2016

iOS 无障碍编程指南

中国信息无障碍产品联盟&信息无障碍研究会

20160527

目录

翻译声明

发布日期：*20160530*

翻译部门：***信息无障碍研究会项目部***

本文档由信息无障碍研究会（ARA）翻译自苹果官方文档《iOS programming Accessibility guide》。除了本文档之外我们还做了网页版及中英文混排网页版。本翻译文档、网页版及中英文混排网页版的所有归中国信息无障碍产品联盟&信息无障碍研究会所有，转载、分享请注明出处，尊重他人的劳动成果。

由于水平和时间有限，翻译文档难免会出现错误，欢迎广大读者、开发者、测试人员等提出自己的宝贵意见。意见和建议可以发送邮件到下面的邮箱（info@siaa.org.cn）。在此诚挚的感谢您提出的宝贵意见！

介绍

iOS3.0 及之后的版本，视觉障碍用户可通过 VoiceOver 使用 iOS 设备。iOS 3.0 引入 UI 无障碍编程接口，可帮助开发人员构建能够让 VoiceOver 用户访问的应用。简单地说，VoiceOver 描述一个应用的用户界面，用户通过语音朗读以及声音，导航浏览该应用中的视图及控件。熟练使用 Mac OS X 的 VoiceOver 用户，能够快速上手，使用其移动设备上的 VoiceOver。

运行在 iOS3.0 及之后版本的 iPhone 应用，应该为 VoiceOver 用户构建无障碍环境。iOS 和 iOS SDK 通过以下方式支持此目标:

- 默认情况下，标准 UIKit 控件和视图是可访问的；
- 提供 UI 无障碍编程接口，此接口为 iPhone 应用程序构建可访问性定义了线形的流程；
- 提供相关工具，帮助开发者在代码中实现无障碍、在应用中测试无障碍。

如果您正在开发或更新一个 iPhone 应用程序，您应该阅读此文档来学习如何开发 VoiceOver 用户可访问的应用。

文档结构

该文档包含如下章节：

- “理解 iOS 上的无障碍特性”简要描述 VoiceOver 如何在设备上运行，同时介绍了可用于构建无障碍应用的编程接口及工具。
- “构建您 iOS 应用程序的无障碍特性”为构建 VoiceOver 用户可访问的应用程序提供深入指导。

辅助参考

本文档之前包含的内容：通过 VoiceOver 和 Accessibility Inspector 对您的应用进行无障碍体验 - 已经被转移到文档*在 iOS 中验证应用的无障碍特性*中。

理解 iOS 上的无障碍特性

从一开始，为了能够帮助所有人容易地使用设备，基于 iOS 的设备包含几个特性，包括可视化的语音邮件，邮件中的大字体，以及网页、照片、地图的缩放。加上以下的无障碍特性，视障、听障、肢体残障人士会更加容易的使用他们的设备:

- 缩放：放大整个设备屏幕。
- 黑底白色：可转换显示上的颜色。
- 单声道音频：把左右声道混合成一个可以在左右声道播放的单声道信号。
- 朗读自动文本：当用户在 iPhone 中打字时，朗读文本纠错与候选建议。
- 语音控制：允许用户通过语音命令拨打电话和控制 iPod 播放。

此外，视力障碍用户能够依赖 VoiceOver 使用他们的设备。

无障碍特性和 VoiceOver

VoiceOver 是苹果公司创新性的读屏技术，通过该技术，用户可以无需观看屏幕就可完全控制其设备。 VoiceOver 在应用中的用户界面与用户触控之间充当媒介，为应用中的元素与行为提供音频描述。当 VoiceOver 激活时，用户不用担心意外地删除联系人或者拨打了电话，因为 VoiceOver 会告知用户在用户界面中所处的位置、可以执行哪种操作，以及该操作会导致的结果。

当应用中与用户交互的所有元素都是可访问时，该应用是无障碍的。当一个界面元素恰当地告知它是无障碍元素时，该界面元素是可访问的。

但是，要想做到有效，一个无障碍用户界面元素还必须提供准确有效的信息，包括屏幕位置、名称、行为、值和类型。这些信息将被 VoiceOver 朗读给用户。iOS SDK 包含了一个编程接口和工具，帮助开发者确认应用中的用户界面元素是可访问的、可用的（更多信息，请参阅“iOS 的无障碍 API 与工具”）。

为什么您应该构建应用的无障碍特性

您应该让您的 iPhone 应用可以被 VoiceOver 用户无障碍使用，因为：

- 它能增加您的用户基数。您已经非常努力地创建了一个好的应用，这是一个好机会让它被更多用户使用。
- 允许用户在不看屏幕的场景下使用您的应用。视力损伤用户在 VoiceOver 的帮助下，能够使用您的应用。
- 有助于您遵循无障碍规范。许多管理部门撰写了无障碍规范，让您的 iPhone 应用能被 VoiceOver 用户访问，有助于符合规范。
- 这是正确的做法。

重要的是支持无障碍并不会影响您在 iPhone 应用上的创新，UI 无障碍编程接口允许您加入轻量经的功能层，但不会改变应用的外观，或者干扰其主要逻辑。

iOS 的无障碍 API 及工具

iOS 3.0 及之后的版本包含了 UI 无障碍编程接口，这个轻量级的 API 可以让 VoiceOver 从一个应用中提取所需要的信息，用以描述用户界面，帮助视力障碍用户使用应用。

