

iOS 人机界面准则

郎启旭 · LANGQIXU.COM



译者按

本文译自 Apple.Inc 推出的设计文档《iOS Human Interface》（2013-10-22 版本），但未取得官方翻译授权。

由于经验不足、水平有限，翻译中难免存在错误和疏漏之处，欢迎来信指正。后续如有更新，也会在我的微信公众号上告知（搜索微信号：langnote），欢迎关注。

郎启旭

2014 年 1 月 25 日



邮箱：langqixu@gmail.com | 知乎：zhihu.com/people/langqixu | 博客：langqixu.com

目录

用户界面设计基础 8

为 iOS 7 而设计 8

依从内容 9

清晰呈现 11

纵深传达 15

iOS 应用解析 21

启动和停止 23

立即启动 23

随时准备停止 24

布局 27

导航 30

模态情境 33

交互和反馈 35

用户对标准手势了如指掌 35

交互性元素引人触控 37

反馈增进理解 41

信息输入轻松容易 42

动画 43

品牌化 45

色彩和字体 47

色彩增进沟通 47

文字清晰易读 48

图标和图形 51

应用图标 51

条栏图标 51

图像 52

术语和措辞 53

与 iOS 整合 55

正确使用标准 UI 元素 55

响应设备方向变化 56

弱化文件和文档处理 57

必要时提供设置方式 58

充分利用 iOS 技术 59

设计策略 60

设计原则 61

美学完整性 61

一致性 61

直接操控 62

反馈 63

隐喻 64

用户控制 64

从概念到产品 65

定义你的 App 65

1. 列出所有你认为用户会喜欢的功能点 65

2. 定义你的目标用户 65

3. 通过定义目标用户来筛选功能点 66

4. 继续向前 66

为任务量身定制 67

原型&迭代 68

案例研究：从桌面到 iOS 69

iPad 中的 Keynote 69

iPhone 中的「邮件」 71

iOS 中的网页内容 72

在 iPhone 5 中运行	74
iOS 技术	81
Passbook	82
多任务处理	84
路线导航	86
社交媒体	88
iCloud	90
App 内购买	92
Game Center	94
通知中心	96
iAd 富媒体广告	99
AirPrint	102
位置服务	104
快速预览	106
声音	107
理解用户期望	107
定义应用的声音行为	108
管理音频中断	111
合适地处理远程多媒体控制事件	113
VoiceOver	114
编辑菜单	115
撤销和重做	117

键盘和输入视图 118

用户界面元素 119

条栏 120

状态栏 120

导航栏 121

工具栏 123

工具栏和导航栏图标 124

标签栏 126

标签栏图标 127

搜索栏 129

范围栏 130

内容视图 131

活动 131

活动视图控制器 132

精选视图 133

容器视图控制器 134

图像视图 135

地图视图 136

页面视图控制器 137

弹出窗口（仅 iPad） 138

滚动视图 141

分栏视图控制器（仅 iPad） 142

表格视图 144

文本视图 149

Web 视图 150

控件 152

活动视图指示器 152

「添加联系人」按钮 153

日期选择器 153

详情展开按钮 154

信息按钮 155

标签 155

网络活动指示器 156

页码控件 156

选择器 157
进度视图 158
刷新控件 158
圆角矩形按钮 159
分段控件 159
滑块 160
步进器 161
开关 161
系统按钮 162
文本框 163

临时视图 165

警告框 165
操作菜单 168
模态视图 170

图标和图像设计 171

图标和图像尺寸 172

应用图标 174
文档图标 177
Spotlight 和「设置」图标 177

启动画面 179

条栏按钮图标 181

报刊杂志图标 184

Web Clip 图标 186

创建可伸缩图像 187

文档修订历史 188

表格

声音 107

表 30-1 音频会话类别及其相关行为 109

条栏 120

表 35-1 用于工具栏和导航栏的标准按钮 125

表 35-2 用于标签栏的标准标签图标 128

图标和图像设计 171

表 39-1 自定义图标和图像的尺寸（像素单位） 173

报刊杂志图标 184

表 43-1 每期报刊杂志图标的最大缩放尺寸 185

用户界面设计基础

- 「为 iOS 7 而设计」 (第 8 页)
- 「iOS 应用解析」 (第 21 页)
- 「启动和停止」 (第 23 页)
- 「布局」 (第 27 页)
- 「导航」 (第 30 页)
- 「模态情境」 (第 33 页)
- 「交互和反馈」 (第 35 页)
- 「动画」 (第 43 页)
- 「品牌化」 (第 45 页)
- 「色彩和字体」 (第 47 页)
- 「图标和图形」 (第 51 页)
- 「术语和措辞」 (第 53 页)
- 「与 iOS 整合」 (第 55 页)

为 iOS 7 而设计

iOS 7 具体体现了以下主旨：

- **依从 (Deference)**。用户界面 (UI) 应当有助于用户理解内容并与之互动，而非对抗。
- **清晰 (Clarity)**。文字在每种字号下都易于阅读，图标表意准确清晰，装饰也恰到其度，并以对功能的无比关注驱动设计。
- **纵深 (Depth)**。视觉上的分层界面和逼真的动作使其更赋活力，提升了用户的愉悦和理解。

无论你是重新设计一个现有的 app，还是创造一个全新 app，都可以考虑借鉴 Apple 重新设计内置 app 的方法：

iOS 7 中的「天气」



iOS 6 中的「天气」



- 首先，剥离界面，将核心功能展露，并重新审视应用的实质。
- 接着，基于 iOS 7 的主旨（依从、清晰和纵深）去提升用户界面设计和用户体验。谨慎且有根据地构建每一个细节和装饰。

- 在这个过程中，随时准备推翻成见，保持怀疑，并以内容与功能为中心来驱动每一个设计上的决定。

依从内容

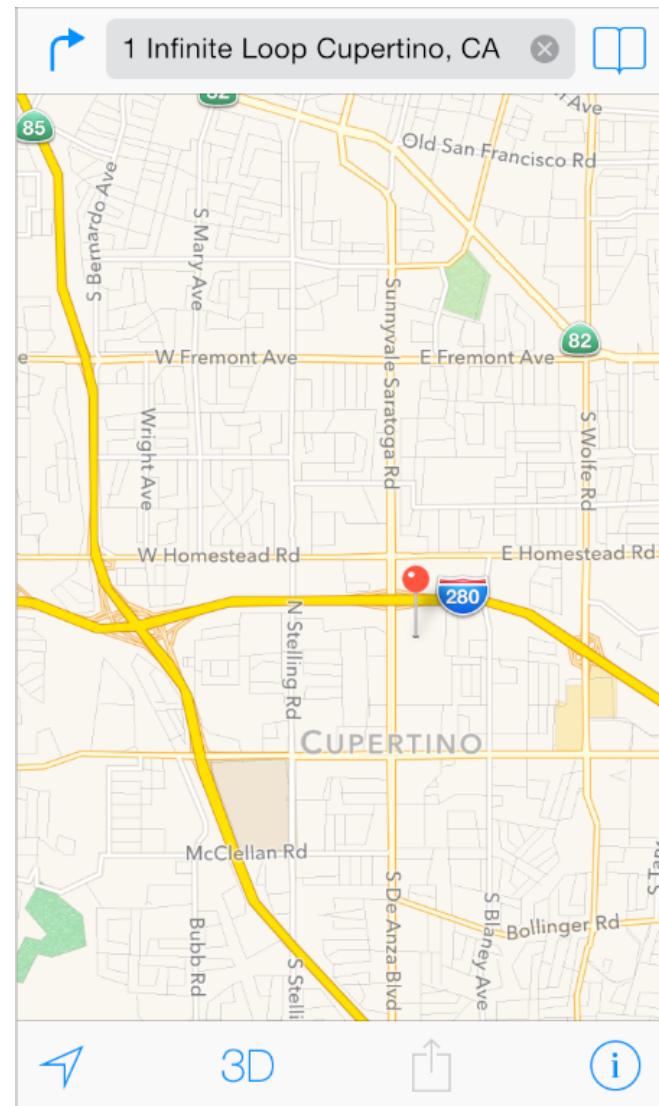
虽然通透、美观的界面和流畅的动画效果是 iOS 7 体验中最为瞩目的部分，但用户的内容才是真正的核心。

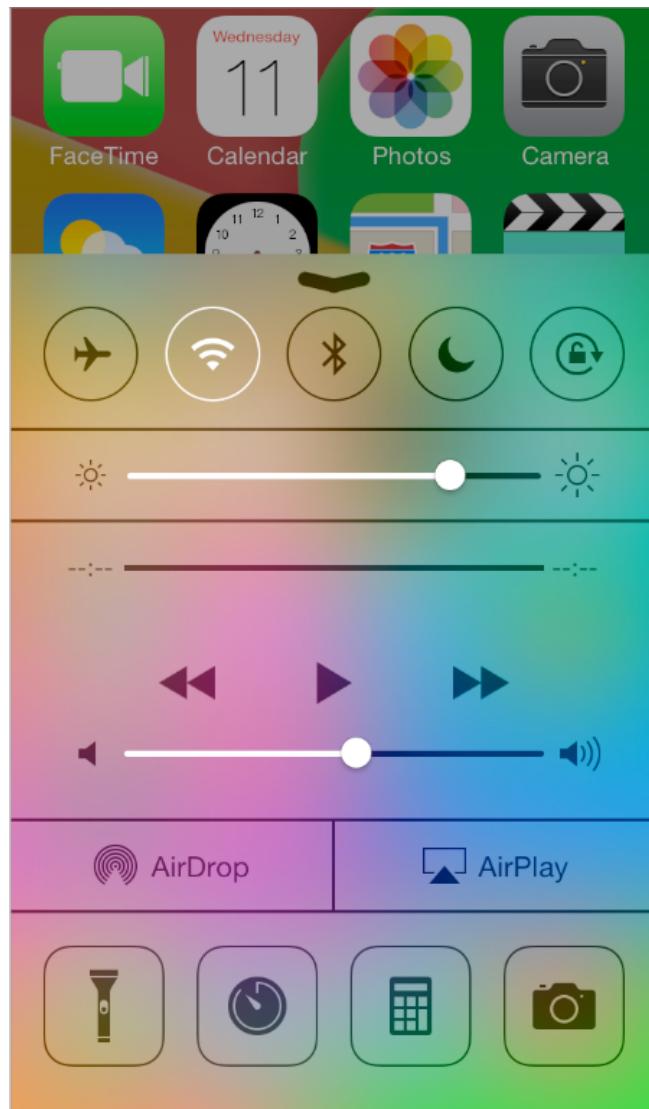
下面这些方法可以确保你的设计追随功能，并依从于用户内容。



充分利用整块屏幕。重新考量对插图和视觉框架的使用，可以考虑让内容扩展到屏幕边缘。「天气」就是一个很好的范例：漂亮的全屏界面非常直观地呈现出某个地点当前天气的关键信息，而且还有多余空间可以显示每个小时的天气数据。

重新考量模拟现实的视觉表现。浮雕、渐变和阴影效果有时会让界面元素变得沉重，进而喧宾夺主。相反，要突出内容并让界面扮演一个辅助性的角色。

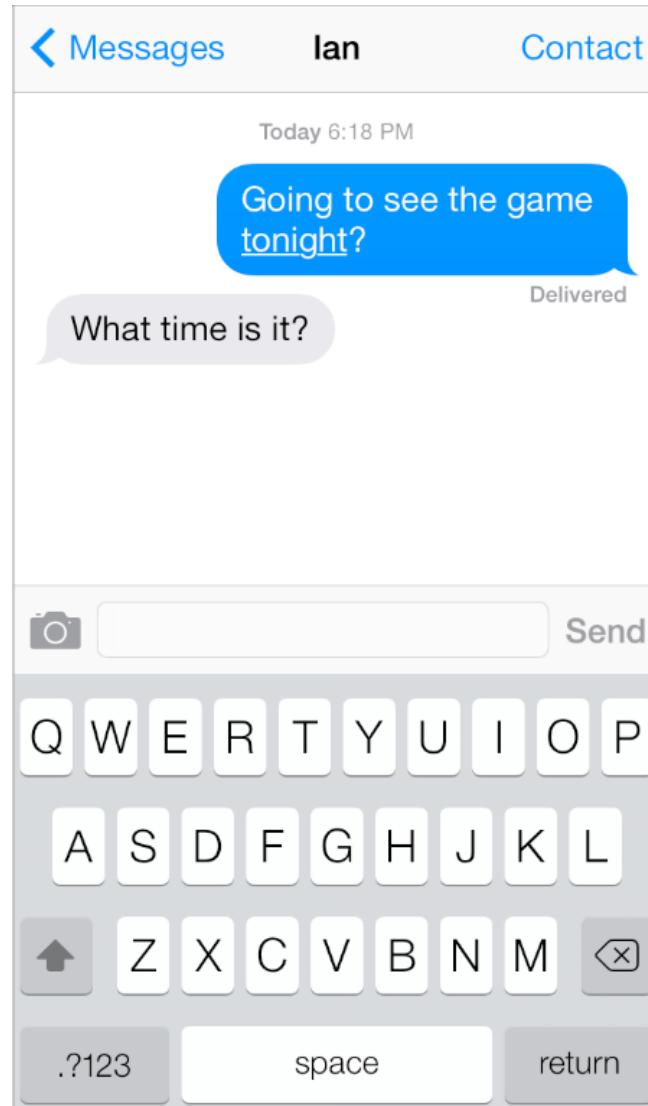




通过半透明的界面元素来暗示背后的内容。半透明效果（如「控制中心」）可以提供情境，帮助用户看到更多可用的内容，并给人一种短暂停留的暗示。在 iOS 7 中，半透明元素只模糊渲染在其正背后的内容，给人一种透过宣纸的感觉，但屏幕上的其他部分并不会模糊。

清晰呈现

清晰呈现是另一种在你的 app 中来确保依从内容的方法。下面这些方法可以使得最重要的内容和功能都能清晰呈现并便于操作。



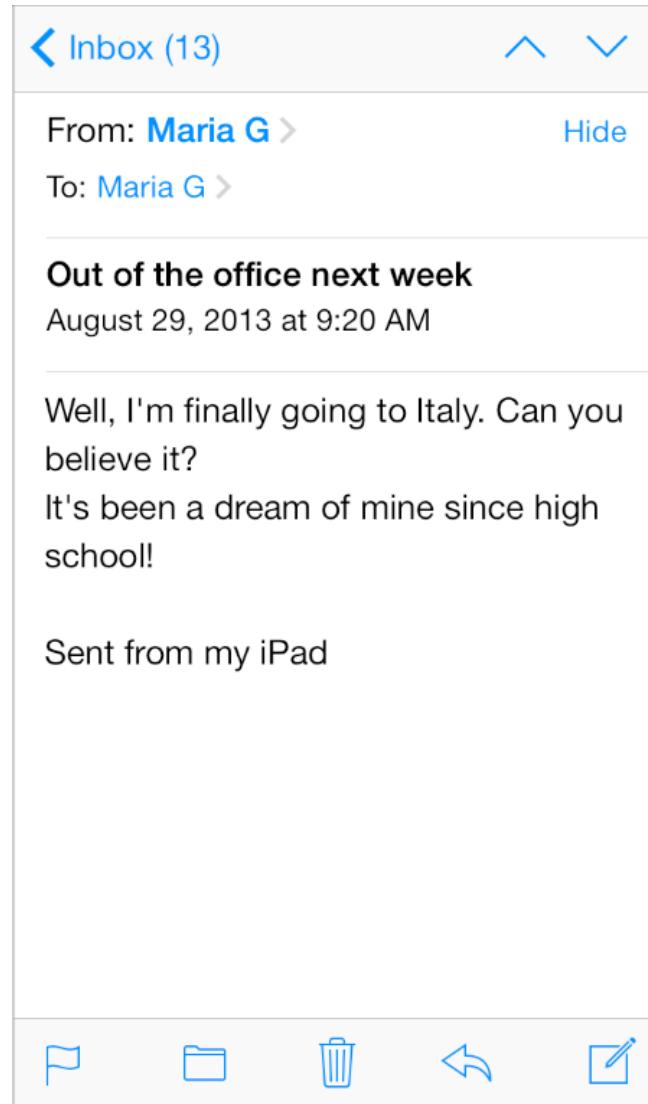
大量使用留白。留白会让重要内容和功能更为突出、更易于理解。同时，留白还可以传达一种安静平和的感觉，这会让 app 看上去更加专注和高效。

用色彩简化界面。使用一种主题色——例如「便签」中的黄色——来突出重点，并巧妙地暗示其交互性。同时，这会给 app 带来一致性的视觉主题。内置的 app 使用了一系列纯净的系统颜色，而每一种颜色在深色浅色两种背景中看起来都很好。

