详解 UIView 的 Tint Color 属性

lansekuangtu · 2015-07-03 · 阅读数 43902

本文来自 简书的技术博客，作者 lansekuangtu

在iOS 7后，UIView新增加了一个tintColor属性，这个属性定义了一个非默认的着色颜色值，其值的设置会影响到以视图为根视图的整个视图层次结构。它主要是应用到诸如app图标、导航栏、按钮等一些控件上，以获取一些有意思的视觉效果。

tintColor属性的声明如下：

```
var tintColor: UIColor!
```

默认情况下，一个视图的tintColor是为nil的，这意味着视图将使用父视图的tint color值。当我们指定了一个视图的tintColor后，这个色值会自动作播到视图层次结构以当前视图为根视图中所有的子视图上。如果系统在视图层次结构中没有找到一个非默认的tintColor值，则会使用系统定义的颜色值(蓝色，RGB值为[0,0.478431,1]，我们可以在IOS中看到这个颜色)。因此，这个值总是会返回一个颜色值，即我们没有指定它。

与tintColor属性相关的还有个tintAdjustmentMode属性，它是一个枚举值，定义了tint color的调整模式，其声明如下：

```
var tintAdjustmentMode: UIViewTintAdjustmentMode
```

枚举UIViewTintAdjustmentMode的定义如下：

```
enum UIViewTintAdjustmentMode : Int {
    case Automatic          // 视图的着色调整模式与父视图一致
    case Normal             // 视图的tintColor属性返回完全未修改的视图着色颜色
    case Dimmed             // 视图的tintColor属性返回一个去饱和度的、变暗的视图着色颜色
}
```

因此，当tintAdjustmentMode属性设置为Dimmed时，tintColor的颜色值会自动变暗，而如果我们在视图层次结构中没有找到默认值，则该值默认是Normal。

与tintColor相关的还有一个tintColorDidChange方法，其声明如下：

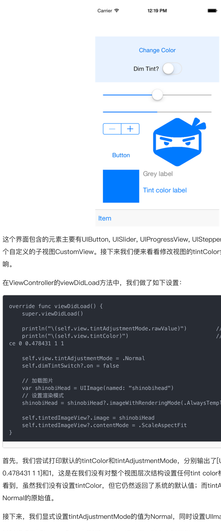
```
func tintColorDidChange()
```

这个方法会在视图的tintColor或tintAdjustmentMode属性改变时自动调用，另外，如果当前视图的父视图的tintColor或tintAdjustmentMode属性改变时，也会调用这个方法，我们可以在这个方法中根据需要去刷新我们的视图。

示例

接下来我们通过示例来看看tintColor的强大功能(示例逼用了Sam Davies写的一个例子，具体可以查看 iOS7 Day–by–Day :: Day 6 :: Tint Color，我就负责搬砖，用swift实现了一下，代码可以在这里下载)。

先来看看最终效果吧(以下都是逼图，请见谅，太懒了)：



这个界面包含的元素主要有UIButton, UISlider, UIProgressView, UIStepper, UIImageView, ToolBar和一个自定义的子视图CustomView。接下来我们便来看看修改视图的tintColor会对这些控件产生什么样的影响。

在ViewController的viewDidLoad方法中，我们做了如下设置：

```
override func viewDidLoad() {
    super.viewDidLoad()

    println("\(self.view.tintAdjustmentMode.rawValue)")          // 输出: 1
    println("\(self.view.tintColor)")                            // 输出: UIDeviceRGBColorSpa
ce 0 0.478431 1 1

    self.view.tintAdjustmentMode = .Normal
    self.dimTintSwitch?.on = false

    // 加载图片
    var shinobiHead = UIImage(named: "shinobihead")
    // 设置渲染模式
    shinobiHead = shinobiHead?.imageWithRenderingMode(.AlwaysTemplate)

    self.tintedImageView?.image = shinobiHead
    self.tintedImageView?.contentMode = .ScaleAspectFit
}
```

首先，我们尝试打印默认的tintColor和tintAdjustmentMode，分别输出了UIDeviceRGBColorSpace 0 0.478431 1 1和1，这是在我们没有对整个视图层次结构设置任何tint color相关的值的情况下的输出，可以看到，虽然我们没有设置tintColor，但它仍然返回了系统的默认值：而tintAdjustmentMode则默认返回Normal的原始值。