UI 无障碍编程接口是 UIKit 的一部分，默认通过标准 UIKit 控件和视图来实现。也就是说，当使用标准控件和视图时，大部分涉及应用的无障碍工作 已经完成。根据应用中自定义的程度，简单地为无障碍用户界面元素提供准确和有用的描述，即能为其打造无障碍特性。

iOS SDK 也提供相应的工具协助开发者打造应用的无障碍特性：

- 在设计 nib 文件时，Interface Builder 观察器面板提供了一个简单的方法用以添加描述性的无障碍信息。了解如何使用，请查看在 Interface Builder 中添加自定义属性信息。
- 在应用里，那些嵌入在用户界面中的无障碍信息，可以在 Accessibility

Inspector 中显示，并且允许开发者运行 iOS Simulator 来验证这些信息。了解如何检验应用中的无障碍信息，请看在 iOS 模拟器中使用 Accessibility Inspector 调试无障碍特性。

另外，开发者可以使用 VoiceOver 测试应用的无障碍特性。了解如何使用 VoiceOver 测试您的应用，请看通过您设备的 VoiceOver 测试无障碍特性。

UI 无障碍编程接口

UI 无障碍编程接口由两个非正式协议、一个类、一个函数，以及少数常量构成。

- `UIAccessibility` 非正式协议：实现 `UIAccessibility` 协议的对象，可以报告无障碍状态（即他们是否是可访问的），并且提供其自身的描述信息。标准 UIKit 控件和视图已默认实现了 `UIAccessibility` 协议。

`UIAccessibilityContainer` 非正式协议：此协议允许 `UIView` 的子类作为分离元素，具有无障碍特性构建部分或者全部对象。当对象包含在视图中，其本身又不是 `UIView` 的子类，导致不能被自动地无障碍访问，此时，该协议非常有用。

- `UIAccessibilityElement` 类：这个类定义了一个能够通过 `UIAccessibilityContainer` 协议返回的对象。当一个元素不是自动地无障碍访问，可以创建一个 `UIAccessibilityElement` 实例来展示它，例如一个非 `UIView` 继承对象，或者一个不存在的对象。

`UIAccessibilityConstants.h` 头文件：这个头文件定义了某些常量，这些常量既可以描述一个可展示的无障碍元素的特征，也可以描述应用发布的通知。

无障碍属性

用来描述无障碍用户界面元素的属性也是 UI Accessibility API 的核心构成。

当用户访问控件和视图，或与之交互时，VoiceOver 会将无障碍属性信息告知给用户。

属性是开发者最有可能用到的编程接口组件。因为它封装了用于区分不同控件和视图的信息。标准 UIKit 控件和视图，开发者只需确保默认属性是适合应用的。对于自定义的控件和视图来说，开发者可能需要提供大部分的属性信息。

UI Accessibility 编程接口定义了如下属性：

- **Label(标签)**：一个简短的本地化词语或短语，用于简洁地描述控件或视图，但不用来定义元素类型。例如“添加”或“播放”。
- **Traits(特质)**：一个或多个独立特质的组合。每个特质描述元素状态、行为、用途中的某个方面。例如，某个元素表现为键盘按键且当前被选定，这个元素可以使用键盘按键(Keyboard Key)和选中(Selected)的组合特质。
- **Hint(提示)**：简要的、本地化短语，用于描述某元素动作的结果，例如：“添加标题”或者“打开购物清单”。
- **Frame(框架)**：元素的框架是元素在屏幕上的坐标，是由 `CGRect` 结构体提供，指明了元素在屏幕上的位置和大小。
- **Value (值)**：一个元素的当前值，当值无法使用标签(label)呈现时。例如，一个滑动条的标签(label)可以是”速度”，但是它当前的值(value)可能是“50%”。

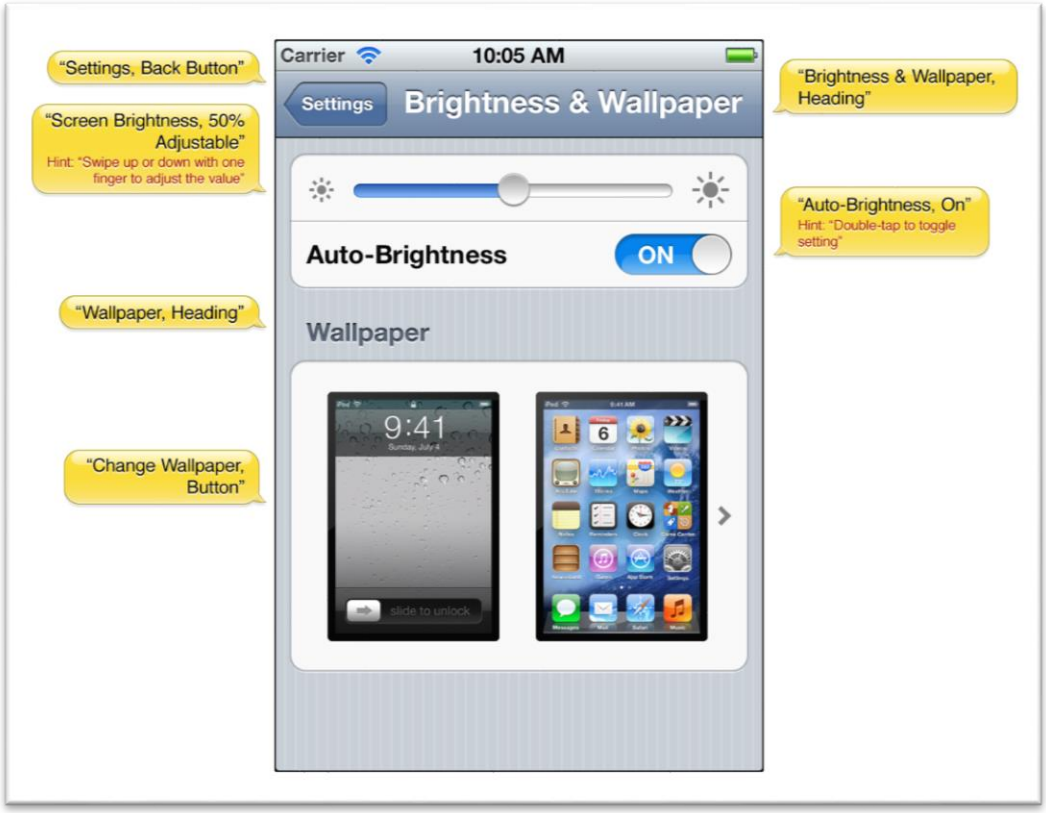
无障碍元素需要为属性提供内容，无论这些内容是被默认提供还是由开发者提供。无障碍元素应该为框架和标签属性提供内容。框架属性是必须的，因为无障碍元素必须能够报告它在用户界面中的位置。（注意：一个从 `UIView` 继承的对象，默认是包含了框架属性的）。标签(label)属性也是必须的，因为它包含了无障碍元素的名称或描述，这些信息是可以被 VoiceOver 朗读出来的。

