NSLayoutAnchor详解

fever105 关注
2 2017.07.17 19:37:23 字数 1,937 阅读 7,739

NSLayoutAnchor详解

> NSLayoutAnchor作为iOS9的新特性，是对Auto Layout体系的完善和补充。如果你还不知道，就一起来看看吧😊。

A. 创建约束

Auto Layout（以下简称自动布局）的核心是约束，即NSLayoutConstraint。创建约束的方式有两种：1. Interface Builder；2. 代码。前者适用于xib和storyboard，比较简单；如果用代码搭建界面，后者是唯一选择。然而，原生API在这方面的表现简直可以用"惨绝人寰"来形容：

方法1：

```
// API 1
+ (NSArray<__kindof NSLayoutConstraint *> *)constraintsWithVisualFormat:(NSString *)fo
```

- 根据特定格式的字符串生成一组约束。这里的格式必须是"VFL语言"（Visual Format Language）：例如`@"H:|-20-[blueView]-20-|"`。看起来很厉害有没有😊，然而并没有什么卵用，因为：1. 约束越复杂，越难描述；2. 字符串越长，越容易出错；3. 只能描述相对约束，不能描述绝对约束（例如有视图A和B，可以表示A的宽是B的两倍，但无法表示A的宽是100pt）。

方法2：

```
// API 2
+ (instancetype)constraintWithItem:(id)view1 attribute:(NSLayoutAttribute)attr1 relate
```

- 使用NSLayoutAttribute定义约束。优点是直观，全面（相对/绝对约束都能创建）。但是，其缺点也很致命：啰嗦。每创建一个约束都要不厌其烦的填入7个参数，产生大量重复代码。其结果是代码通货膨胀：写了几百行，界面才刚刚搭好。

B. NSLayoutAnchor（布局锚点）

面对这种情况，市面上出现了许多第三方库，重新封装原生API，简化约束的创建过程。其中不乏Masonry这样优秀的开源库。

显然，Apple也意识到了这个问题，所以在iOS9引入了NSLayoutAnchor——即"布局锚点"的概念——专门用于创建约束。文档这样对其进行解释：

> The NSLayoutAnchor class is a factory class for creating NSLayoutConstraint objects using a fluent API. Use these constraints to programatically define your layout using Auto Layout.
>
> NSLayoutAnchor是一个专门用于创建NSLayoutConstraint的工厂类，其API简明，流畅。利用所创建的约束，以代码的形式搭建使用自动布局技术的界面。

B.1 NSLayoutAttribute与NSLayoutAnchor

上一节所提到的两个创建约束的方法中，方法2使用`NSLayoutAttribute`创建约束。layout attribute（布局属性）可以代表视图的方位或尺寸。属性可以划分为三类：尺寸，水平方向（x轴），垂直方向（y轴）：

尺寸	水平方向（x轴）	垂直方向（y轴）
宽高（width/height）	左右（left/right），前后（leading/trailing），水平中心（centerX）	上下（top/bottom），垂直中心（centerY），上下基准线（firstBaseline/lastBaseline）

伴随着NSLayoutAnchor的出现，UIView增加了一组属性，与上表中的NSLayoutAttribute一一对应：

```
@property(readonly, strong) NSLayoutYAxisAnchor *bottomAnchor
@property(readonly, strong) NSLayoutXAxisAnchor *centerXAnchor
@property(readonly, strong) NSLayoutYAxisAnchor *centerYAnchor
@property(readonly, strong) NSLayoutYAxisAnchor *firstBaselineAnchor
@property(readonly, strong) NSLayoutDimension *heightAnchor
@property(readonly, strong) NSLayoutYAxisAnchor *lastBaselineAnchor
@property(readonly, strong) NSLayoutXAxisAnchor *leadingAnchor
@property(readonly, strong) NSLayoutXAxisAnchor *leftAnchor
@property(readonly, strong) NSLayoutXAxisAnchor *rightAnchor
@property(readonly, strong) NSLayoutYAxisAnchor *topAnchor
@property(readonly, strong) NSLayoutXAxisAnchor *trailingAnchor
@property(readonly, strong) NSLayoutDimension *widthAnchor
```