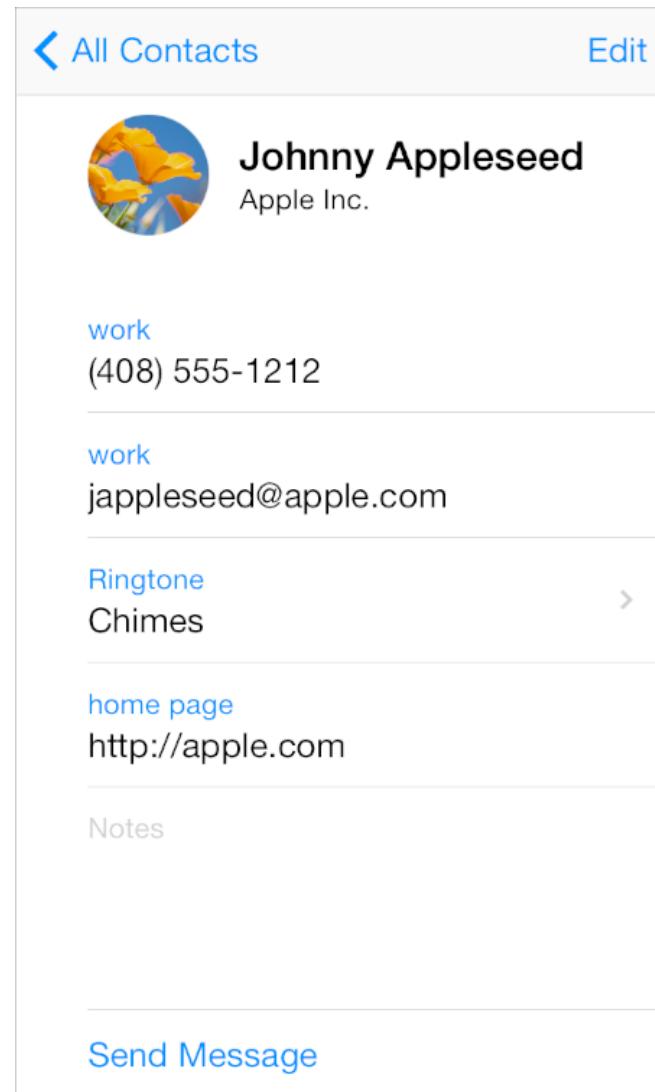
◀ Notes

To be, or not to be, that is the question:
Whether 'tis Nobler in the mind to suffer
The Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
And by opposing end them: to die, to sleep
No more; and by a sleep, to say we end
The Heart-ache, and the thousand Natural shocks
That Flesh is heir to? 'Tis a consummation
Devoutly to be wished. To die, to sleep,
To sleep, perchance to Dream; Aye, there's the rub,
For in that sleep of death what





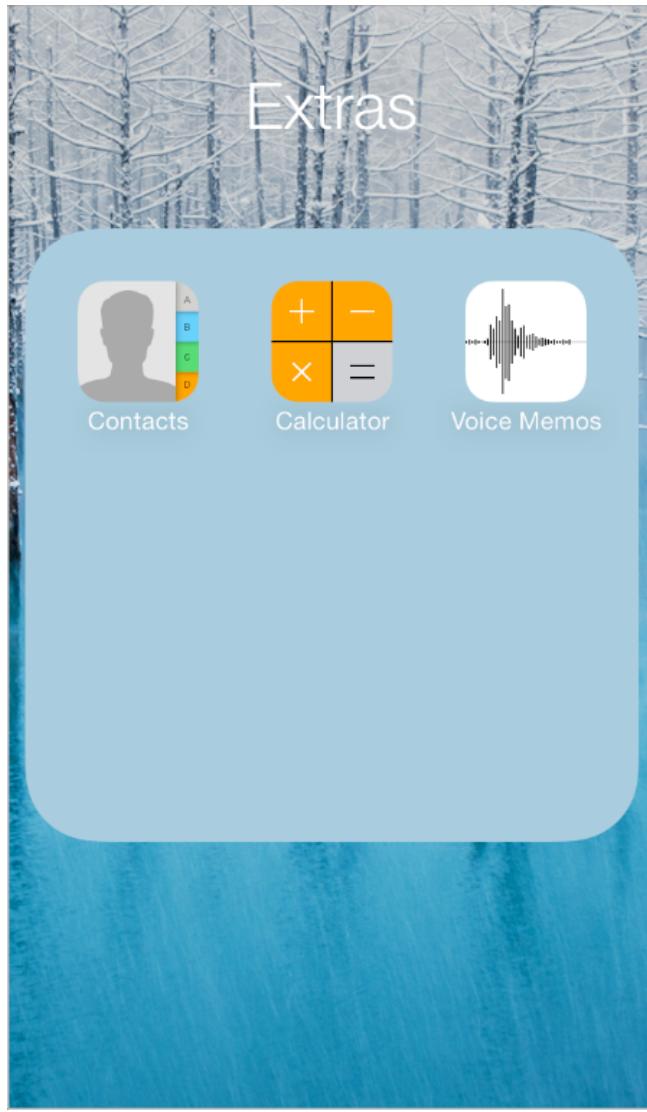
使用系统字体以确保易读性。 iOS 7 的系统字体能够自动调整字间距和行高，这会让文本内容易于阅读，且在任意字号下都显示良好。无论你是使用系统字体还是自定义字体，请确保使用「动态字体」(Dynamic Type)，这样在用户选择不同的字号时你的 app 可以作出响应。



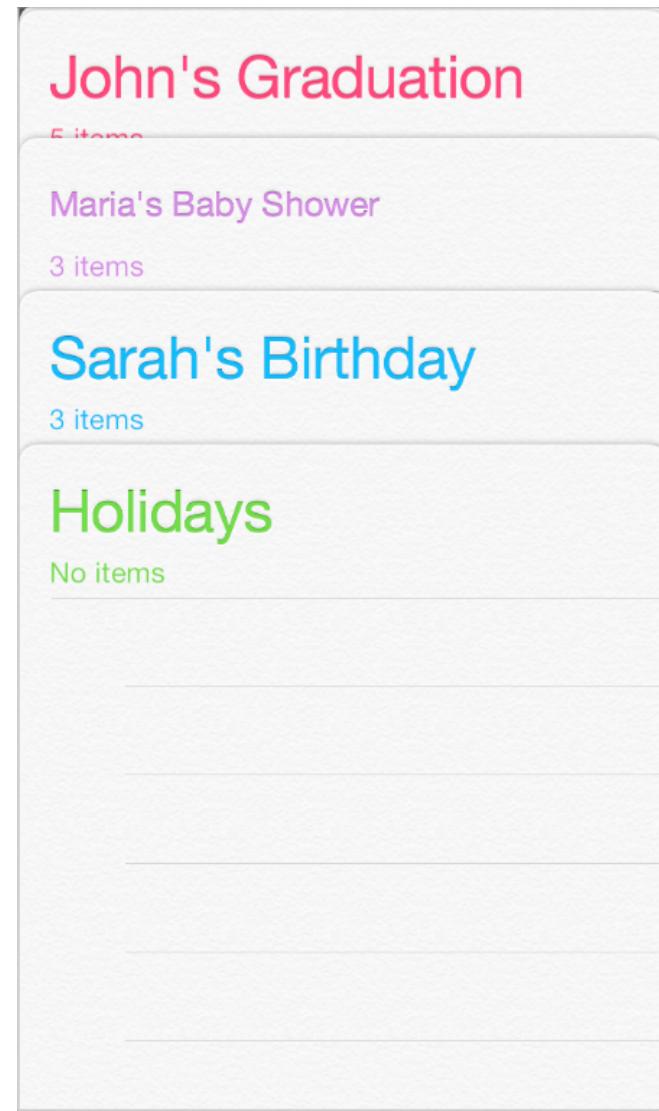
拥抱无边框按钮。在 iOS 7 中，所有的条栏按钮都没有边框。无边框按钮会在按钮的内容区域内使用情境、颜色和一个动作导向的标题来暗示其交互性。如果合适，还可以通过显示纤细的边框或者浅色背景来让按钮更加突出。

纵深传达

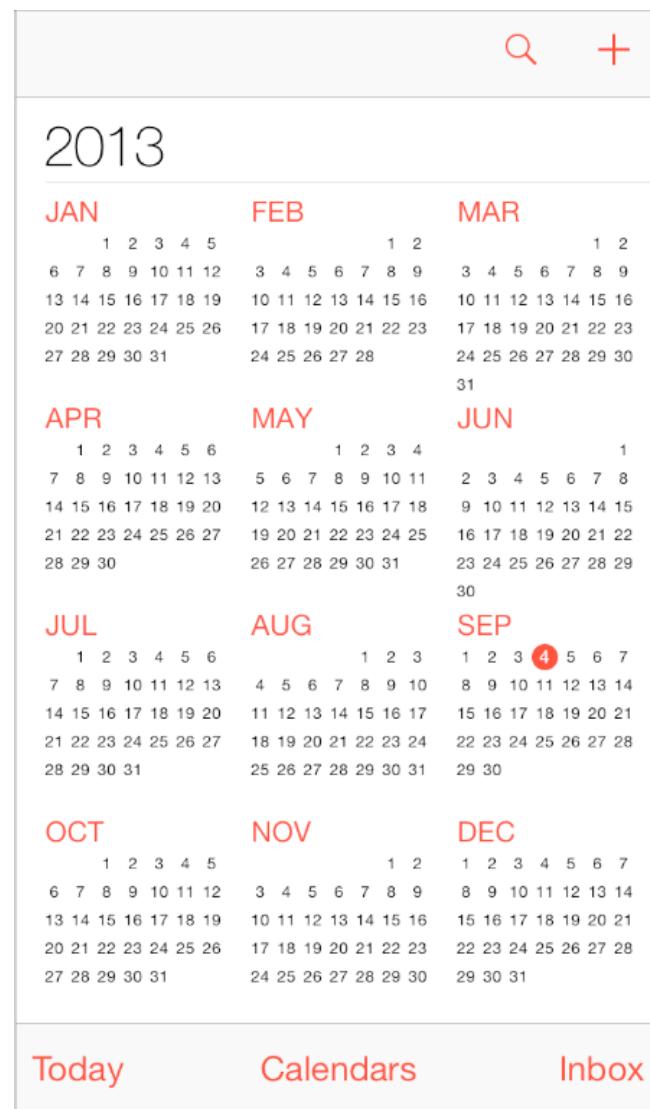
iOS 7 会在不同的分层界面 (layers) 上显示内容，这些包含层次和位置的分层界面能帮助用户理解界面对象之间的关系。



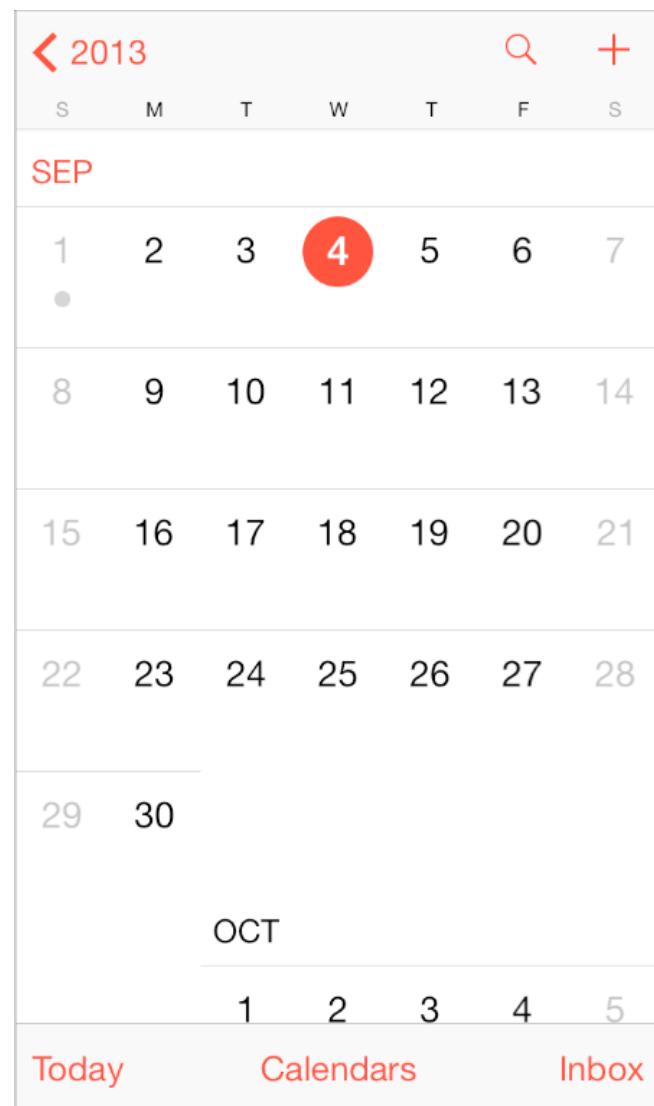
通过在主屏幕之上浮现一个半透明背景，来将文件夹中的内容和屏幕其他部分区分开来。



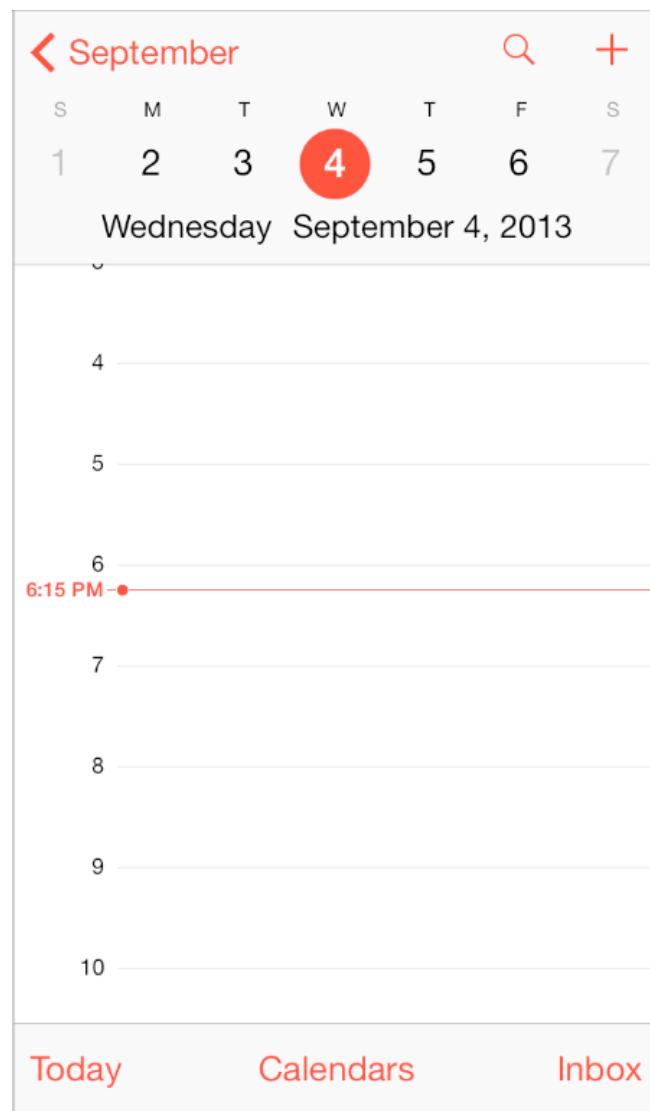
「提醒事项」在多个图层中显示列表，如图所示。当用户使用其中一个列表时，其他的列表会在屏幕底部被收起。