如果这些属性不适用于元素，无障碍元素就无需为提示(hint)和特质(traits)属性提供内容。例如，某个不能触发动作的元素，无需提供提示(hint)。

只有当元素内容是可改变的且不能只通过标签(label)描述时，无障碍元素才为其提供信息。例如，文本域包含一个邮件地址，可能有个标签描述“邮件地址”，然而其内容是基于用户输入的，并且内容通常的形式为“username@address”。图

1-1 展示了一些 VoiceOver 应该提供的信息。

图 1-1 VoiceOver 朗读由无障碍元素提供的信息



VoiceOver 用户在使用您的应用时，是依赖听觉获取标签和提示。因此，确保应用中的每一个无障碍元素提供准确全面的描述是极其重要的。标准 UIKit 控件和视图的默认值信息通常是准确的，但是开发者还是应该检查确保其正确。对于自定义控件和视图，开发者可能不得不亲自提供部分或者全部信息。请参看“标记好用的标签(label)和提示(hint)”中的指导，了解如何提供这些信息。

构建您 iOS 应用程序的无障碍特性

要做到可用性，一个 iPhone 应用必须将用户界面元素中的相关信息提供给 VoiceOver 用户。更高层次上，这意味着您应该确保：

- 每个能够与用户交互的用户界面元素都是无障碍的，包括那些仅仅提供信息的元素，比如，静态文本，以及执行动作的控件。
- 所有可以无障碍使用的元素都包含准确并且有用的信息。

除了上述原则外，开发者还可以做一些事情来提升 Voiceover 用户对表格视图的体验，并确保应用中的动态元素始终无障碍。

让用户界面元素具有可访问性

正如“无障碍与 VoiceOver”所提及的，用户界面元素如果能告知其为无障碍元素，那么该元素是可访问性的。尽管可访问并不能足够让用户界面元素对 Voiceover 用户可用，但这是应用程序进行无障碍化过程的第一步。

也如“iOS 的无障碍 API 及工具”中所述，标准的 UIKit 控件和视图是自动具有可访问性的。如果只使用了标准的 UIKit 控件，开发者可能不需要太多其他工作来确保应用的无障碍。在这种情况下，下一步工作是确保应用中这些控件提供的默认属性信息是有意义的。详情参见“提供准确和帮助性的属性信息”。

如果创建一个自定义视图来展示信息或是与用户进行交互，开发者必须确保这些视图的无障碍特性。在完成这些之后，开发者还需要确保这些视图提供的无障碍信息可以帮助到用户。（参见“提供准确和有用的属性信息”）。

从无障碍的角度来说，自定义视图可能是一个独立的视图也可能是一个容器视图。*独立视图(individual view)*不包含其他需要无障碍化的视图。例如，一个自定义 UIControl 子类会显示一个图标，并且具有按钮的行为，但是除了按钮本身，其并不包含任何可以与用户进行交互的元素。请阅读“让自定义的独立视

图具有无障碍特性”了解如何让一个独立视图具有无障碍特性。

另一方面，*容器视图*包含其他可以和用户进行交互的元素。例如，在一个自定义 UIView 子类中绘制一个几何图形，用户可以与该图形呈现的元素进行交互，这与容器视图不同。容器视图内的独立元素不能自动获得无障碍特性（因为它们不是 UIView 的子类），也不会提供任何无障碍信息。请阅读“让自定义容器视图的内容具有无障碍特性”了解如何让容器视图的内容具有无障碍特性。

让自定义的独立视图具有无障碍特性

如果应用包含一个需要与用户进行交互的自定义独立视图，开发者必须确保这个视图的无障碍特性。（回想一下，一个独立视图是指：一个视图不包含其它可与用户交互的视图。

除了使用 Interface Builder 构建自定义视图的无障碍特性，还可以通过两种编程方法让自定义独立视图具有无障碍特性。第一种方法是：在实例化自定义视图的代码中设置其无障碍状态。如下面的代码片段所示：

```
@implementation MyCustomViewController
- (id)init
{
   _view = [[[MyCustomView alloc] initWithFrame:CGRectZero] autorelease];
   [_view setIsAccessibilityElement:YES];
/*  此处设置属性*/
}
```