不难看出，对于视图来说，layout anchor（布局锚点）与layout attribute（布局属性）的意义相同。既然布局属性可以分类，布局锚点也可以分类。

NSLayoutAnchor有三个子类，对应三种锚点类型：

NSLayoutDimension（尺寸）	NSLayoutXAxisAnchor（水平方向，x轴）	NSLayoutYAxisAnchor（垂直方向，y轴）
宽高（widthAnchor/heightAnchor）	左右（leftAnchor/rightAnchor），前后（leadingAnchor/trailingAnchor），水平中心（centerXAnchor）	上下（topAnchor/bottomAnchor），垂直中心（centerYAnchor），上下基准线（firstBaselineAnchor/lastBaselineAnchor）

特定锚点类型只能创建相应类型的约束：NSLayoutDimension创建尺寸约束；NSLayoutXAxisAnchor创建水平方向上的约束；NSLayoutYAxisAnchor创建垂直方向上的约束。根据类型不同，子类会重写父类方法，甚至增加新方法。

B.2 使用NSLayoutAnchor创建约束

假设有视图A，布局要求如下：

1. A高100pt；
2. A的宽度是父视图的一半；
3. A的垂直中心与父视图重合；
4. A的前边距离父视图的后边为-50pt。

使用NSLayoutAnchor创建约束，代码如下：

```
[A.heightAnchor constraintEqualToConstant:100.0].active = YES; // A高100pt
[A.widthAnchor constraintEqualToAnchor:A.superView.widthAnchor multiplier:0.5 constant
[A.centerYAnchor constraintEqualToAnchor:A.superView.centerYAnchor].active = YES; // A
[A.leadingAnchor constraintEqualToAnchor:A.superView.trailingAnchor constant:-50.0].ac
```

分析代码，可以得出结论：

1. 使用NSLayoutAnchor创建约束的流程：
   1. 确定要约束的视图；
   2. 选择相应的布局锚点；
   3. 根据约束类型选择合适的方法，创建约束。
2. 设置约束属性active，等同于调用UIView的`addConstraint:`或`removeConstraint:`；
3. 相较于传统API更简洁，避免重复；
4. 无法直接创建布局锚点。

C. UILayoutGuide（布局参照）

同时在iOS9引入的还有UILayoutGuide，其功能直接依赖NSLayoutAnchor实现。文档这样对其进行解释：

> The UILayoutGuide class defines a rectangular area that can interact with Auto Layout. Use layout guides to replace the dummy views you may have created to represent inter-view spaces or encapsulation in your user interface.
>
> UILayoutGuide用于定义一个可以参与自动布局的矩形区域。替代占位视图（dummy view）实现元素之间的等间距效果，或封装界面模块。

UILayoutGuide包含也一组NSLayoutAnchor属性，几乎与UIView一致。所以，二者的布局方式也一致，并且可以混合使用：即一个约束中的两个元素，一个来自视图，一个来自布局参照。

C.1 UILayoutGuide的作用

占位视图（dummy view）可以大大简化实现某些布局效果时的工作量。所谓占位视图，就是透明，没有任何内容的UIView。这种视图只定义一个矩形区域，在自动布局体系中起到占位作用。例如，若干个宽度不等的控件，要求横向排列它们，并且间距相等。此时必须使用占位视图作为间距，否则需要大量计算，十分麻烦。

占位视图也有缺点：一是需要创建和维护；二是大量的占位视图存在于视图结构中，会影响性能；二是占位视图可能会拦截触摸事件。为了解决上述问题，layout guide（布局参照）运应而生。

例如，视图属性`layoutMarginsGuide`就是通过UILayoutGuide实现，表示减去留白后的视图区域。类似的，视图属性`readableContentGuide`表示可读内容所占的区域。