当用户在「日历」的年度、月份和日视图之间切换时，强烈的转场动画会给他们一种纵深和层次感。滚动如图所示的年视图，用户一眼就能看到当天的日期并执行其他日历任务。



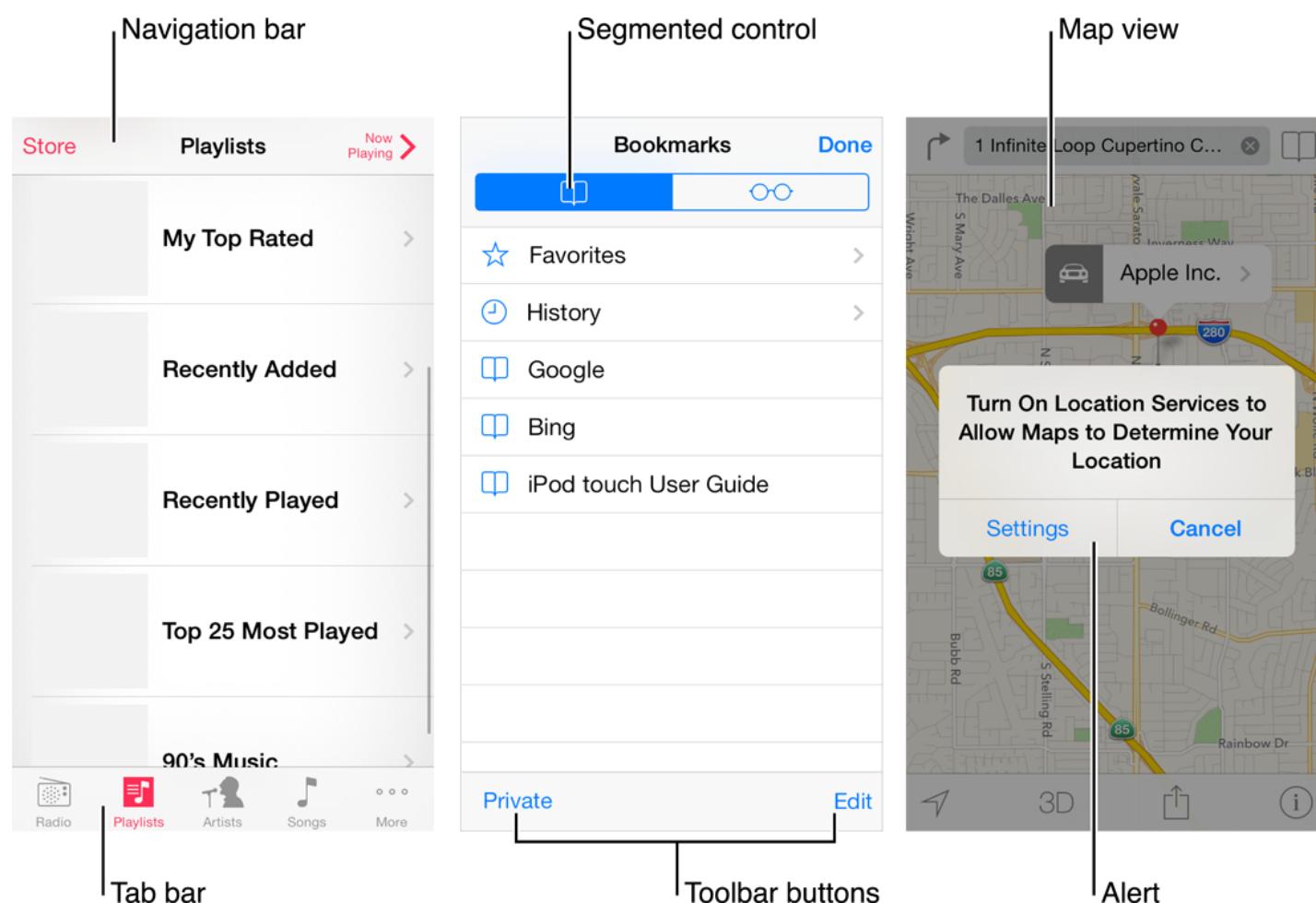
当用户选中某个月份，年度视图会以放大效果消失，并随之展现月份视图。当天的日期仍然保持红色高亮，而年份则出现在返回按钮中，这样用户便能准确了解自己现在所处的位置，是从哪里过来的，以及如何返回。



类似的转场动画还发生在用户选中某一天时：月份视图会向上下两侧分开，从而将本周推到屏幕顶部，随之显示被选中的某天的每小时视图。通过这些转场效果，「日历」强化了年度、月份和日视图之间的层级关系。

iOS 应用解析

几乎所有 iOS 应用都会使用一些由 UIKit 框架定义的 UI 组件。了解这些基本组件的名称、角色和作用，将会帮助你在设计 app 界面时做出正确合理的决定。



UIKit 提供的界面元素大致可以分为四类：

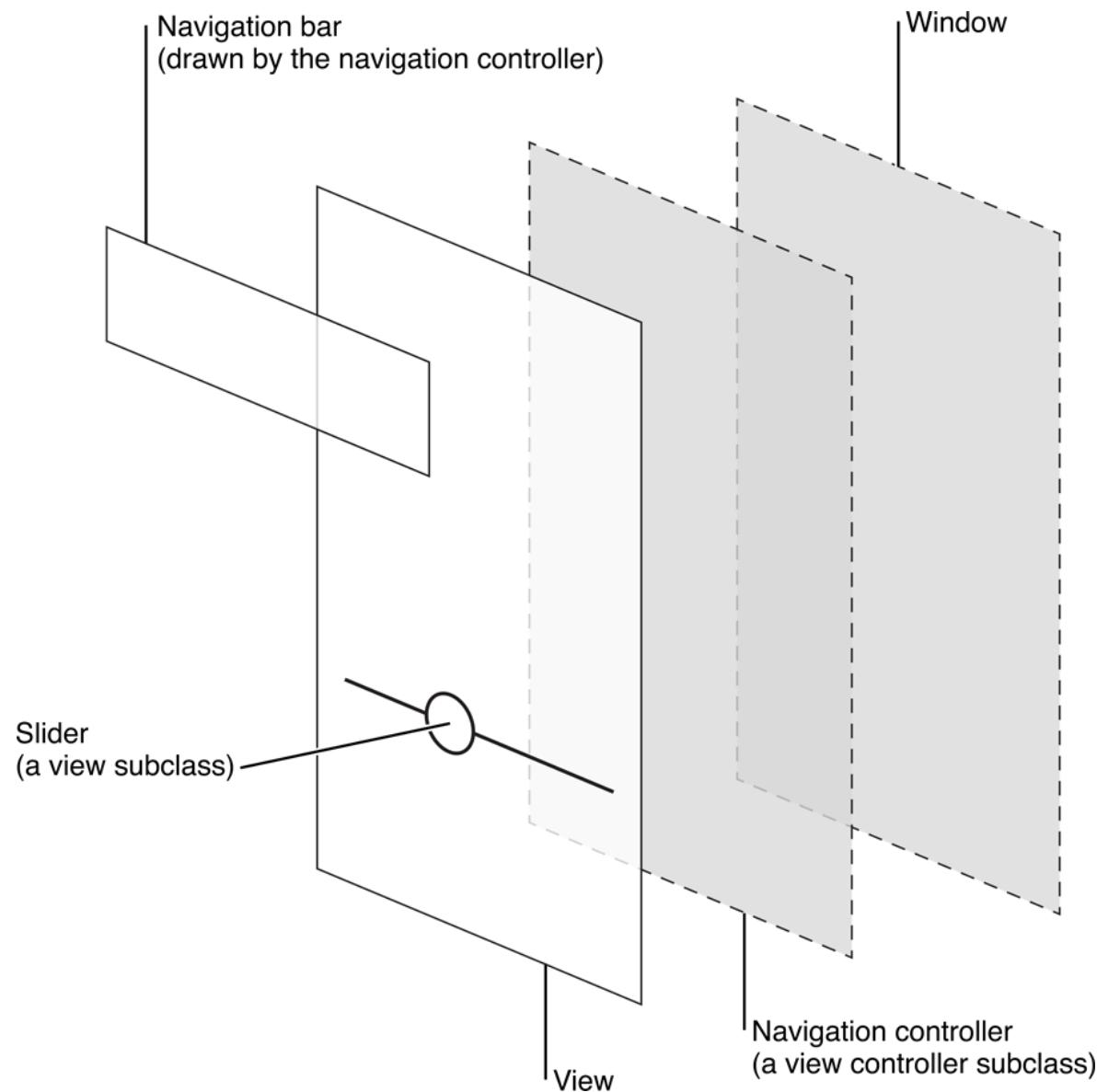
- **条栏。** 条栏包含告知用户其所在位置的情境信息，以及帮助用户浏览或执行操作的控件。
- **内容视图。** 内容视图容纳 app 特有的内容，并支持诸如滚动，插入，删除和条目重排等行为。
- **控件。** 控件可以触发操作或展示信息。
- **临时视图。** 临时视图会简要地给用户呈现重要信息，并附带操作和功能。

除了定义界面元素，UIKit 还定义了一些可执行功能的对象，例如手势识别、绘制、辅助功能和打印支持。

从编程角度来说，由于界面元素都继承自 `UI View`，因此它们都可以被看成不同类型的 **视图**。视图知道如何绘制其自身的界面，以及感知用户在其范围内的触控行为。所有的视图类型包括控件（例如按钮和滑块），内容视图（例如精选视图和表格视图）和临时视图（例如警告框和操作列表）。

若要在你的 app 中管理一组或一系列的视图，你通常要用到视图控制器。视图控制器可以协调视图显示，实现用户交互的功能，还能管理一个页面到另一页之间的转场效果。例如，「设置」使用了一个导航控制器来显示其一系列的层级视图。

下面的示例展示了视图和视图控制器是如何协作呈现出一个 iOS app 的。



虽然开发者们通常会基于视图和视图控制器去思考设计一个 iOS 应用，但用户更倾向于这是在体验一个又一个的页面。从这个角度来说，每一个页面都对应着 app 中一个明确的视觉状态或模式。

注意：每个 iOS 应用都包含窗口——但不同于电脑程序的窗口——iOS 中的窗口没有任何可见部分，而且不能被移动到屏幕上的其他位置。大多数 iOS app 都只包含一个窗口，但那些支持外接显示器的 app 可以拥有更多窗口。

在《iOS 人机界面准则》中，我们会使用页面（Screen）这个能被大多数用户理解的词。作为开发者，你也将会在不同的上下文中读到页面这个词，但这时指的是你可以用来访问外部显示设备的 `UIScreen` 对象。

启动和停止

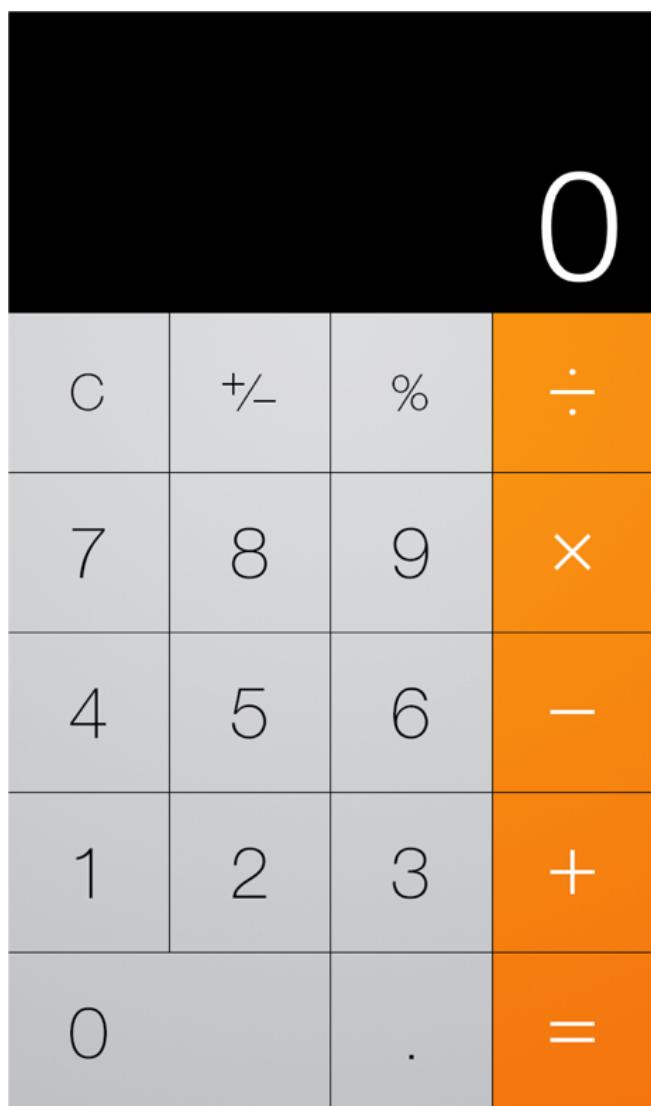
立即启动

一般来说，在接触新 app 的一两分钟内人们就会决定是否继续使用。如果你把握住这转瞬即逝的机会迅速呈现有用的内容，就可以激发新用户的兴趣，并给所有用户带来出色的体验。

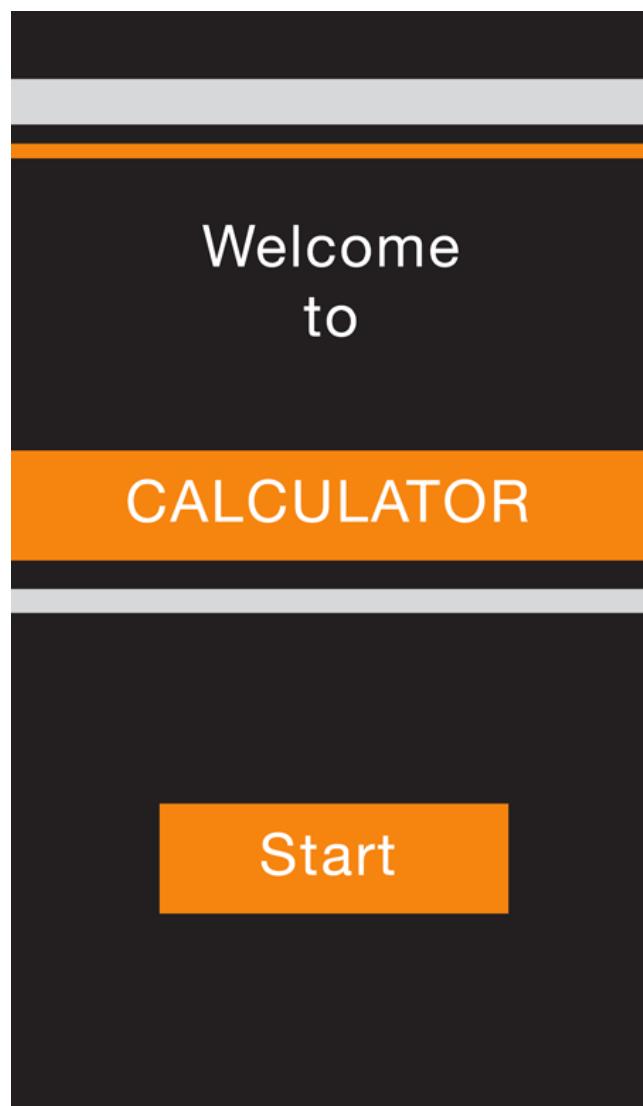
重要：不要在人们安装好你的 app 后告诉他们需要重启设备。重启会耽误时间，还会让你的 app 看起来不太可靠且难以上手。如果你的 app 由于内存使用或其他问题导致它只能在系统刚启动时才能很好运行，那你需要处理这些问题。有关开发高度优化的 app 的指南，请参阅《iOS App Programming Guide》中「Using Memory Efficiently」一节。

尽可能避免使用启动画面或其他类似的启动体验。最好让用户可以立即开始使用你的 app。

推荐



不推荐



避免要求用户提供设置信息，代之以：

- **关注 80% 用户的需求。**这样，大部分用户不需要进行任何设置，因为你的 app 已经按他们期望的方式设置好了。如果有一些功能只有极少数用户可能需要或者大部分用户只会用一次，请抛弃这样的功能。
- **尽可能通过其他方式获取信息。**如果你可以使用任何内置程序或设备中用户所提供的信息，那就从系统中获取这些信息，而不要让用户再输一次。
- **如果你确实需要提供设置信息，请让人们在你的 app 中输入。**然后，尽快存储这些信息（例如，在你 app 的设置中）。这样一来，在有可能体验到你 app 的乐趣前人们不会被迫切换到 iOS 的「设置」中了。如果人们将来需要修改这些信息，可以在任何时间前往 app 的设置页面进行修改。

尽可能将登录延后。最好在用户没有登录时就可以通过导航来浏览你的 app 并使用部分功能。用户在在可以做点有用的事情前通常已经从那些强制他们登录的 app 中离开了。

一般来说，以设备的默认方向启动。在 iPhone 上，其默认方向是竖屏；在 iPad 上，则是设备当前的方向。如果你的 app 只支持横屏，你应该始终以横屏方式启动，然后让用户在需要时旋转设备。

注意：对于只支持横屏的 app，最好能同时支持横屏竖屏两种状态，即 Home 键会在左侧或右侧的状态。如果设备已经处于横屏，除非有很好的理由支持，否则只支持横屏的 app 应当就以此方向启动。另外，对只支持横屏的 app，Home 键应该在其右侧。（如需了解更多关于支持不同设备方向的信息，请参阅「[响应设备方向的改变](#)」（第 56 页））

显示一个和 app 首屏极其相似的启动画面。iOS 会在你的 app 启动的瞬间显示这个启动画面，这可以让用户感觉你的 app 非常快，并让你有足够的时间去加载内容。参阅「[启动画面](#)」（第 180 页）了解如何创建启动画面。

如果可能，避免在用户首次加载你的 app 时让他们阅读免责声明或确认终端用户许可协议。你可以让 App Store 显示你的免责声明或者终端用户许可协议（EULA），以便人们可以在下载 app 前读到它们。如果你确实需要在 app 内展示这些东西，请确保以一种和 UI 保持和谐的方式将其整合进去，并在商业需求和用户体验间把握平衡。

App 重启后要恢复其状态，以便用户可以从中断的地方继续使用，而不要让用户去记住如何返回之前所在的位置。如需了解更多关于高效重建和恢复 app 状态的方式，请参阅「[State Preservation and Restoration](#)」。