另一种方法是在您的自定义子类实现中，UIAccessibility 协议中使用 isAccessibilityElement。如下面的代码片段所示：

```
@implementation MyCustomView
   /*此处使用属性方法 */
 - (BOOL)isAccessibilityElement
{
    return YES;
}
```

注意： 在这两段代码中，需要设置无障碍属性的代码都有明确注释。展示如何实现的完整代码，请阅读“程序化定义自定义属性信息”。

让自定义容器视图的内容具有无障碍特性

如果应用中展示了一个自定义视图，包含其他与用户交互的元素，则需要让包含元素分别具有无障碍特性。同时，开发者必须让容器视图本身不再具有无障碍特性，因为用户是与容器中的内容进行交互，不是和容器本身交互。

为此，您自定义的容器视图应该实现 `UIAccessibilityContainer` 协议。这个协议会定义一些方法让被包含元素存放于一个数组中。

如下代码片段展示了一个自定义容器视图的部分实现。注意，这个容器视图只会在调用 `UIAccessibilityContainer` 协议时，才会创建无障碍元素数组。因此，如果 iPhone 的无障碍状态并没有被激活，那就不会创建该数组。

表 2-1 实现自定义容器中的内容做为独立的无障碍元素

```
@implementation MultiFacetedView
- (NSArray *)accessibleElements
{
    if ( _accessibleElements != nil )
    {
        return _accessibleElements;
    }
    _accessibleElements = [[NSMutableArray alloc] init];

    /*  创建一个无障碍元素展示第一个被包含的元素，并初始化它做为 MultiFacetedView 的组件*/
    UIAccessibilityElement *element1 = [[[UIAccessibilityElement alloc] initWithAccessibilityContainer:self] autorelease];

    /* 此处设置第一个包含元素的属性*/
    [_accessibleElements addObject:element1];

    /* 为第二个包含元素执行相同步骤*/
    UIAccessibilityElement *element2 = [[[UIAccessibilityElement alloc]
```

```
initWithAccessibilityContainer:self] autorelease];

        /*此处设置第二个包含元素的属性*/
        [_accessibleElements addObject:element2];

        return _accessibleElements;
    }

    /* 容器本身并不具有无障碍特性，所以 MultFacetedView 在 isAccessiblityElement 中应该返回 NO*/
    - (BOOL)isAccessibilityElement
    {
        return NO;
    }

    /* 如下方法实施 UIAccessibilityContainer 协议 */
    - (NSInteger)accessibilityElementCount
    {
        return [[self accessibleElements] count];
    }

    - (id)accessibilityElementAtIndex:(NSInteger)index
    {
        return [[self accessibleElements] objectAtIndex:index];
    }

    - (NSInteger)indexOfAccessibilityElement:(id)element
    {
        return [[self accessibleElements] indexOfObject:element];
    }
    @end
```

提供准确和有用的属性信息

提供准确和有用的属性信息有以下两个步骤：

- 制作简洁、准确、有用的信息
- 确认应用中无障碍元素所给出的内容是正确的

如果使用自定义的视图，开发者必须为它们提供适当的属性信息。请参“阅制作有用的标签和提示”、“创建提示指南”和“定义恰当的特质”。

即使只是用标准的 UIKit 控件和视图，也可能找到比默认提供的属性信息更适合应用环境的内容。更多信息请阅读“优化默认属性信息”。

无论是标准还是自定义的 UI 元素，如果需要提供或者更改其无障碍属性，开发者都可以使用 Interface Builder（请参阅“在 Interface Builder 中自定义属性信息”）或者编程化处理（请参阅“编程化处理自定义属性信息”）。

优化默认属性信息

作为标准 UIKit 控件和视图内建无障碍特性的一部分，iOS 也为这些元素提供了默认属性信息，这些信息可以向 VoiceOver 用户描述这些元素。大多数情况下，应用如果使用标准控件和视图，这种信息是合适的。但是，自定义属性信息可以增强 VoiceOver 用户的体验。

如果用一个标准 UIKit 控件和视图显示系统提供的图标和标题，开发者首先需要确认是以预期的目的使用它（请参阅“*iOS 人机界面指南*”了解更多信息）。然后认真考虑，在应用中，默认的标签属性是否准确地传达结果，如果没有，考虑提供一个提示属性。

举例来说，如果在 `UIBarButtonItem` 对象中，开发者使用系统提供的加（+）图标，在导航栏中放置一个添加按钮，它会自动包含一个默认的标签属性：添加。如果用户可以明确地知道每次点击这个按钮会添加哪个项目，那就不需要提供提示属性。但是如果可能产生误解，开发者应该在应用中提供一个自定义的

提示，用以描述使用该按钮的效果，比如“添加一个账户”或者“添加一个评论”。

如果您在标签的 UIKit 视图中显示一个自定义的图标或者图像，比如 `UIBarButtonItem` 对象，需要提供一个自定义的标签属性用以描述它。

制作有用的标签和提示

当 VoiceOver 用户使用应用时，他们依赖 VoiceOver 的描述了解应用的作用以及如何使用，由于这些描述代表了大量的 VoiceOver 用户对于应用的体验，所以有必要尽可能让其准确有用。该指南会帮助开发者创建有用的标签和提示，让应用对于残障人士来说更加简单易用。