C.2 如何使用UILayoutGuide

布局参照的使用流程如下：

1. 创建UILayoutGuide对象；
2. 将其加入视图（调用视图方法`addLayoutguide:`）；

   将layout guide加入视图，意味着以视图的坐标系定义矩形区域。
3. 添加约束，以定义位置和尺寸。

   同视图一样，所有约束必须构成明确，可满足的布局。

下面例子列举了布局参照的两个使用场景

C.3 使用场景1：多视图等间距布局

假设有视图A，B，C，宽度相同，水平排列，要求AB，BC之间的间距相等。代码如下：

```
UILayoutGuide *spaceAB = [[UILayoutGuide alloc] init]; // 创建间距AB
[self.view addLayoutGuide:spaceAB]; // 加入所在视图

UILayoutGuide *spaceBC = [[UILayoutGuide alloc] init]; // 创建间距BC
[self.view addLayoutGuide:spaceBC]; // 加入所在视图

[spaceAB.widthAnchor constraintEqualToAnchor:spaceBC.widthAnchor].active = YES; // 间距
[self.A.trailingAnchor constraintEqualToAnchor: spaceAB.leadingAnchor].active = YES; /
[self.B.leadingAnchor constraintEqualToAnchor: spaceAB.trailingAnchor].active = YES; /
[self.B.trailingAnchor constraintEqualToAnchor:spaceBC.leadingAnchor].active = YES; //
[self.C.leadingAnchor constraintEqualToAnchor:spaceBC.trailingAnchor].active = YES; //
```

C.4 封装界面模块

将复杂界面划分为一个个模块，可以降低布局复杂度。假需要将一个标签L（label）和一个文本框TF（text field）封装为一个模块。代码如下：

```
UILayoutGuide *container = [[UILayoutGuide alloc] init]; // 创建容器
[self.view addLayoutGuide:container]; // 加入所在视图

// 设置容器内容
[self.L.lastBaselineAnchor constraintEqualToAnchor:self.TF.lastBaselineAnchor].active
[self.L.leadingAnchor constraintEqualToAnchor:container.leadingAnchor].active = YES; /
[self.TF.leadingAnchor constraintEqualToAnchor:self.L.trailingAnchor constant:8.0].act
[self.TF.trailingAnchor constraintEqualToAnchor:container.trailingAnchor].active = YES
[self.TF.topAnchor constraintEqualToAnchor:container.topAnchor].active = YES; // 文本框
[self.TF.bottomAnchor constraintEqualToAnchor:container.bottomAnchor].active = YES; //

// 设置外部约束
[container.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor].active = YES
[container.trailingAnchor constraintEqualToAnchor:self.view.trailingAnchor].active = Y
[container.topAncor constraintEqualToAnchor:self.view.bottomAnchor constant:20.0].acti

```

D. UILayoutSupport

提到UILayoutGuide，不得不说UILayoutSupport。二者有些相似，刚开始比较容易混淆。这里要强调：

1. UILayoutSupport是一个协议，而非对象。仅被UIViewController属性`topLayoutGuide`和`bottomLayoutGuide`实现，用于定义视图内容与栏位（UIKit Bar）的显示层级关系。
2. 自iOS9开始，UILayoutSupport的功能通过NSLayoutAnchor实现，包含如下布局参照：

```
// 控制器布局参照的最高点
@property(readonly, strong) NSLayoutYAxisAnchor *topAnchor;
// 控制器布局参照的最低点
@property(readonly, strong) NSLayoutYAxisAnchor *bottomAnchor;
// 控制器布局参照的高度
@property(readonly, strong) NSLayoutDimension *heightAnchor;
```

显然，利用这个协议也可创建约束。下面代码将滚动视图的区域限制在上下栏位之间，即滚动视图的内容不会显示在栏位下方。（这里的self是一个UIViewController）

```
[self.scrollView.topAnchor constraintEqualToAnchor:self.topLayoutGuide. bottomAnchor];
[self.scrollView. bottomAnchor constraintEqualToAnchor:self.bottomLayoutGuide.topAncho
```

25人点赞 > iOS Dev ···

"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下

fever105 iOS dev on the way 关注
总资产8 共写了10.1W字 获得415个赞 共353个粉丝

写下你的评论...