随时准备停止

iOS 应用永远不会显示一个「关闭」或「退出」选项。人们退出一个 app 的方式就是切换到另一个 app、返回主屏幕或者让设备进入睡眠模式。

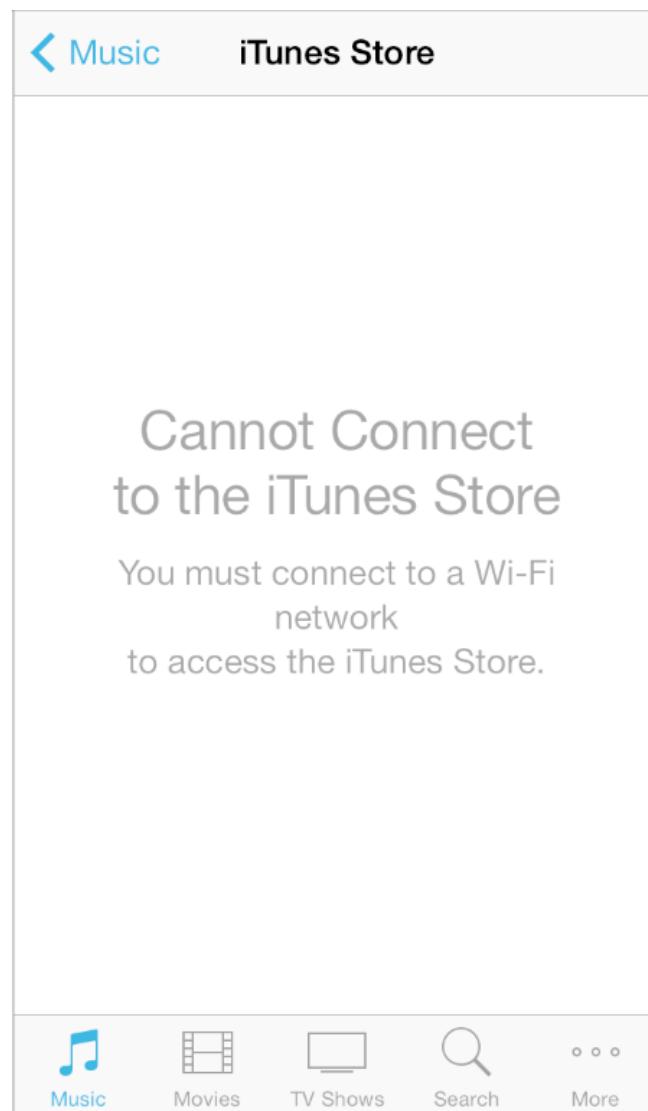
当人们从你的 app 切换离开，iOS 的多任务处理会将其挂到后台，并切换到新 app 的界面中。为了应对这种情况，你应当：

- **尽快并尽可能频繁地在合理范围内存储用户数据。**这样做是因为处于后台的 app 随时都有可能被退出或结束。

- **当 app 停止时，尽可能最多地保存当前状态的细节。**这样的话，当人们切换回来时就不会失去之前所处的情境。例如，如果你的 app 有显示可滚动的数据，那在停止时要保存当前所处的滚动位置。你可以前往「State Preservation and Restoration」了解更多关于高效重建和恢复 app 状态的方式。

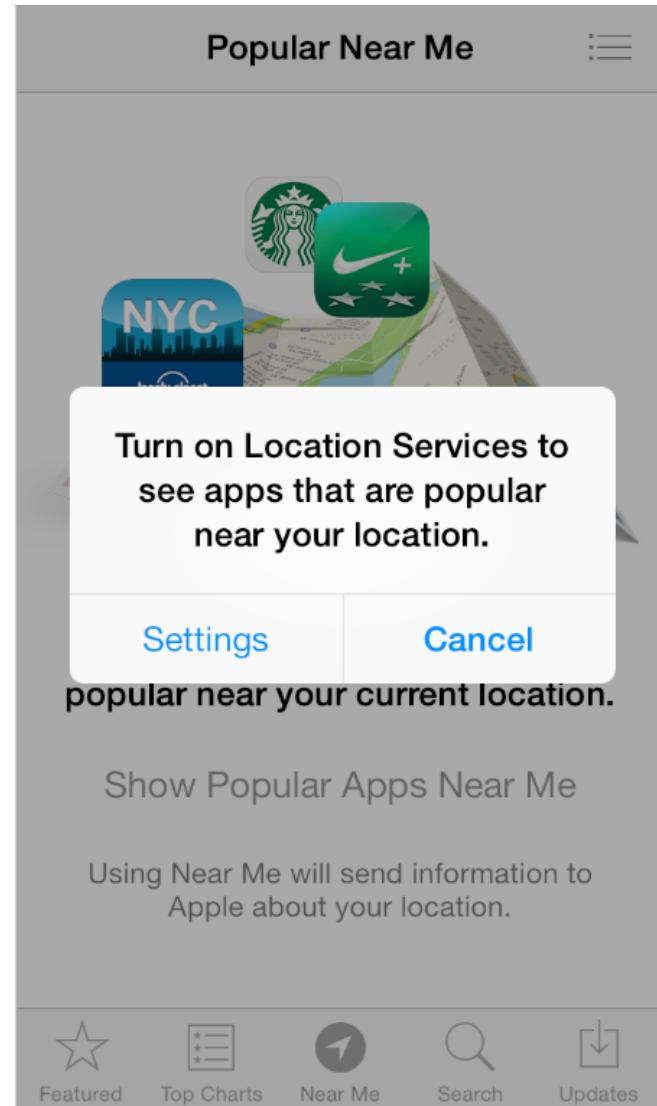
有些 app 可能需要在用户运行其他 app 时一直在后台运行。例如，当用户在另一款 app 中检查待办事项或者玩游戏时，他们会希望让正在播放音乐的 app 能继续播放。想要了解如何正确优雅地处理多任务，请参阅「[多任务处理](#)」（第 84 页）。

永远不要以程序化的方式自动退出一个 iOS app。用户有可能会将其认为是程序崩溃了。如果由于一些可预知的问题导致程序无法使用，那你需要告诉用户发生了什么、他们能做点什么。以下是两个不错的方法：



如果所有的功能都不可用，就向用户展示一个界面去描述问题并建议用户如何纠正。这能给用户一种反馈，让他们觉得你的 app 并没有出错。这也会赋予用户以控制感，让他们来决定是采取纠正措施并继续使用你的 app，还是切换到另一个 app。

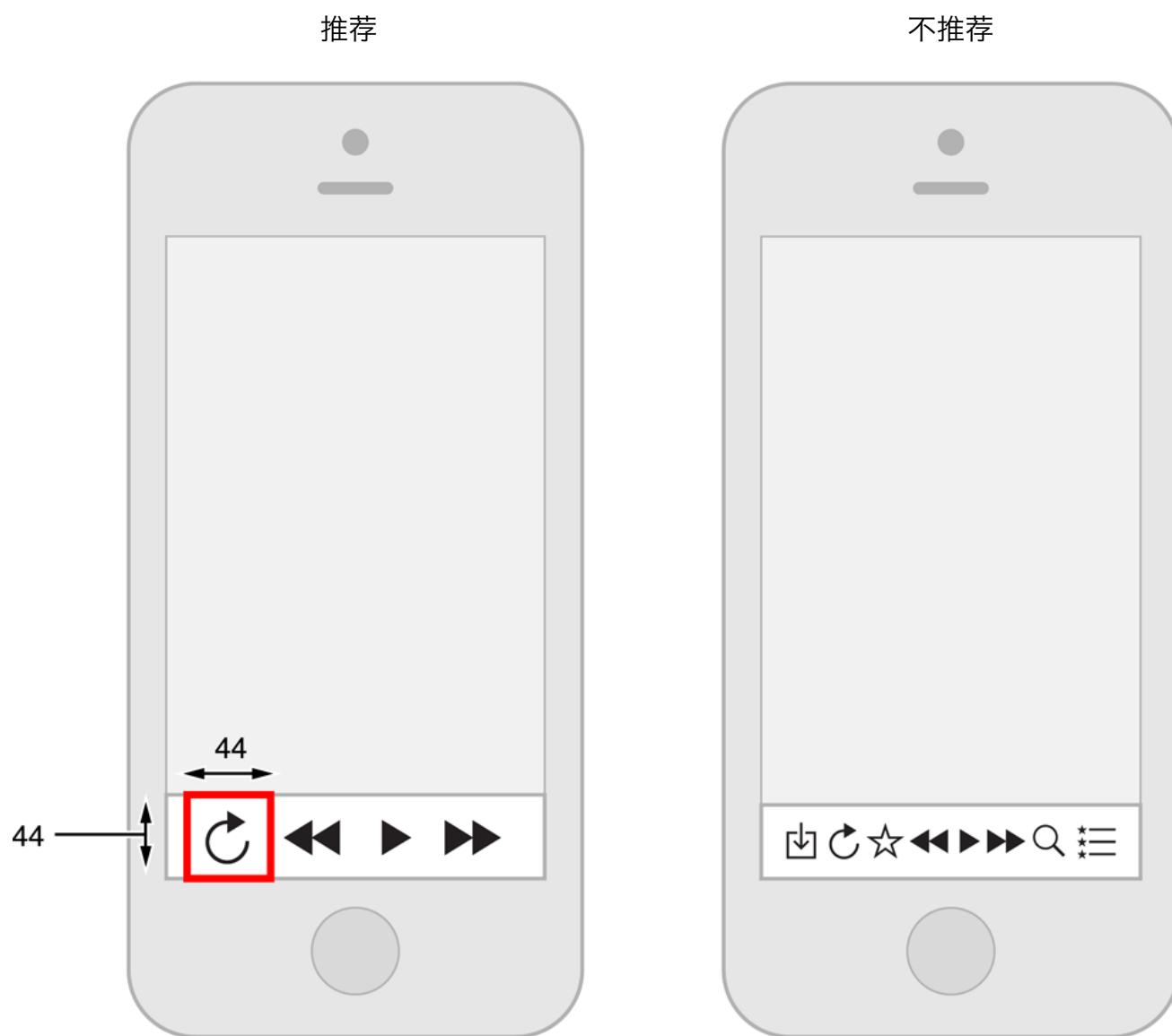
如果只有部分功能不可用，那在用户用到这些功能时提供解释界面或者提示。这样的话，人们还可以使用 app 的其他功能。如果你决定使用警告框，请确保只在用户尝试使用不可用的功能时展示。



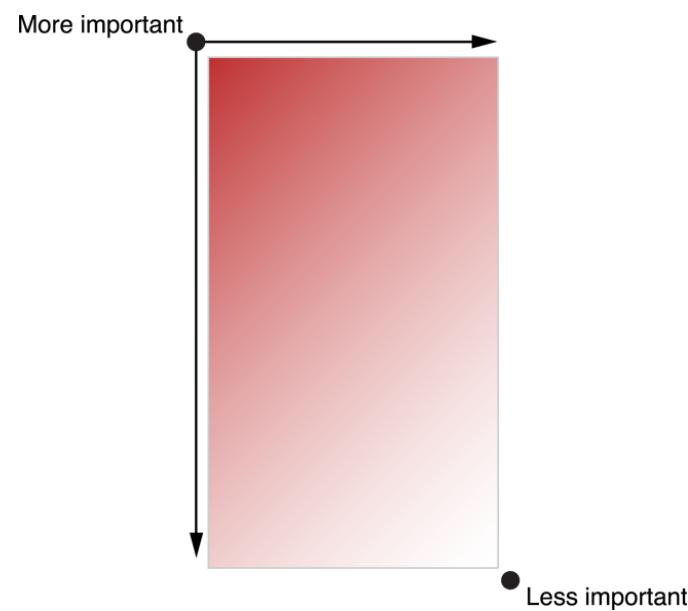
布局

布局不止关乎界面元素在 app 中如何呈现。通过你的布局能让用户知道哪些东西是最重要的、他们能做什么以及所有这些是如何联系起来的。基于你的程序所运行的设备及其所处的方向（横向或竖向），你的布局可能需要随之调整。

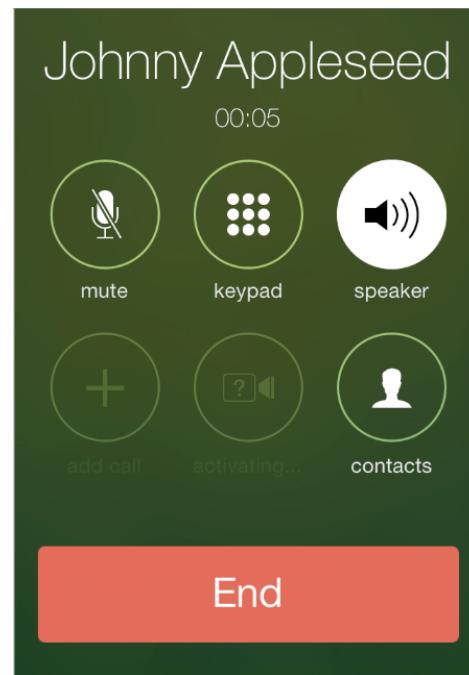
给每个交互元素以足够的间距，以便用户对内容和控件进行操作。可点击控件的点击区域不小于 44×44 点。



将重要内容和功能放到更重要的位置，以便用户可以更容易地关注主要任务。一些可行的方法是，将主要元素放置在屏幕的上半部分，并靠近屏幕左侧（基于从左至右的阅读习惯）：



利用视觉上的权重和平衡来向用户显示界面元素之间的相对重要程度。相比尺寸较小的元素，那些大尺寸的元素更吸引目光，看上去也更重要。同时，较大尺寸的元素也更容易被用户点击，这让它们在某些复杂环境里常用的 app（例如「电话」和「时钟」）中更为有用。



利用对齐来减少浏览和信息传达的分组和层级。对齐会让 app 看上去紧凑有序，还能让用户在滚动一整个屏幕的信息时保持目光焦点。不同信息分组之间的缩进和对齐，暗示着这些分组是如何关联，这让用户可以更容易地找到特定条目。

确保主要内容在默认尺寸下可以被阅读和理解。例如，用户不应需要滚动才读到重要文本，或者放大图片才能看到主要内容。

随时准备应对文字大小的变化。当用户在「设置」中更改文字大小时，他们会希望大多数的 app 可以作出合适的响应。为适应一些文字字号的变化，你可能需要调整布局；如需了解关于在 app 中显示文字的更多信息，请参阅「[文字清晰易读](#)」（第 49 页）。