创建标签指南

标签属性定义了用户界面中的元素，无论是标准的还是自定义的，每个无障碍用户界面元素都必须为标签属性提供内容。

> **注意：**表格中的一行也可以包含一个标签属性。然而，创建表格行标签的方法和其他类型控件和视图的方法不一样。具体的创建方法参见“优化表格视图无障碍特性”。

判定标签应该传达什么内容的一个好办法，就是思考视觉用户通过视觉能够看到什么。如果用户界面设计地很好，视觉用户应该可以通过阅读标题或理解图标，了解在当前应用中控件和视图的作用。这些就是 VoiceOver 用户可从标签属性获取到的信息。

如果提供一个自定义的控件和视图，或者开发者在标准控件或视图中展示一个自定义的图标，需要提供如下的标签(label)：

- **非常简要地描述元素：**理想化的标签(label)是由一个单词构成，比如，添加、播放、删除、搜索、收藏、音量。

努力设计应用，尽量用一个单词标识一个元素，并能使它在当前环境下的作

用显而易见。然而，有时候可能需要使用一个简单的短语才能标识一个元素。遇到这种情况，创建一个短的句子，比如“播放音乐”、“添加姓名”或者“添加事件”。

- **不要包含控件或者视图的类型：**这类信息被包含在元素的特质属性中，应永远不重复出现在标签中(label)。

例如，如果在一个添加按钮的标签中，包含控件类型，VoiceOver 用户每次访问该控件的时候都会听到“添加按钮按钮”，这种体验非常讨厌，可能刺激用户停止使用您的应用。

- **单词首字母大写：**可以帮助 VoiceOver 用适当的音调读取标签。

- **不要在结束处用句号：**标签不是一句话，因此不应该使用句号结束。

- **本地化：**本地化无障碍属性中的所有字符串，一定能让应用被更广泛的受众使用。通常情况，VoiceOver 是以用户设置的语言进行朗读。

创建提示指南

提示属性描述的是：在控件或视图上执行一个动作的结果。当元素标签无法明确指出动作结果的时候，应该给出一个提示。

例如：在应用中提供一个播放按钮，按钮周边环境可让用户很容易明白，他们按下按钮时会发生什么。但是，如果允许用户通过轻敲歌曲列表中歌名播放歌曲，开发者应该提供一个提示用以描述结果，因为列表项的标签仅介绍自己的属性（此例中指歌名），用户点击时不知道会发生什么。

> **注意：** VoiceOver 的用户，在其设备上，通过 VoiceOver 设置中的选项，可以选择是否听取可获得的提示。默认情况下，朗读提示是开启的。

如果用户在控件或者视图上的动作结果，不能通过标签明确的表明，创建一个这样的提示：

- **尽量简单介绍结果：**尽管没有多少控件和视图需要提示，也应尽可能的提供

简短的介绍。避免了用户还没使用元素，就浪费大量时间去听取相关信息。

但是，不要为了提示的简短，而丧失语法的清晰和优雅。例如，将“添加一个城市”，改为“添加城市”，这并不意味着提示的长度简短了，反而听起来更别扭，并且表意不够清晰。

- **动词开头，省略主语：**确认使用第三人称单数的动词形式。例如“Plays,”而不是强制的，如“Play.”。要避免使用命令语式，因为其放在提示中，像是一条命令。比如，不要告诉用户“Play the song”，要告诉用户按下元素会“Plays the song.”。

要想找到正确的用词，想象一下，您正在为一位朋友介绍一个控件的用法，可能会这样说：“点击这个控件播放歌曲”。通常该句话的第二个短语（此例中为“播放歌曲”）可以作为一个提示。

- **大写字母开头，句号结尾：**即使提示是一个短语，不是一个句子，也要在提示末尾使用句点，这样有助于 VoiceOver 以适当的语气进行播报。

- **不要包含行为的名称或手势：**一个提示不用告诉用户如何执行该动作，它告诉用户的是当行为发生时会发生什么。因此，不要创建类似于“存储保存您的修改”或者“后退返回到之前的屏幕”的提示。

这极其重要，因为 VoiceOver 用户使用特定手势与应用进行交互。如果在一个提示中命名了不同的手势，会造成一定的混淆。

- **不要包含控件和视图的名称：**用户从标签属性里得到信息，所以开发者不应该在提示中重复相关信息。因此，不要创建类似于这样的提示“存储保存您的修改”或者“后退返回到之前的屏幕”。

- **不要包含控件和视图的类型：**用户已经知道控件表现为一个按钮或者搜索域，因为该信息可从元素特质获取。因此，不要创建类似于“按钮用于添加名称”或者“滑动用于控制比例”的提示。

- **本地化：**与无障碍标签类似，提示应该可以读取用户的偏好语言。

定义恰当的特质

特质属性包含一个或者多个独立特质，共同描述一个无障碍用户界面元素的行为。由于一些独立的特性可以组合起来一起描述一个单独元素，因此一个元素的行为可以被准确地表述。

注意： 独立的特质的组合通常是使用或运算符。代码实例以外，本文档的短语“组合”无特别指定组合方法。

一个标准的 UIKit 控件，如一个按钮或者文本域，在特质属性中会提供默认的内容。如果在应用中仅用标准 UIKit 控件（特指没有用任何方式自定义其行为），这些控件的特质属性不会有任何变化。