全部评论 1 只看作者 按时间倒序 按时间正序

暗物质
3楼 10.26 15:48
👍
赞 回复

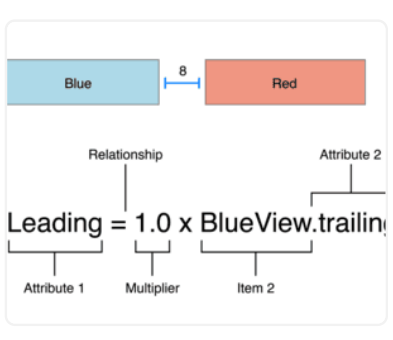
被以下专题收入，发现更多相似内容

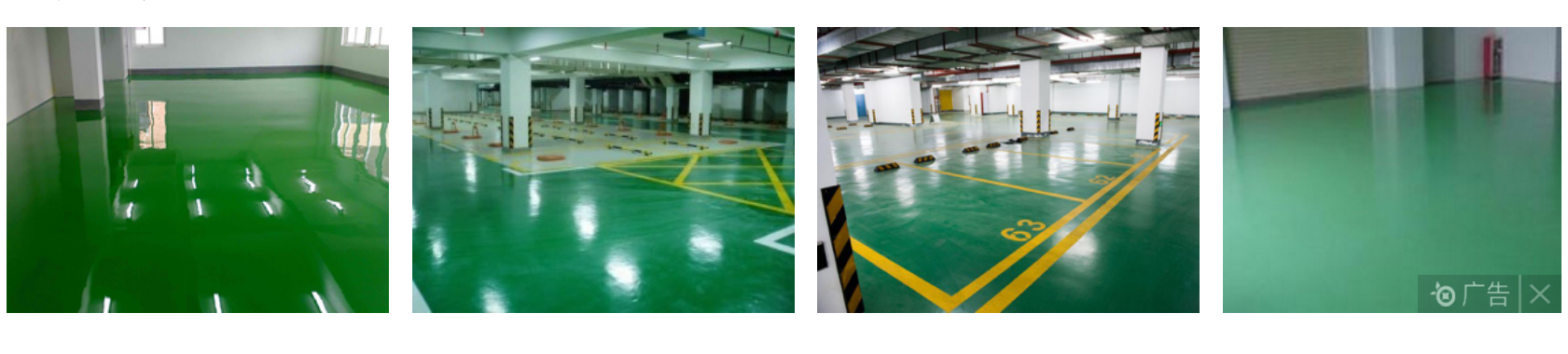
+收入我的专题 iOS-布局动画 iOS开发系列 IOS

推荐阅读 更多精彩内容 >

自动布局指南-Part 1：入门
翻译自“Auto Layout Guide”。 1 入门 1.1 理解自动布局 自动布局根据视图层级结构中视图上的...
lakerszhy 阅读 2,637 评论 3 赞 25

自动布局 Auto Layout (原理篇)
目录 0、前言 一、Auto Layout前世今生 二、Auto Layout基础知识 1.Auto Layout...
浮游lb 阅读 17,785 评论 3 赞 79





Android - 收藏集
Android 自定义View的各种姿势1 Activity的显示之ViewRootImpl详解 Activity...
passiontim 阅读 158,608 评论 24 赞 689

夜里花
摄自天水市天水湖，时在2017年3月24日20时许
百无一用知止 阅读 114 评论 0 赞 0

喜剧圆满，世界安好！
昨晚重温了一遍周星驰的喜剧之王。 突然发现， 随着年龄的增长， 对社会认知的增加， 每一次重温这部电影 都能找寻到...
沙子叔 阅读 69 评论 0 赞 0

写下你的评论... 评论1 赞25 ···

fever105 关注
总资产8

Asset Catalog 的理解与使用
阅读 3,285

记一次奇葩的JSON解析
阅读 357

推荐阅读

UIStackView使用介绍
阅读 225

07 ConstraintLayout-使用
阅读 1,006

08 MotionLayout-MotionScene
阅读 235

iOS UITableViewCell+UIStackView组合实现功能
阅读 517

Jetpack-Compose 学习笔记（二）—— Compose 布局你学会了么?
阅读 638