尽可能避免在界面中出现不一致的样式。通常来说，拥有相似功能的元素看上去也应该很像。用户常常会认为那些他们所看到的不一致背后必定富有深意，然后花时间去理解它们。

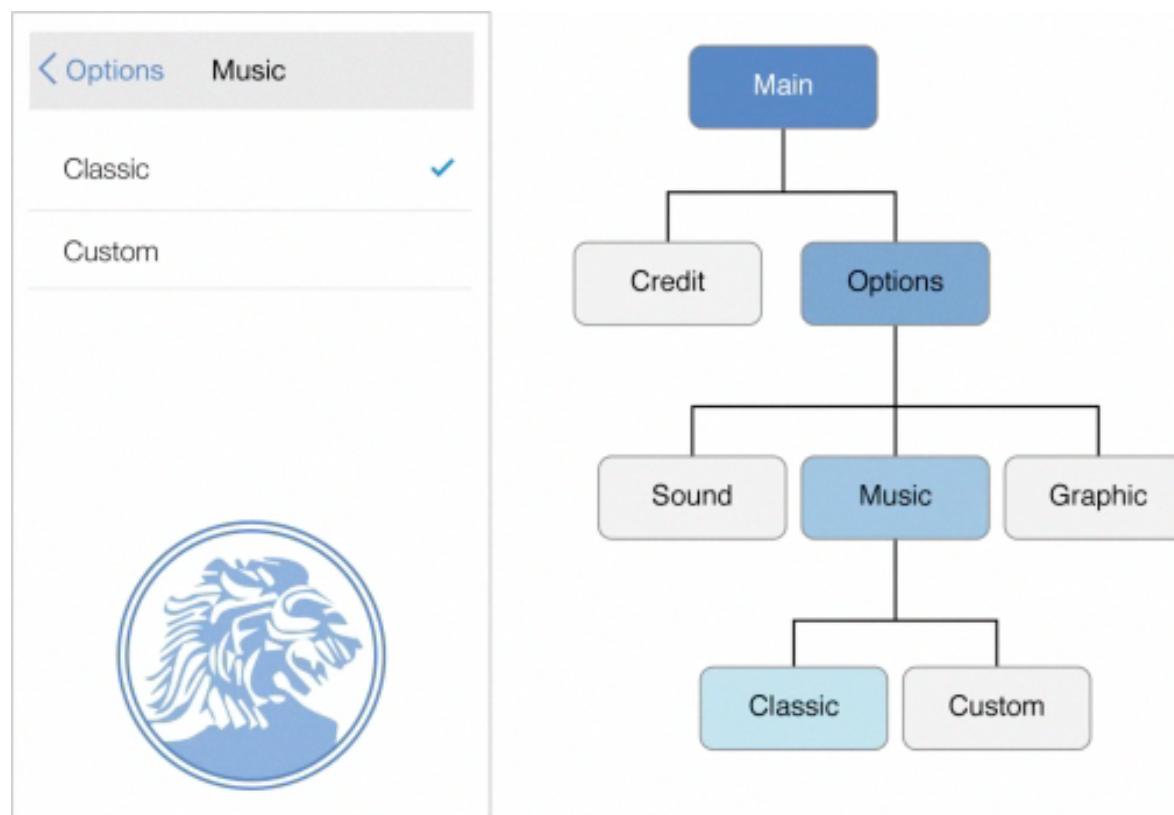
导航

除非与用户期望相悖，否则人们并不会注意到 app 的导航体验。你需要做的就是让导航与应用的结构、意图相契合，而不让用户注意到导航本身。

广义上来说，这里有三种主要的导航样式，分别对应三种不同的应用结构：

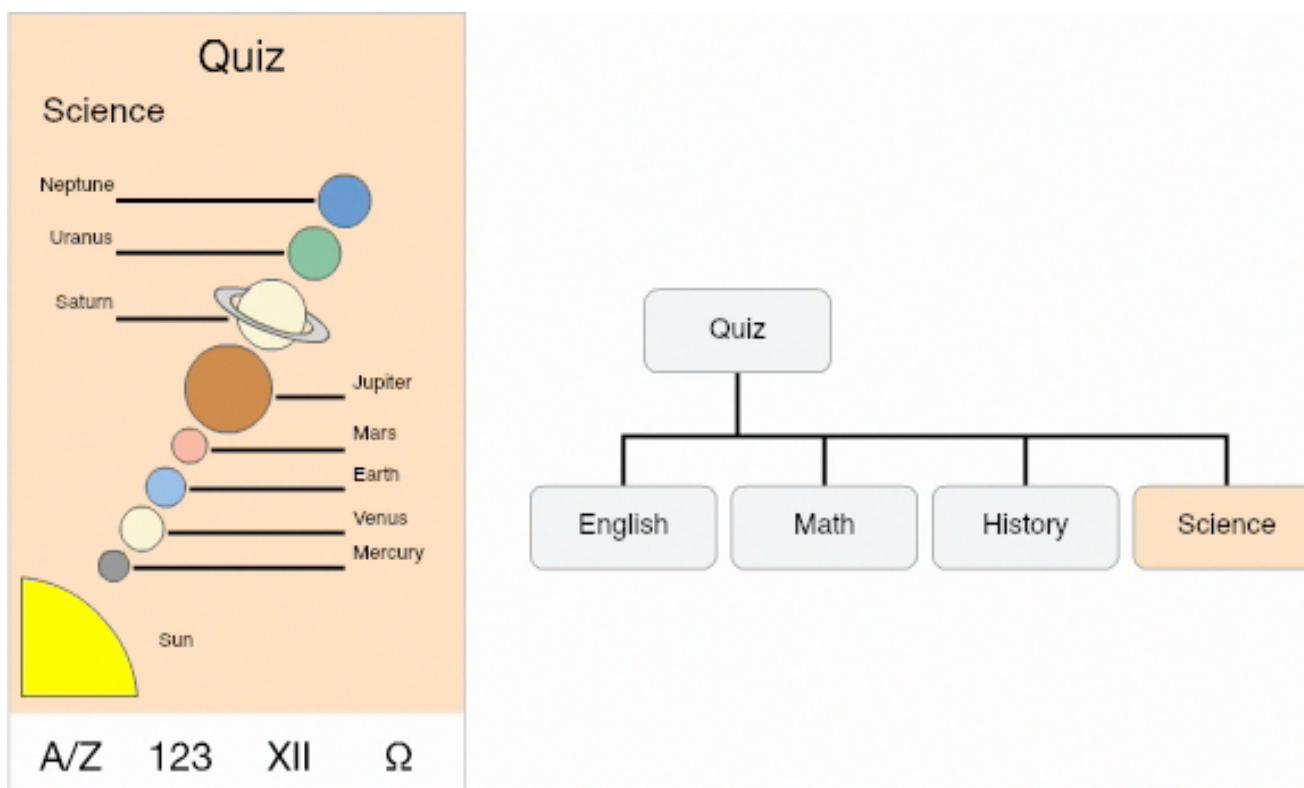
- 层级式
- 扁平式
- 内容/体验主导式

在层级式 app 中，用户通过在每个页面进行一次选择进行导航直至到达目标位置。要到达另一个位置，用户必须原路返回几步（或者从最开始重新出发）并作出不同的选择。「设置」和「邮件」是使用层级式结构 app 的绝佳示例。



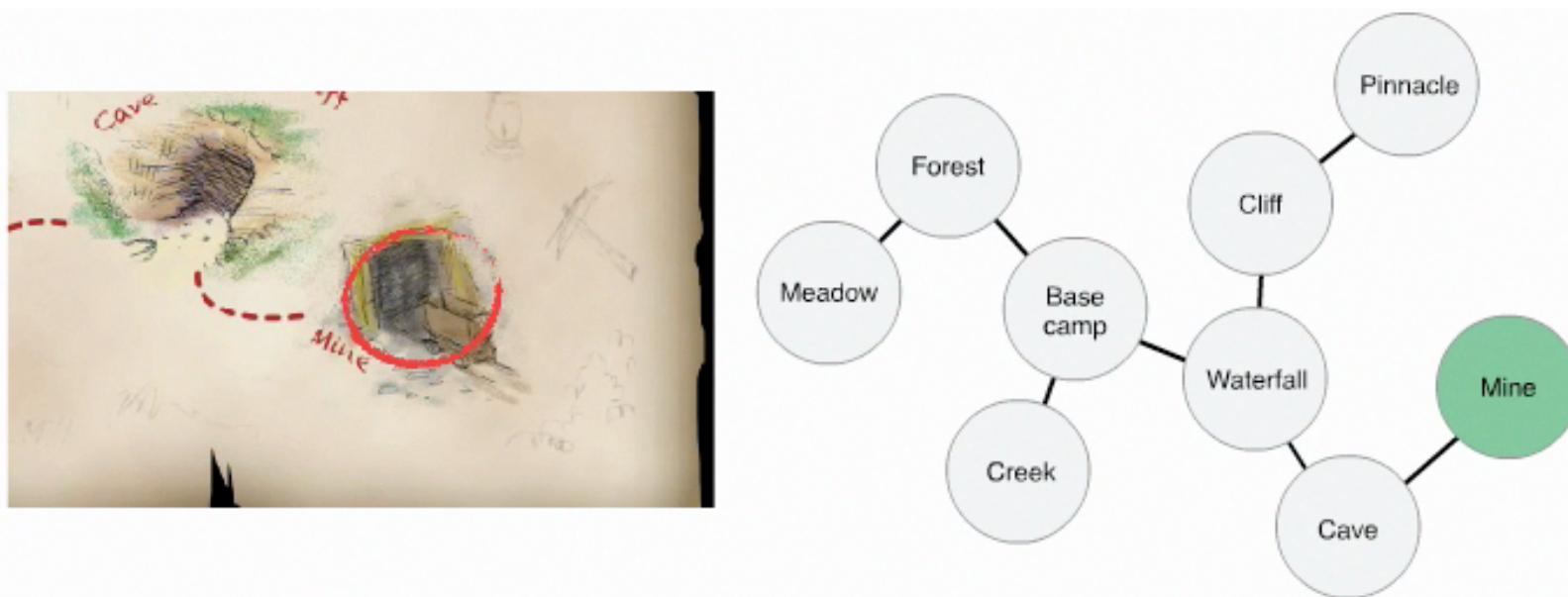
请查看此文档的 HTML 版本以观看视频。

在一个扁平式信息结构的 app 中，由于所有的主要类别都可以从主页面进入，用户可以直接从一个主要类别调到另一个类别。「音乐」和 App Store 便是使用扁平式结构的范例。



请查看此文档的 HTML 版本以观看视频。

在 app 中使用内容/体验主导式的信息结构并不奇怪，导航同样是由内容或体验定义的。例如，用户要浏览一本书，是通过从一个页面移动到下一页或是在内容目录中选择一页；在游戏中，导航尝尝是体验的一个重要部分。



请查看此文档的 HTML 版本以观看视频。

在某些情况下，组合使用多个不同的导航样式会很不错。例如在扁平信息结构中，同一类目下的项目最好能以层级形式来展示。

永远要让用户知道自己正处于应用中的什么位置，并清楚如何去往他的下一个目标。无论使用哪种符合应用结构的导航样式，最重要的是让用户的使用路径符合逻辑、可以预知且易于学习。

UIKit 中定义了一些用来实现层级式与扁平式导航的标准界面元素，同时也提供了一些元素来帮助你实现以内容为主导的导航样式，例如针对书本风格或媒体浏览风格的应用。而以体验为主导的导航样式——例如一款游戏应用——一般来说都是建立在自定义的元素与交互行为上的。

用导航栏来让用户在层级数据之间轻松穿梭。导航栏的标题能告诉用户目前在层级关系中所处的位置；后退按钮则可以轻松回到上一层次。如需了解更多信息，请参阅「[导航栏](#)」（第 121 页）。

用标签栏来展示平级分类的内容或功能。标题栏可以很好支持扁平结构的信息，因为它能随时随自如地实现在不同类目之间的切换。如需了解更多信息，请参阅「[标签栏](#)」（第 126 页）。

用页码控件来指示有多个子项目或多屏内容。页码控件可以很好地告知用户有多少个页面，以及当前正处于哪个页面。例如，「天气」使用了页码控件以显示用户已经打开了多少个特定位置的天气页。如需了解更多信息，请参阅「[页码控件](#)」（第 156 页）。

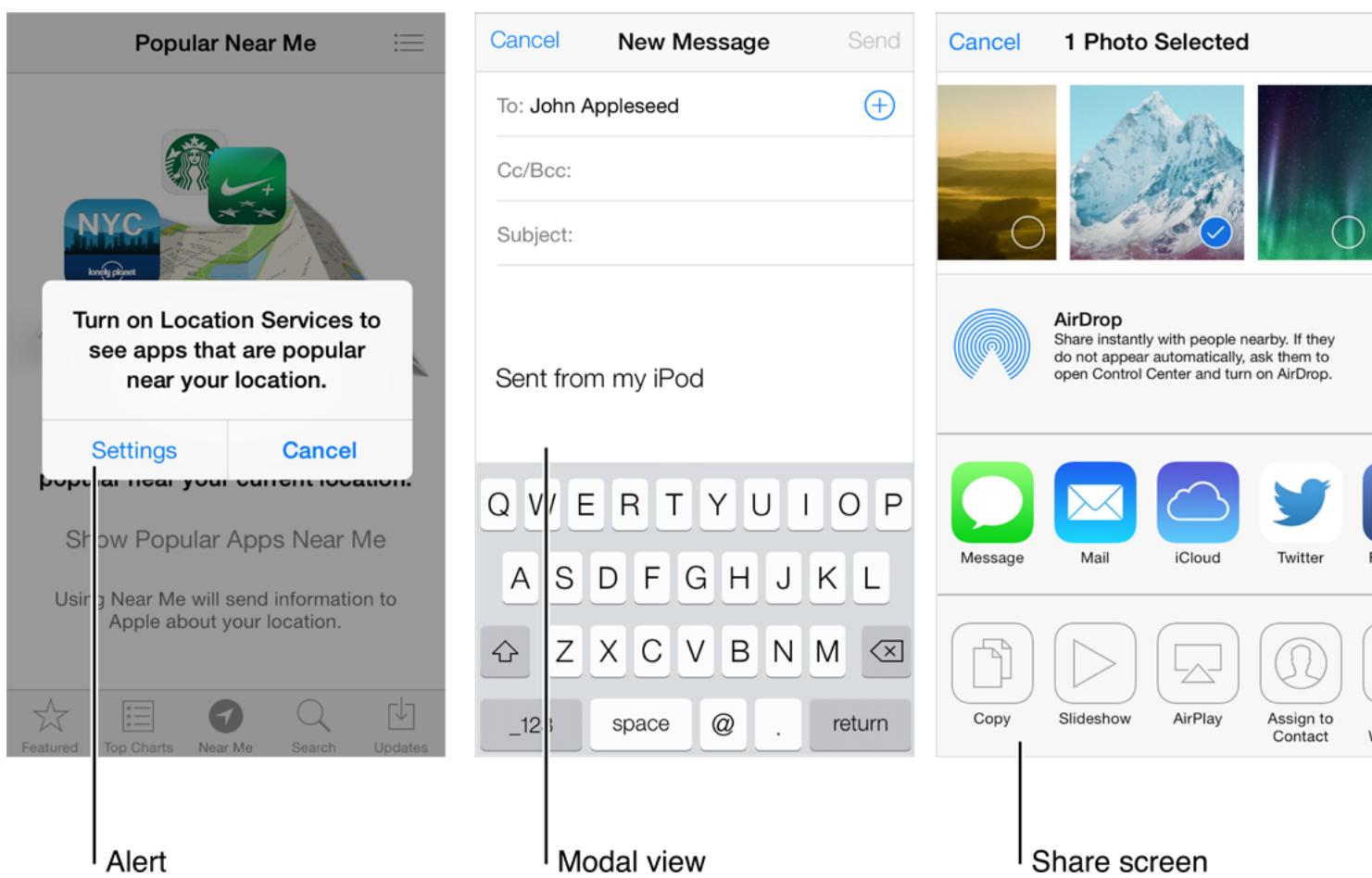
总的来说，最好是给用户以唯一的路径前往每个页面。如果用户需要多种情境中查看某个页面，请考虑使用临时视图（例如模态视图、操作列表或警告框）。如需了解更多信息，请参阅「[模态视图](#)」（第 170 页）、「[操作菜单](#)」（第 168 页）和「[警告框](#)」（第 165 页）。

UIKit 也提供以下的相关控件：

- 「[分段控件](#)」（第 159 页）。分段控件可以给用户一种办法，以在页面中查看内容的不同类别或方面，而无需通过导航到一个新页面去。
- 「[工具栏](#)」（第 123 页）。尽管工具栏看上去和导航栏或者标签栏很像，但它不是导航。反而，工具栏给用户一种可以在操作当前页面内容的控件。

模态情境

模态——一个承载内容或体验的模式——自有其优点和缺点。它可以帮用户完成某些任务或者不受干扰地获取信息，但也会暂时性地阻止用户与 app 的其他部分交互。



理想情况下，用户能够以非线性的方式和 iOS app 交互，所以最好能精简你 app 中模态体验。通常，只在以下这些情况下考虑创建模态情境：

- 非常需要吸引用户注意
- 必须完成自包含任务（或明确放弃），以避免让用户的 data 处于不明确状态

保持模态任务简短精炼。你不会希望用户将模态视图看成是你 app 中的一个小程序。如果子任务过于复杂，用户在进入模态情境时会忽略他们的主要任务。在创建一个涉及层级视图的模态任务时需要尤为谨慎，因为用户会感到迷茫以及忘记如何原路返回。如果模态任务必须在多个视图中包含子任务，确保在层级之间给用户一个唯一、清晰的路径，并避免产生循环。如需了解使用模态视图的准则，请参阅「[模态视图](#)」（第 170 页）。

始终提供一个明显而安全地退出模态任务的方式。当用户退出模态视图时，他们通常会知道其任务会被结束。

如果任务需要一系列多层次的模态视图，请确保用户在轻点次顶层视图中的「完成」按钮时知道会发生什么。检查任务流程，以决定次级视图中的「完成」按钮是仅仅完成视图中的部分任务，还是完成整个任务。鉴于存在混淆的可能性，请尽可能避免在附属视图中使用「完成」按钮。

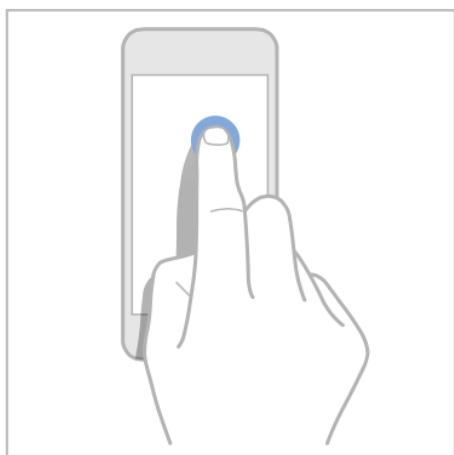
使用警告框传达必要——且可操作——的信息。警告框会中断用户的体验过程，并需要一次点击才能结束，因此让用户获知警告框出现的合理性是很重要的。如需了解更多信息，请参阅「[警告框](#)」（第 165 页）。

尊重用户关于接收通知信息的设定。在「设置」中，用户会设置希望以怎样的方式接收来自你的 app 的通知信息。请确保遵循这些设定，以免用户关闭来自你 app 的所有通知消息。