接下来，我们显式设置tintAdjustmentMode的值为Normal，同时设置UIImageView的图片及渲染模式。

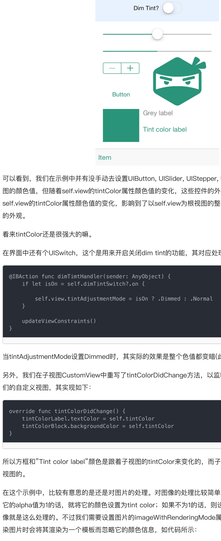
当我们点击"Change Color"按钮时，会执行以下的事件处理方法：

```
@IBAction func changeColorHandler(sender: AnyObject) {

    let hue = CGFloat(arc4random() % 256) / 256.0
    let saturation = CGFloat(arc4random() % 128) / 256.0 + 0.5
    let brightness = CGFloat(arc4random() % 128) / 256.0 + 0.5

    let color = UIColor(hue: hue, saturation: saturation, brightness: brightness, alpha: 1.
0)
    self.view.tintColor = color
    updateViewConstraints()
}

private func updateProgressViewTint() {
    self.progressView?.progressTintColor = self.view.tintColor
}
```

这段代码主要是随机生成一个颜色值，并赋值给self.view的tintColor属性，同时去更新进度条的tintColor值。

注：有些控件的特定组成部件的tint color由特定的属性控制，例如进度就有2个tint color：一个用于进度条本身，另一个用于背景。

点击"Change Color"按钮，可得到以下效果：



可以看到，我们在示例中并有没手动去设置UIButton, UISlider, UIStepper, UIImageView, ToolBar等子视图的颜色值，但随着self.view的tintColor属性颜色值的变化，这些控件的外观也同时跟着改变，也就是说self.view的tintColor属性颜色值的变化，影响到了以self.view为根视图的整个视图层次结构中所有子视图的外观。

看来tintColor还是很强大的嘛。

在界面中还有个UISwitch，这个是用来开启关闭dim tint的功能，其对应处理方法如下：

```
@IBAction func dimTintHandler(sender: AnyObject) {
    if let isOn = self.dimTintSwitch?.on {

        self.view.tintAdjustmentMode = isOn ? .Dimmed : .Normal
    }

    updateViewConstraints()
}
```

当tintAdjustmentMode设置Dimmed时，其实际的效果是整个色值都变暗(此处无图可追)。

另外，我们在子视图CustomView中重写了tintColorDidChange方法，以监听tintColor的变化，以更新我们的自定义视图。其实现如下：

```
override func tintColorDidChange() {
    tintColorLabel.textColor = self.tintColor
    tintColorBlock.backgroundColor = self.tintColor
}
```

所以方框和"Tint color label"颜色是跟着子视图的tintColor来变化的，而子视图的tintColor又是继承自父视图的。

在这个示例中，比较有意思的是还是对图片的处理。对图像的处理比较简单粗暴，对一个像素而言，如果它的alpha值为1的话，就将它的颜色设置为tint color；如果不为1的话，则设置为透明的。示例中的忍者头像就是这么处理的，不过我们需要设置图片的imageWithRenderingMode属性为AlwaysTemplate，这样渲染图片时会将其渲染为一个模板而忽略它的颜色信息，如代码所示：

```
var shinobiHead = UIImage(named: "shinobihead")
// 设置渲染模式
shinobiHead = shinobiHead?.imageWithRenderingMode(.AlwaysTemplate)
```

题外话

插个题外话，跟主题关系不大。

在色彩理论(color theory)中，一个tint color是一种颜色与白色的混合。与之类似的是shade color和tone color。shade color是将颜色与黑色混合，tone color是将颜色与灰色混合，它们都是基于Hues色调的。这几个色值的效果如下图所示：

一些基础的理论知识可以参考Hues, Tints, Tones and Shades: What's the Difference?或更专业的一些文章。

小结

如果我们想指定整个App的tint color，则可以通过设置window的tint color。这样同一个window下的所有子视图都会继承此tint color。

当弹出一个alert或者action sheet时，iOS7会自动将后面视图的tint color变暗，此时，我们可以在自定义视图中重写tintColorDidChange方法来执行我们想要的操作。

有些复杂控件，可以有多个tint color，不同的tint color控件不同的部分。如上面提到的UIProgressView，又如navigation bars, tab bars, toolbars, search bars, scope bars等，这些控件的背景着色颜色可以使用barTintColor属性来处理。

参考

- UIView Class Reference
- iOS7 Day–by–Day :: Day 6 :: Tint Color
- Appearance and Behavior
- Tints and shades
- Hues, Tints, Tones and Shades: What's the Difference?

论坛热帖 本周最热 | 本月最热

热门文章 本周最热 | 本月最热

1 苹果发布iOS 15.0.2：终于修复iP...
2 来作场！苹果新品发布会又要来...
3 只有裁员过年，有门这数月翻红...
4 泄露照片展现苹果 M1X Macboo...
5 缺芯减产？交货中断？信号门？...

Cocoa社区 元宇宙