如果是自定义行为的标准控件，开发者可能需要将某个新的特质与该控件默认的特质组合。如果创建自定义的控件视图，开发者需要提供该元素特质属性的内容。

UI 无障碍编程接口定义了 12 个独立的特质，其中某些是可以被组合的。某些特质是通过特定控件类型的行为标识一个元素（如按钮）或者对象类型（如图像）。其它特质通过描述特定元素的展示行为标识元素，如播放声音的功能。

如下的特质可用于在应用中标识元素：

- 按钮(Button)
- 链接(Link)
- 搜索域(Search Field)
- 键盘按键(Keyboard Key)
- 静态文本(Static Text)
- 图像(Image)
- 播放声音(Plays Sound)
- 选中(Selected)
- 总结元素(Summary Element)
- 常态更新(Updates Frequently)

- 不可用(Not Enabled)
- 空(None)

通常，控件相对应特质能够结合描述行为的特质，例如，“按钮”特质可能和“播放音乐”特质用以表述一个自定义控件，其表象是一个按钮，点击即可播放音乐。

大多数情况下，应该考虑特质对应特定控件，尤其是，按钮、链接、搜索域、键盘键的特质，是互斥的。这时，不应该使用超过一个特质去表述应用中的某个元素，如果此元素有其它行为，可以将第一个特质和某一个行为特质结合。

例如，假设显示在应用中的一个图像，其响应的是，当用户点击时会在 iPhone 中的 Safari 打开一个链接，开发者应该通过结合图像和链接的特质表述该元素。另如，一个键盘键当点击时修改其它键盘键，开发者应该通过结合键盘键和选中的特质标识该元素。

了解如何标识控件的案例，可以用 Accessibility Inspector 看在标准控件中已被设定的默认特质。更多关于如何使用 Accessibility Inspector，请参阅通过 Accessibility Inspector 在 iOS Simulator 无障碍排错”。

在 Interface Builder 中自定义属性信息

安装 iOS SDK 3.0 后，获得的 Interface Builder 版本包含帮助应用达到无障碍特性的功能。如果应用包含标准 UIKit 的控件和视图，可通过 Interface Builder 做所有的无障碍工作。

使用 Interface Builder，可以设置元素的无障碍状态，并且为标签(label)和特质(traits)属性提供自定义内容。为此，在 nib 文件中选择用户界面元素，并且打开 Identity inspector，在 inspector 中的 Accessibility 处，应该可以看到如图 2-1 所示内容：

如图 2-1 所示，用于 nib 文件中的标准文本域默认是无障碍的，并且标签(label)、提示(hint)、特质(traits)的属性中包含默认的信息，（注意，文本域显示占位符文本，默认的标签就是占位符文本），在 Identify inspector 中可以提供新的信

息改变任何默认值。如在图 2-2 所示，（图 2-2 显示的也是 Accessibility Inspector 如何显示文本域中的无障碍信息。参阅通过 Accessibility Inspector 在 iOS Simulator 无障碍排错）。

图 2-1 显示在 Interface Builder 中的标准文本域的默认无障碍信息

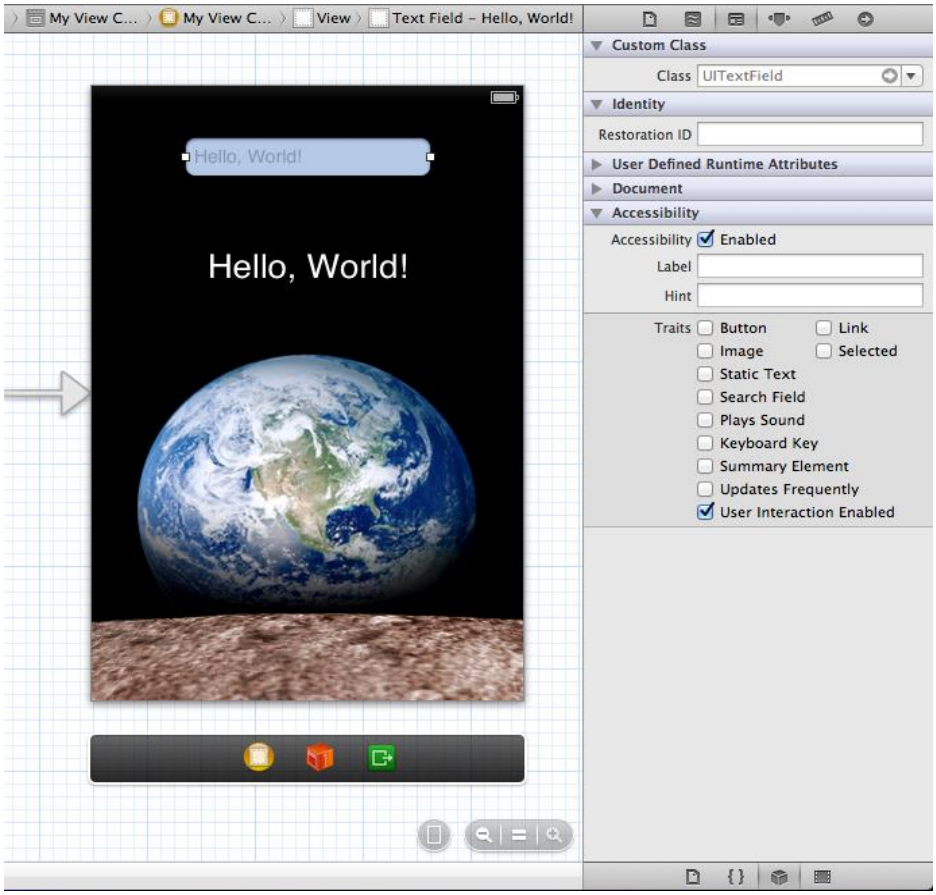
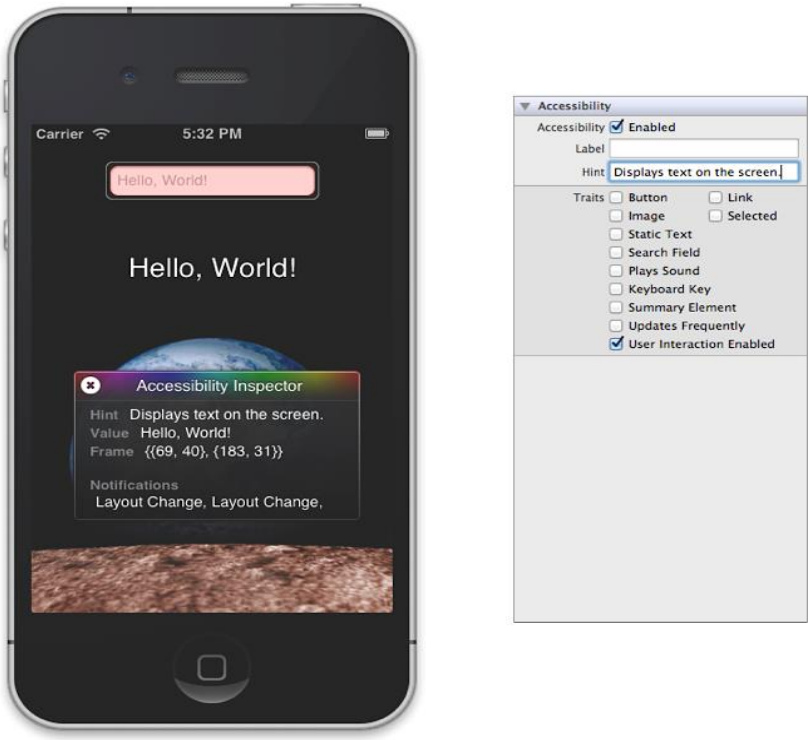