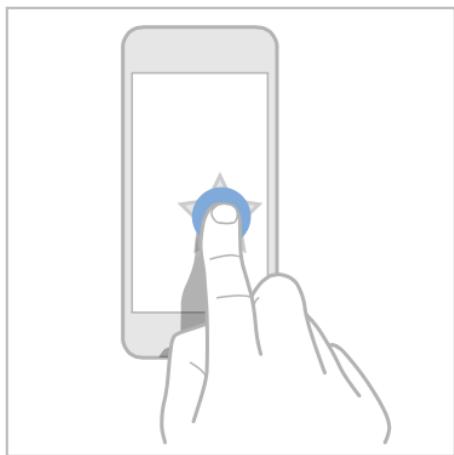
交互性和反馈

用户对标准手势了如指掌

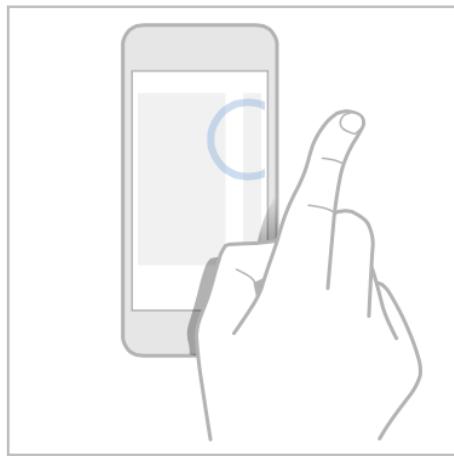
人们使用手势——例如轻点、拖拽和双指开合——来和 app 以及他们的 iOS 设备交互。使用手势能在人与设备之间建立起一种亲密的人性化联系，并增强用户对直接操控屏幕对象的感知。一般来说，人们会希望他们所使用的 app 中的手势操作保持一致。



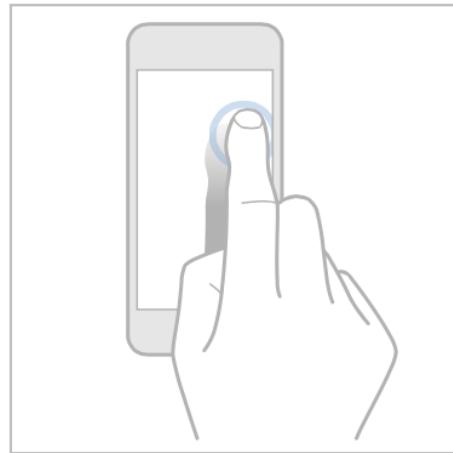
轻点 (Tap) 用来按下或选中一个控件或项目



拖拽 (Drag) 用来滚动或切换内容（即，从屏幕一边移动到另一边）
可以拖拽一个对象

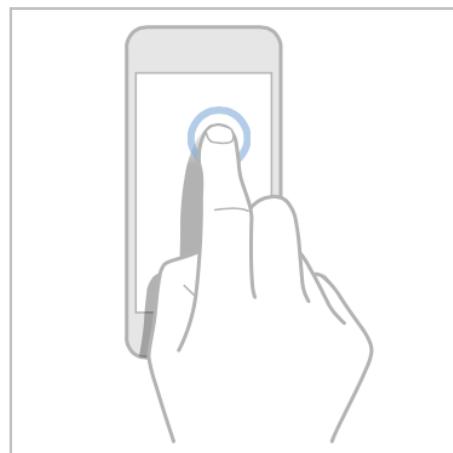


滑动 (Flick) 用来快速滚动或切换



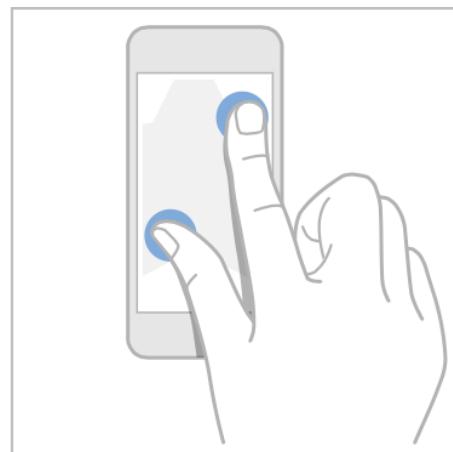
轻扫 (Swipe) 使用一只手指轻扫，可以返回上一个页面，可以显示在分栏视图中的隐藏视图（仅 iPad），或表格视图里某行的「删除」按钮。

在 iPad 上，四指轻扫可以在多个应用之间进行切换。

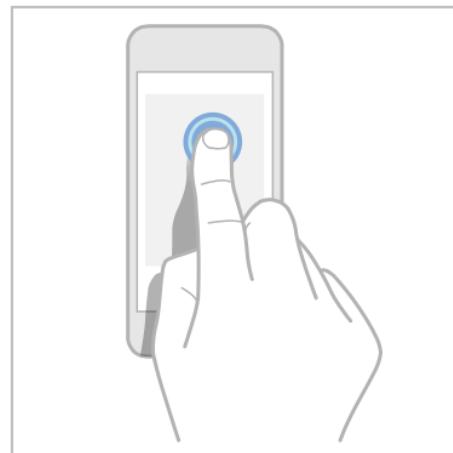


双击 (Double tap) 用来放大内容或图片，并将其置于屏幕中央。

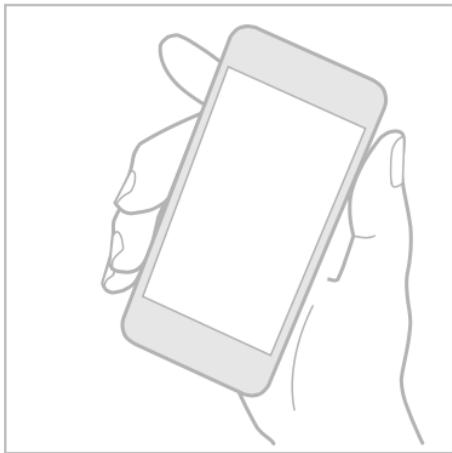
如果内容已被放大，则会将其缩小。



双指开合 (Pinch) 双指张开或闭合以放大或缩小内容



长按 (Touch and hold) 在可编辑或可选择的文本中会显示放大镜视图，用来定位光标。



摇晃 (Shake) 用来执行撤销或重做操作。

除了用户熟知的标准手势，iOS 还定义了一些在系统全局应用的操作手势，比如展开「控制中心」或「通知中心」。无论用户当前在使用什么应用，都可以通过这些手势进行操作。

避免为标准手势赋予不同的行为。除非你的 app 是游戏，否则重新定义标准手势的行为可能会让用户迷惑，并使其难以使用。

避免重复创建和标准手势具有相同行为的自定义手势。用户习惯于标准手势的行为，他们并不希望学习了不同的方式却做了同样的事情。

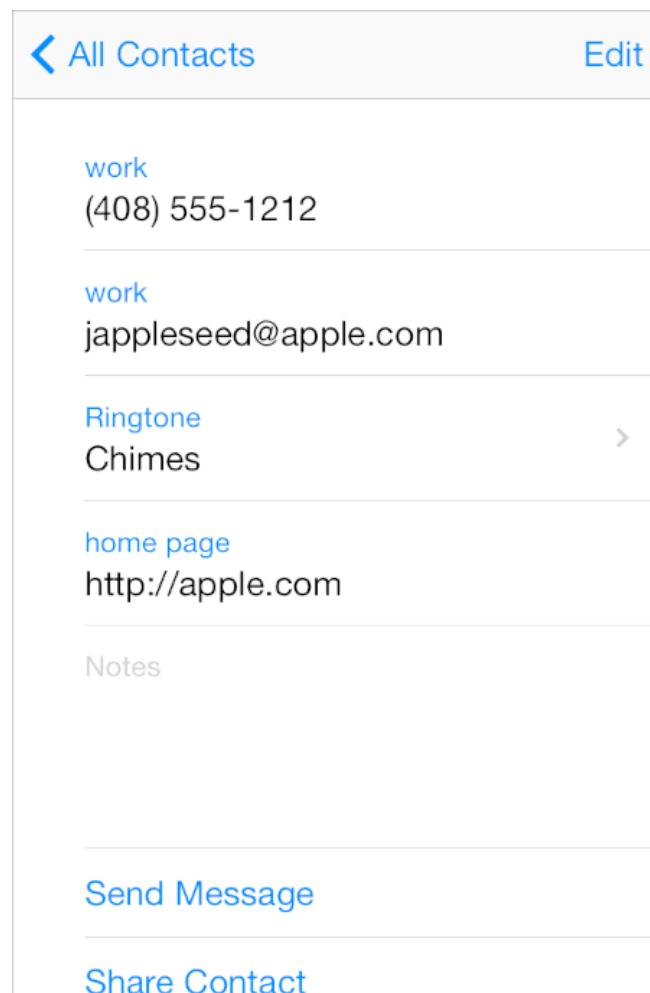
可以使用复杂手势作为完成任务的快捷方式，但不能是唯一的执行方式。尽可能给用户提供简单直接的操作方式，哪怕需要一两次额外的点击。简单的手势让用户更沉浸于体验和内容，而非交互本身。

通常，应避免定义新的手势，除非你的 app 是游戏。在游戏和其他沉浸类 app 中，自定义手势可以成为体验乐趣的一部分。但在那些帮助人们完成重要事务的 app 中，最好使用标准手势，这样用户就不需要努力去发现或记住它们。

在 iPad 上，可以考虑使用多指手势。iPad 的大屏幕为自定义多指手势带来了广阔空间，包括那些多人操作的手势。尽管复杂的手势并适用于所有 app，但对于那些人们会花费大量时间在其中的应用来说，例如游戏或创造内容的 app，这会提升体验。另外要记住，非标准的手势通常不容易被发现，不要让这类手势成为执行操作的唯一方式。

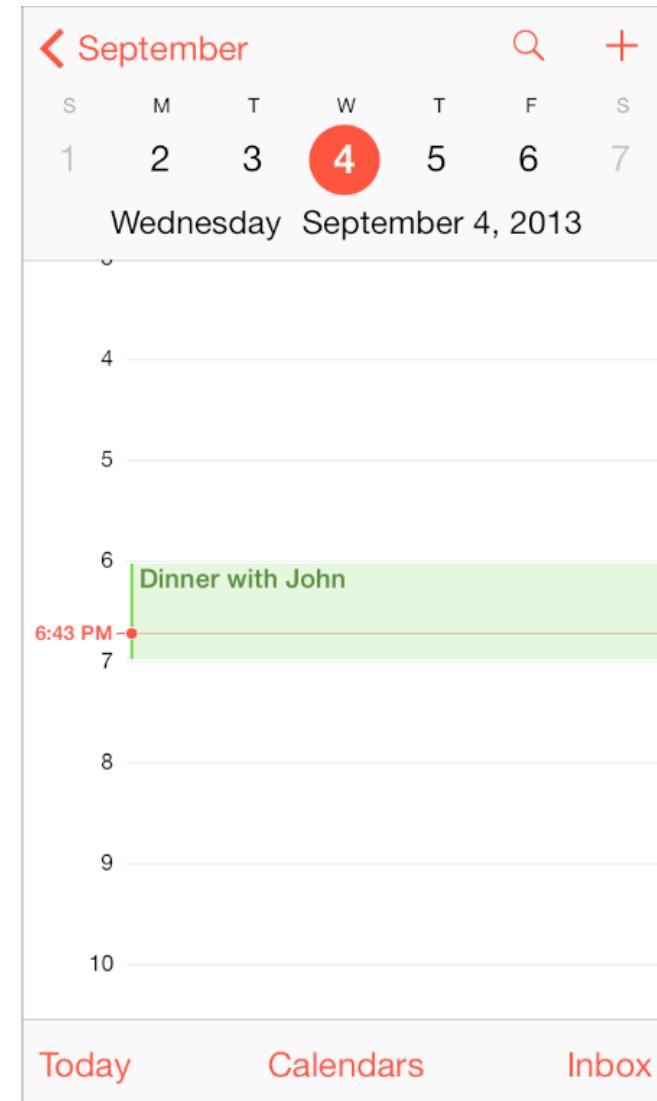
交互性元素引人触控

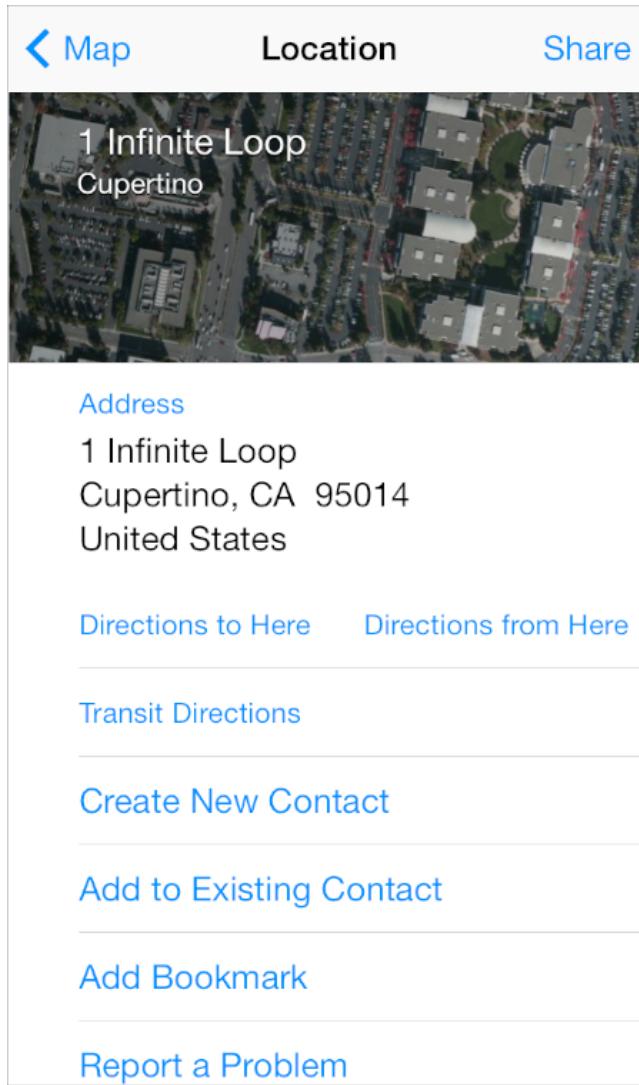
为达到显著的交互性，内置 app 使用了各式各样的暗示，包括颜色、位置、情境和富有含义的图标和标签。用户基本不需要额外的装饰来告诉他们页面中的元素是可交互的或者暗示它会用来做什么。



主题色 (key color) 会给用户以强烈的交互性视觉指示，尤其是在那些没有大量使用其他颜色的 app 中。在「联系人」中，蓝色表明了那些交互性元素并给 app 以一个统一且易于辨认的视觉主题。

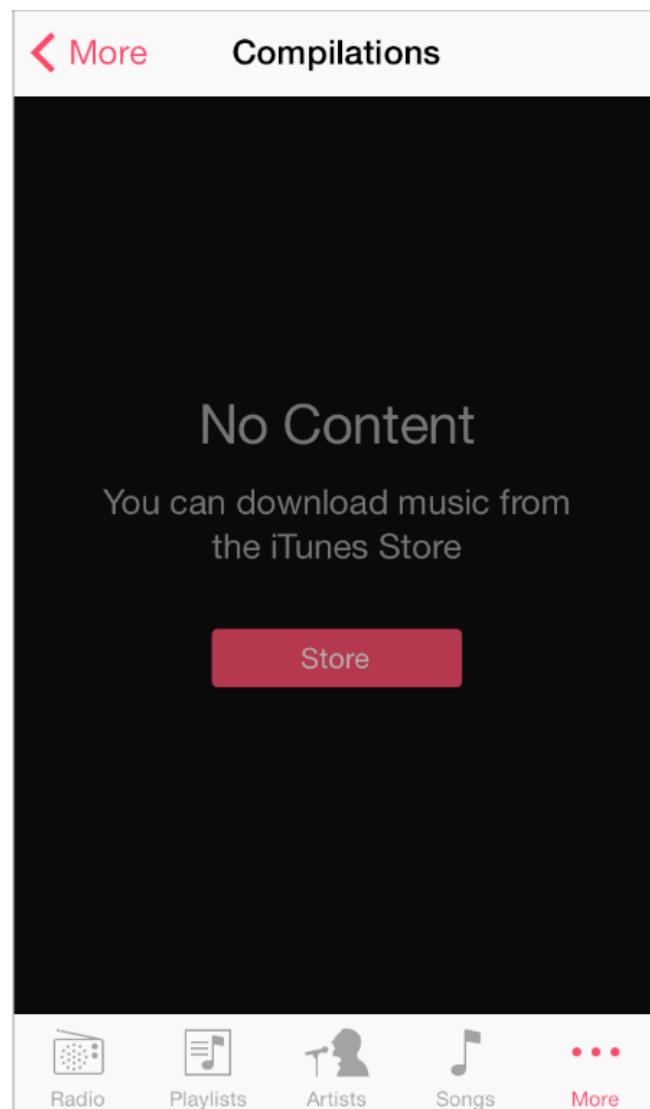
返回按钮使用了一些暗示来传达其交互性和传达其功能：它出现在导航中，其中显示着一个后退的箭头和一个描述上一个页面的标题，而且通常会使用一个主题色。



