29.3: ScrollView with AutoLayout	171
Section 29.4: Detecting when UIScrollView finished scrolling with delegate methods	176
Section 29.5: Enable/Disable Scrolling	176
Section 29.6: Zoom In/Out UIImageView	177
Section 29.7: Scroll View Content Size	178
Section 29.8: Restrict scrolling direction	178
Chapter 30: UIStackView	179
Section 30.1: Center Buttons with UIStackview	179
Section 30.2: Create a horizontal stack view programmatically	183
Section 30.3: Create a vertical stack view programmatically	184
Chapter 31: Dynamically updating a UIStackView	185
Section 31.1: Connect the UISwitch to an action we can animate switching between a horizontal or vertical layout of the image views	185
Chapter 32: UIScrollView with StackView child	186
Section 32.1: A complex StackView inside Scrollview Example	186
Section 32.2: Preventing ambiguous layout	187
Section 32.3: Scrolling to content inside nested StackViews	188
Chapter 33: UIScrollView AutoLayout	189
Section 33.1: ScrollableController	189
Section 33.2: UIScrollView dynamic content size through Storyboard	193
Chapter 34: UITextField	195
Section 34.1: Get Keyboard Focus and Hide Keyboard	195
Section 34.2: Dismiss keyboard when user pushes the return button	195

[Click here to download full PDF material](#)