图 2-2 在 Interface Builder 中提供无障碍信息



编程化处理自定义属性信息

如果愿意，开发者可以通过编程为属性提供自定义信息。如果开发者完全不用 Interface Builder 或者更喜欢在代码中生成视图，这可能是开发者想用的方法。

在“让自定义的独立视图具有无障碍特性”中介绍过，开发者可以在视图子类或在代码里实例化视图设置无障碍信息。两种方法都是有效的，但是为什么开发者可能需要在子类中实现属性方法而不是通过代码实例化实现，原因是：如果显示数据是动态的，或者频繁改变，如日期，开发者应该实现子类方法返回需要的刷新数据。在这些情况下，如果仅在实例化子类时设置属性，返回的数据可能是过期的。

此章节的代码片段是基于“让自定义的独立视图具有无障碍特性”，包含一些属性特有的方法。例如：如果开发者想在子类中实现无障碍方法，将会用到类似表 2-2 的代码：

表 2-2 在自定义子类实现中提供属性信息

```
@implementation MyCustomView
- (BOOL)isAccessibilityElement
{
    return YES;
}

- (NSString *)accessibilityLabel
{
    return NSLocalizedString(@"MyCustomView.label", nil);
}
 /* 这个自定义控件的表像为一个按钮*/
- (UIAccessibilityTraits)accessibilityTraits
{
    return UIAccessibilityTraitButton;
}

- (NSString *)accessibilityHint
{
    return NSLocalizedString(@"MyCustomView.hint", nil);
}
@end
```

如果您要在代码中使用 `UIAccessibility` 协议的属性设定方法，实例化自定义视图，将会用到类似于表 2-3 中的代码。

表 2-3 在代码中提供属性信息，实例化自定义的子类对象

```
@implementation MyCustomViewController
- (id)init
{
  _view = [[MyCustomView alloc] initWithFrame:CGRectZero];

  [_view setIsAccessibilityElement:YES];
```

```
    [_view setAccessibilityTraits:UIAccessibilityTraitButton];
    [_view setAccessibilityLabel:NSLocalizedString(@"view.label", nil)];
    [_view setAccessibilityHint:NSLocalizedString(@"view.hint", nil)];
}
```

优化表格视图无障碍特性

如果应用中展示了一个表格，每个单元格中不只包含文本（或除了文本）条目，开发者可以做某些事情让其无障碍。同样地，如果表格视图中每行展示的信息不止一条，那可以把这些信息整合到一条易于理解的标签中，以优化 VoiceOver 用户的体验。

注意: 如果表格单元中包含任何的标准表格视图元素，例如信息展示器、详细信息展示按钮、或删除控件，开发者不需要做任何事就能让这些元素具有无障碍特性。但是如果表格单元包括其它控件类型，比如一个开关或者滑动条，开发者需要确认这些元素有恰当的无障碍特性。

如果应用中的表格单元包含不同元素的混合，检查用户是与单元进行交互还是与单元里的独立元素进行交互。如果用户需要访问单元里的独立元素，开发者应该：

- 使每个独立元素具有分离的无障碍特性。
- 确保表格单元自己*不具有*无障碍特性。
- 简要介绍单元格中的全局内容，并用此介绍做为单元格的标签(label)属性。注意，该情况下，标签(label)被视为单元格中的一个无障碍元素。