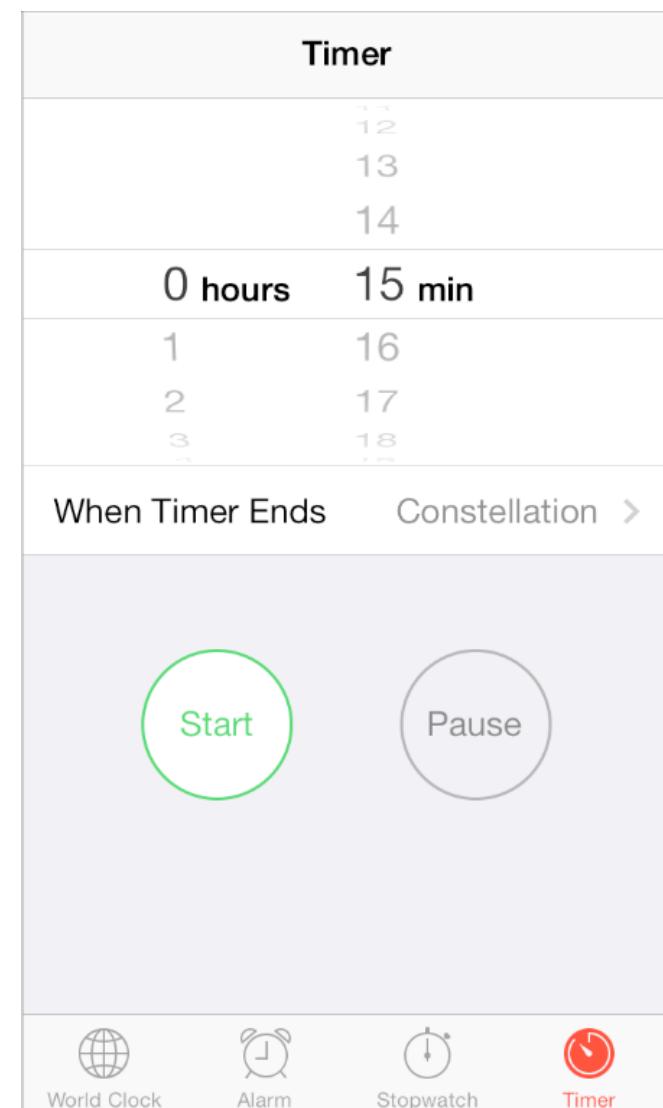


图标或者标题可以提供清晰的行动号召来吸引用户点击。例如，在「地图」中的标题——如「添加书签」和「到这里的路线」——清晰地描述了用户可以进行的操作。结合主题色和行动性的标题，便不再需要按钮边框和其他装饰。

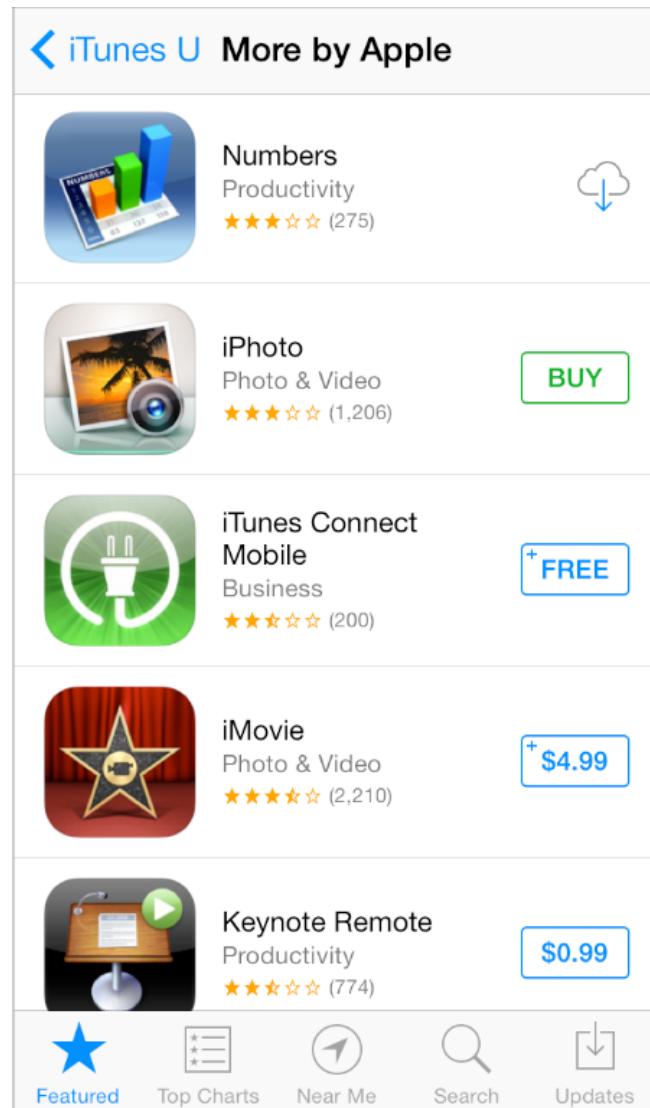
在内容区域，只在需要时为按钮添加边框或背景。条栏、操作列表和警告框中的按钮不需要边框，因为用户知道这些区域的大部分项目是可交互的。另一方面，在内容区域可能会需要边框或背景，以将其和内容的其他部分区别开来。例如，「音乐」、「时钟」和 App Store 在几个特定情境中都使用了有边框的按钮。



「音乐」用按钮背景将其和上方的解释性文本区分开来。



「时钟」在「秒表」和「计时器」页面中使用了有框按钮，使得用户即使身处错综复杂的环境也能注意到「开始计时」和「暂停」按钮并轻松点击。

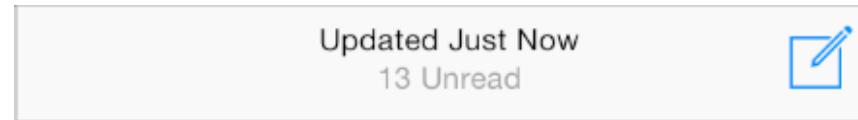


App Store 在表格列表的每一行中都使用了有框按钮，以强调点击某行获取更多信息和点击购买按钮之间的区别。

反馈增进理解

反馈有助于用户了解 app 正在做什么、发现他们接下来能做什么以及了解他们的操作会有什么结果。UIKit 的控件和视图提供了多种类型的反馈。

尽可能在用户界面中整合状态和其他相关的反馈信息。无需采取任何操作或从内容中打断，用户就可以获得相应类型的信息。例如，「邮件」会在工具栏中显示更新状态，这样不会和用户内容相冲突。



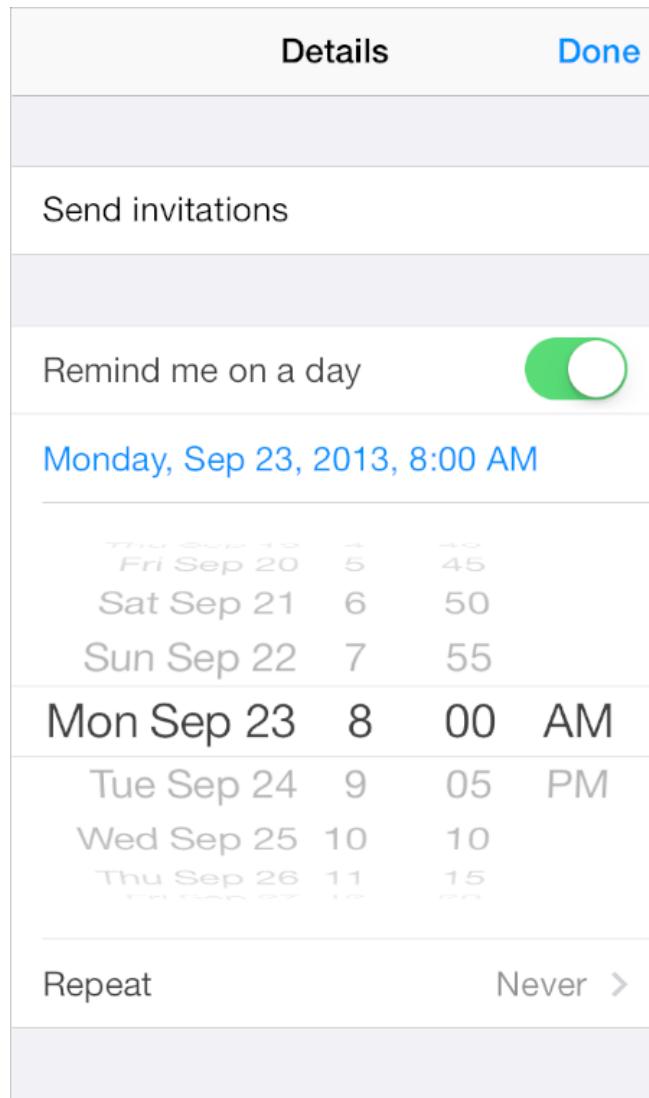
避免不必要的警告框。警告框是一种高效有力的反馈形式，但它只应用于传达最重要——理想情况下可操作——的信息。如果用户看过太多没有包含重要信息的警告框，他们很快就会学会忽略这些提示。如需了解更多关于使用警告框的信息，请参阅「警告框」（第 165 页）。

信息输入轻松容易

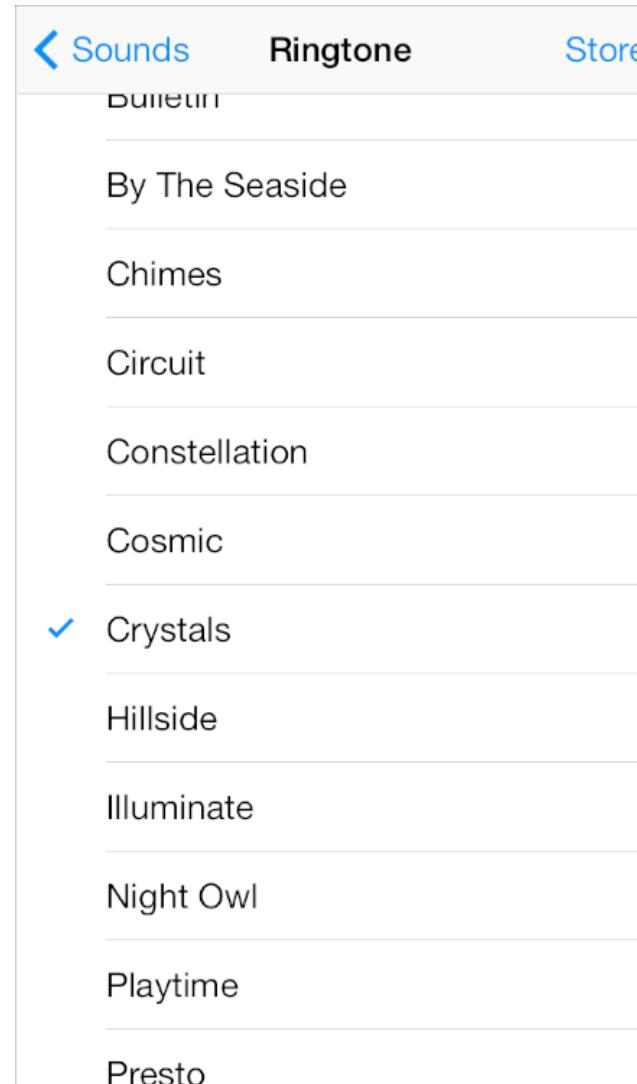
无论人们是点击控件还是使用键盘，信息输入都会耗费时间和精力。如果在 app 解决问题前就要求人们输入太多信息，那他们会感觉受挫。

使用选项来让用户输入变得轻松容易。例如，你可以使用选择器或者表格视图代替文本框，因为对大多数人来说，从一个列表中找到并选中某一项比输入文字要容易。

「提醒事项」中的时间选择器



「设置」中的选项列表



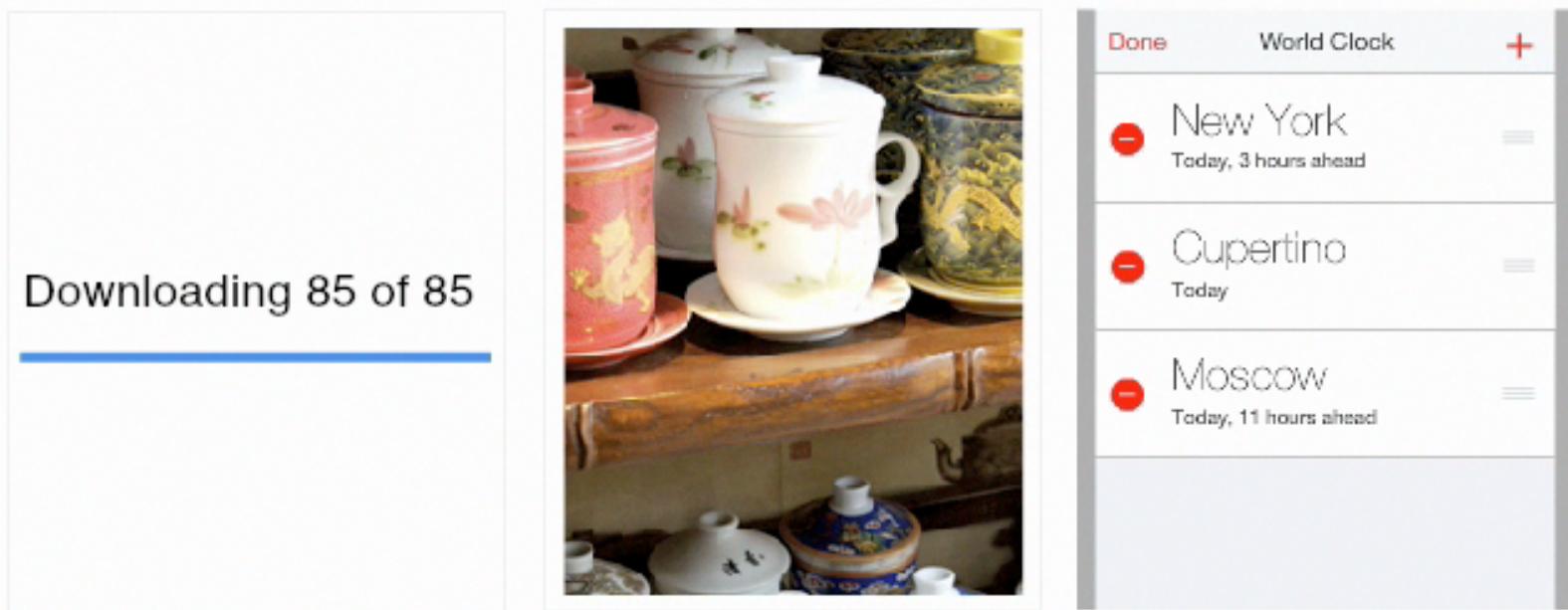
适当时，从 iOS 获取信息。人们在他们的设备中存储了很多信息。如果你可以很容易地自己找到这些信息，那就不要求人们输入它们，例如他们的联系人或日历信息。

通过给予用户有用的信息来平衡信息输入的请求。付出和回报的感觉有助于用户觉得他们正在你的 app 中取得进展。

动画

漂亮而精致的动画效果遍及 iOS 界面，它们让 app 体验更为诱人、更具活力。恰到好处的动画效果可以：

- 传达状态并提供反馈
- 增强直接操控的感受
- 帮助人们将他们行为的结果可视化



请查看此文档的 HTML 版本以播放视频。

谨慎地添加动画效果，尤其是在并不提供沉浸体验的 app 中。过度或者无意义的动画效果会阻塞 app 流程、降低性能并分散用户注意力。

尤其要有目的并克制地使用动画效果和 UIKit 动态行为，并确保其效果经过测试。适当地使用动画效果，可以提升用户感受和乐趣；但过度使用动画会让 app 看上去无所适从、难以操控。

如果合适，创建与内置动画效果相一致的自定义动画。人们习惯于内置 iOS app 优雅的动画效果。事实上，人们往往把视图之间流畅的转场效果、改变设备方向时的流体响应以及基于物理原理的滚动看成是 iOS 体验的一部分。除非你正在创建一个沉浸型的 app（例如游戏），自定义动画应当和内置的动画效果相差无几。