开发者可能已经意识到表格单元中可能会包含多个项目，如文本和控件，被 UI 无障碍编程接口所定义，匹配容器视图的标准。然而，开发者不必将单元格标识为容器视图，或实现 `UIAccessibilityContainer` 协议中的任何方法，因为表格单元被自动指派为一个容器。

如果表格中包含单元格，这些单元格在离散组块中提供信息，开发者应该考虑从这些标签属性的组块中组合信息。当实现该功能，VoiceOver 用户可以仅使

用一个手势就能获取单元格内容的意义，而不需要单独访问每个独立块中的信息。

用一个很好的例子解释该功能的效果：内建股票应用，它不用单独字符串去提供公司名、当前股票价格和价格波动，而是组合到一个标签(label)中，这样听起来类似：“苹果公司，432.39 美元，涨幅 1.3%。”。注意标签中的逗号，当结合多个片段的时候，可以使用逗号，这样 VoiceOver 就会在每个逗号短暂停留，让用户更容易理解信息。

此处的代码片段展示了如何组合两个独立元素中的信息到一个单独标签：

```
@implementation CurrentWeather
/*  该视图提供了天气信息，它包涵一个城市子视图和一个温度子视图，每个视图有独立标签  */
- (NSString *)accessibilityLabel
{
    NSString *weatherCityLabel = [self.weatherCity accessibilityLabel];
    NSString *weatherTempLabel = [self.weatherTemp accessibilityLabel];

    /*  结合城市和温度的信息，VoiceOver 用户可以使用一个手势来获取天气信息*/
    return [NSString stringWithFormat:@"%@, %@", weatherCityLabel, weatherTempLabel];
}
@end
```

让动态元素具有无障碍特性

如果应用中的界面元素会动态改变，开发者需要确认元素提交的无障碍信息是准确及时的。当应用屏幕布局改变发生时，需要发送一个通知，这样 VoiceOver 能够帮助用户导航新布局。UI 无障碍编程接口提供两个通知类型，可以用在这类屏幕改变事件发生的情况下。（了解这些通知，请在 *UIAccessibility 协议参考*

中的“通知”。）

如果用户界面元素根据不同应用条件下有不同的状态，那开发者需要在代码中加入返回元素所有状态对应无障碍信息的逻辑。由于这些改变是用户的动作的结果，所以最好把逻辑添加到子类实现中，而不是实例化子类的代码中。

下面的代码展示了如何处理动态状态的改变以及如何在屏幕布局发生改变时发送通知。代码中的 `UIView` 子类的行为类似一个自定义键盘按键。按键的无障碍标签会根据实例是否代表一个字母或其他类型的字符，和 shift 键是否按下来进行改变。按键也会根据实例内容和选中状态返回不同的无障碍特性。注意，表 2-4 的代码假设会有许多方法查询键盘的状态：

表 2-4 为当前情况返回正确的无障碍信息，并且发送布局改变通知

```
@implementation BigKey
- (NSString *)accessibilityLabel
{
    NSString *keyLabel = [_keyLabel accessibilityLabel];
    if ( [self isLetterKey] )
{
        if ( [self isShifted] )
        {
            return [keyLabel uppercaseString];
        }
        else
        {
            return [keyLabel lowercaseString];
        }
    }
    else
    {
        return keyLabel;
    }
}
```

```
- (UIAccessibilityTraits)accessibilityTraits
{
    UIAccessibilityTraits traits = [super accessibilityTraits] |
UIAccessibilityTraitKeyboardKey;

    /*是否这是一个 shift 键，并被按下，用户需要知道 shift 当前是有效的
*/
    if ( [self isShiftKey] && [self isSelected] )
    {
        traits |= UIAccessibilityTraitSelected;
    }

    return traits;
}

- (void)keyboardChangedToNumbers
{
    /* 此处代码，执行一个数字键盘的变化 */

    /*发送一个屏幕布局改变的通知。*/

    UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification,
nil);
}
@end
```

让非文本数据具有无障碍特性

有时应用会展示那些不是自动兼容无障碍方法的数据。例如，如果展示一个

图像，开发者应该在它的无障碍标签中提供它的描述，这样 VoiceOver 用户就可以理解图像传达的信息。或者，如果使用图形方式提供信息，比如显示星级的评价系统，开发者应该确认无障碍标签传达了图形所表达的意思。

如下的代码段使用自定义视图样例展示了与评分条目对等的星级条目。该代码展示了视图如何根据绘制的星星数量返回一个恰当的无障碍标签。

```
@implementation RatingView
/* Other subclass implementation code here. 在此处实现其它子类的代码。*/

- (NSString *)accessibilityLabel
{
    /*_starCount 是一个实例化变量，包含了画多少颗星星*/
    NSInteger starCount = _starCount;
    if ( starCount == 1 )
    {
        ratingString = NSLocalizedString(@"rating.singular.label", nil); // 此处，
ratingString 是星数
    }
    else
    {
        ratingString = NSLocalizedString(@"rating.plural.label", nil); // 此处，
ratingString 是星数
    }

    return [NSString stringWithFormat:@"%d %@", starCount, ratingString];
}
@end
```

文档修订记录

这个列表描述了《Accessibility Programming Guide for iOS 》的修改记录。

日期	注释
2012-02-16	新增缺失返回声明的代码清单。
2011-07-20	修改少量错误。
2010-07-07	把标题从"Accessibility Programming Guide for iPhone OS."修改为现在的标题。
2009-05-29	添加新的文件介绍如何使一个 iPhone 应用程序让残疾用户可访问。