在你的 app 中使用一致的动画效果。和其他类型的自定义设计一样，最重要的是要一致地使用自定义动画，以便用户可以随着使用你的 app 而积累经验。

一般来说，要确保自定义动画真实可信。人们往往愿意接受外观上的艺术创新，但当他们体验到毫无意义或违背物理规律的动态效果时仍然会感到怪异和困惑。例如，如果你通过在屏幕顶部往下滑动触发一个视图，你应当可以通过向上滑动关闭这个视图，这样做有助于用户想起视图是从何而起。如果你通过在屏幕底部往下滑动来关闭同一个视图，你会打破用户认为屏幕顶部存在可用视图的心智模型。

品牌化

成功的品牌化不仅仅是为 app 注入品牌资产。最优秀的 app 会将现有品牌资产和独特的外观整合起来，给用户带来愉悦而难忘的体验。

iOS 让品牌化变得容易，使用自定义图标、颜色和字体就能创建一套独特的 UI，从而让你的 app 和其他人区分开来。当你在设计这些元素时，请记住以下两点：

- 每一个自定义元素都要好看而且正常运作，但它也需要看上去从属于 app 中的其他元素，无论这些元素是自定义的还是标准的。
- 为了让其在 iOS 7 中显得自然，你的 app 不需要和内置 app 长得很像，但它确实需要融合依从、清晰和纵深（如需了解更多关于这些设计主旨的信息，请参阅「[为 iOS 7 而设计](#)」（第 9 页））。花时间去弄清楚在你的 app 中什么是依从、清晰和纵深，并将其通过你的自定义元素传达出来。

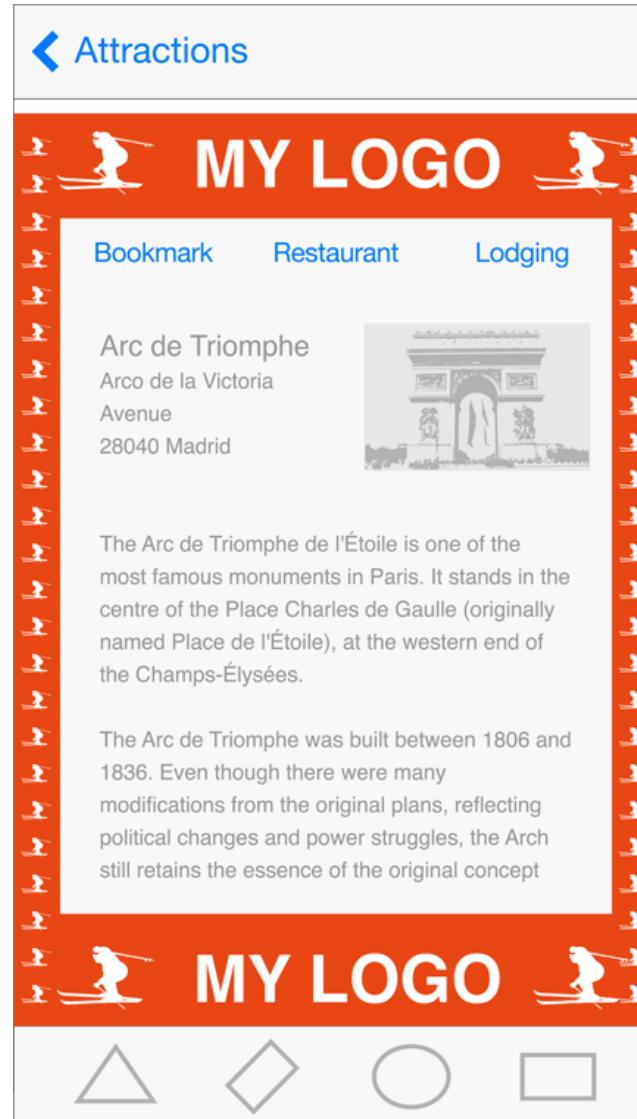
当你需要在 app 中提醒用户品牌的存在时，请遵循以下准则：

以优雅谦逊的方式整合品牌资产。人们使用你的 app 是为了搞定问题或者娱乐；他们并不希望感觉像是在被强迫观看一个广告。为了获得最佳的用户体验，你需要通过你所选择的字体、颜色和图像安静地让用户感知到品牌的存在。

推荐



不推荐



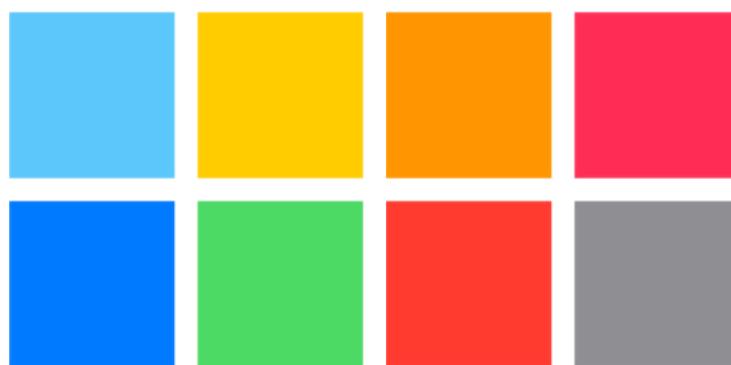
不要挤占人们所在意的内容的空间。例如，在屏幕顶部显示一个附加的固定条栏，只用来显示品牌资产，这会意味着内容的显示空间变少了。相反，尊重用户内容，并考虑以更少干扰的方式遍布展现品牌，例如使用自定义颜色、字体、或巧妙地自定义屏幕的背景。

不要在整个 app 中处处展现你的标识。移动设备的屏幕相对来说很小，标识的每一次展现都会挤占用户所希望看到的内容的空间。更重要的是，在 app 中展现标识和在网页中显示标识目的不同：用户常常会访问了一个网页而不知道它的所有者，但要说用户不用看应用图标就会打开一个 iOS app，这不太可能。

颜色和字体

色彩增进沟通

在 iOS 7 中，颜色有助于暗示交互性、传达活力并提供视觉上的一致性。内置 app 使用了一系列纯粹干净的颜色，使得它们无论是单独还是整体看起来都非常棒，而且还包含了亮色和暗色两种背景。



如果你要创建多种自定义颜色，请确保它们在一起会协调。例如，如果粉色对于你的 app 风格来说必不可少，那你应当创建一系列相配的粉色来用在整个 app 之中。

注意在不同情境下的颜色对比。例如，如果在导航栏背景和条栏按钮标题之间没有足够的对比，用户会很难看到这些按钮。一个经验法则是，需要区分的颜色之间的对比度至少要达到 50%。在设备上查看颜色对比以测试效果，这要在不同的光线情况中进行，包括晴天的户外。

建议：一种发现需要更高对比度的区域的方式是，降低 UI 的饱和度并以灰度模式查看其显示效果。如果你在交互和非交互元素或灰度版本的背景之间很难发现区别，你可能需要增加这些元素之间的对比。

当你自定义条栏的颜色时，要将半透明的条栏和 app 内容考虑进去。如果你需要创建一个条栏颜色以匹配特定颜色，例如当前品牌中的一个颜色，你可能需要试验许多颜色才能得到你想要的效果。条栏的外观同时受到 iOS 内置的半透明效果和条栏背后的 app 内容的影响。

API 备注：如果需要给条栏按钮项目着色，请使用 `tintColor` 属性；若要给条栏本身着色，则使用 `barTintColor` 属性。如需了解更多关于这些条栏属性的信息，请参阅《`UINavigationBar Class Reference`》、《`UITabBar Class Reference`》、《`UIToolbar Class Reference`》和《`UISearchBar Class Reference`》。

要考虑色盲人群。大多数色盲用户很难区分红色和绿色。测试你的 app，确保你没有在任何地方将红色和绿色作为区分两种状态或值的唯一方式（一些图像编辑软件有这样的工具可以帮助你验证色盲的情况）。通常，使用不止一种方式去表示元素的交互性是一个不错的想法（如需了解更多关于在 iOS 7 中表示交互性的信息，请参阅「[交互性元素引入触控](#)」（第 38 页））。

考虑选择一个主题色以显示交互性和状态。内置 app 中的主题色包括「标签」的黄色和「日历」的红色。如果你要定义一个主题色以显示交互性和状态，确保你的 app 中的其他颜色不会与之冲突。

避免在交互和非交互元素中使用相同颜色。颜色是用户界面元素显示其交互性的一种方式。如果交互和非交互元素用一样的颜色，那么用户会很难知道他们点的是哪里。

颜色增进沟通，但并不总是你所希望的方式。每个人看到的颜色都不同，而且在许多文化中人们对颜色的含义如何对应也有所不同。花点时间去了解你所使用的颜色在其他国家和文化中可能会被如何解读。你需要尽可能确保 app 中的颜色传达着合适的信息。

大部分情况下，不要让颜色干扰用户。除非颜色是你的 app 主旨中不可或缺的部分，否则它通常应仅是一种恰到好处的升华。

文字清晰易读

最为首要的是，文字必须清晰易读。如果用户根本看不清你的 app 中的文字，那字体设计得再漂亮也无济于事。在 iOS 7 app 中使用「动态字体」（Dynamic Type），可以实现：

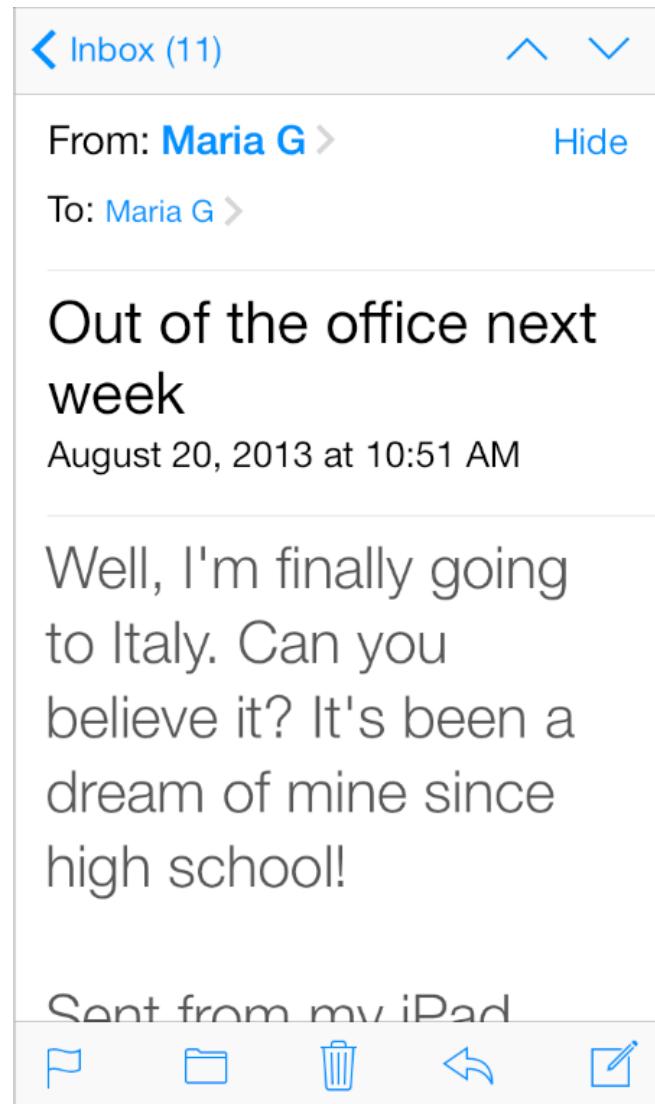
- 自动调整每一种字体大小的字间距和行高
- 为不同语义的文本块指定不同的文本样式，例如正文、脚注或大标题
- 文字会适当响应用户对文字大小的设置更改（包括辅助功能中的文字大小）

注意：如果你在使用自定义字体，仍然可以根据系统设置的文字大小缩放字体。当用户改变设置，你的 app 要做出适当的响应。

对你来说，使用「动态字体」可能需要一些工作量。如需了解如何使用文本样式并确保你的 app 在用户改变文字大小设置时得到通知，请参阅《Text Programming Guide for iOS》中「Text Styles」一节。

在响应文字大小的变化时，优先让内容变化。对用户来说，不是所有内容都同等重要。当用户选择一个更大的文字大小时，他们想让他们所在意的内容易于阅读；他们一般并不希望页面中的每一个字都变大。

例如，当用户在「辅助功能」中选择了一个更大的文字大小，「邮件」中的收件人和消息正文以大号文字显示，但一些不太重要的文本——例如时间和发件人——还是以较小大小显示。



如果合适，在用户选择不同的文字大小时调整布局。例如，在用户选择一个小号的文字大小时，你可能会想将单栏的正文文本布局更改为两栏布局。如果你决定为不同文字大小调整布局，比起在每一种可能的大小都改变布局，你可能要选择以大小分组来实现——如较小、中等和较大。

Make sure all styles of a custom font are legible at different sizes. One way to do this is to emulate some of the ways iOS displays font styles at different text sizes. For example:

确保自定义字体的所有样式在不同大小下都清晰可见。这样做的一个途径是，模仿 iOS 在不同文字大小下显示字体样式的一些方式。例如：

- 即便用户选择了最小文字大小，文字也不应小于 22 点。作为对照，正文样式在大字号下使用 34 点字体大小作为默认文字大小设置。
- 通常来说，每一档文字大小设置的字体大小和行间距的差异是 2 点。例外情况是两个标题样式，在最小、小和中等设置时都使用相同字体大小、行间距和字间距。
- 在最小的三种文字大小中，字间距相对宽阔；在最大的三种文字大小中，字间距相对紧密。
- 标题和正文样式使用一样的字体大小。为了将其和正文样式区分，标题样式使用加粗效果。
- 导航控制器中的文字使用和大号的正文样式文字大小（明确来说，是 34 点）。

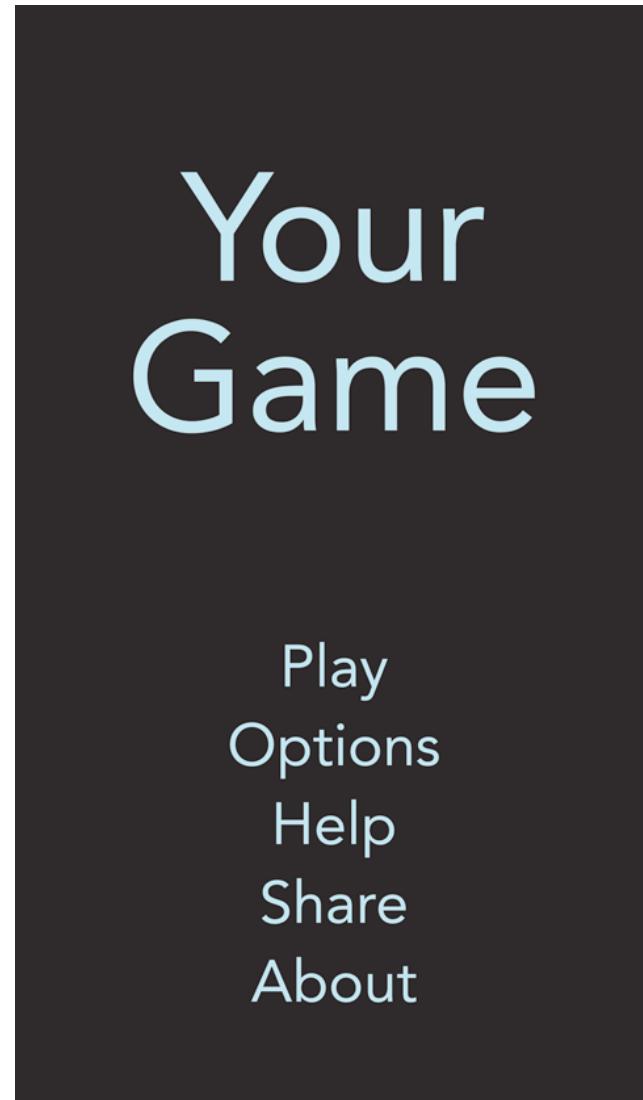
- 文本通常使用常规体和中等大小，而不是用细体和粗体。

通常，在你的 app 中只使用一个字体。几个不同的字体混搭会让你的 app 像是杂乱无章。相反，可以使用一个字体和仅仅几个样式和大小。根据不同的语义用途，使用 `UIFont` 文本样式 API 以定义不同的文本区域，例如正文或标题。

不推荐



推荐



图标和图像

应用图标

每个 app 都需要一个漂亮的应用图标。很多时候，人们会完全根据你应用图标的样子，来建立关于你的产品品质、目的和可靠性的第一印象。

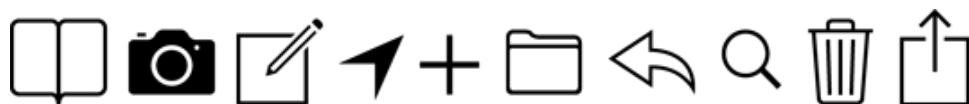


当你在构思应用图标时，你应将以下几点牢记于心。此外，当你准备开始创建图标时，请参阅「[应用图标](#)」（第 175 页）一节以获得详细指导和说明。

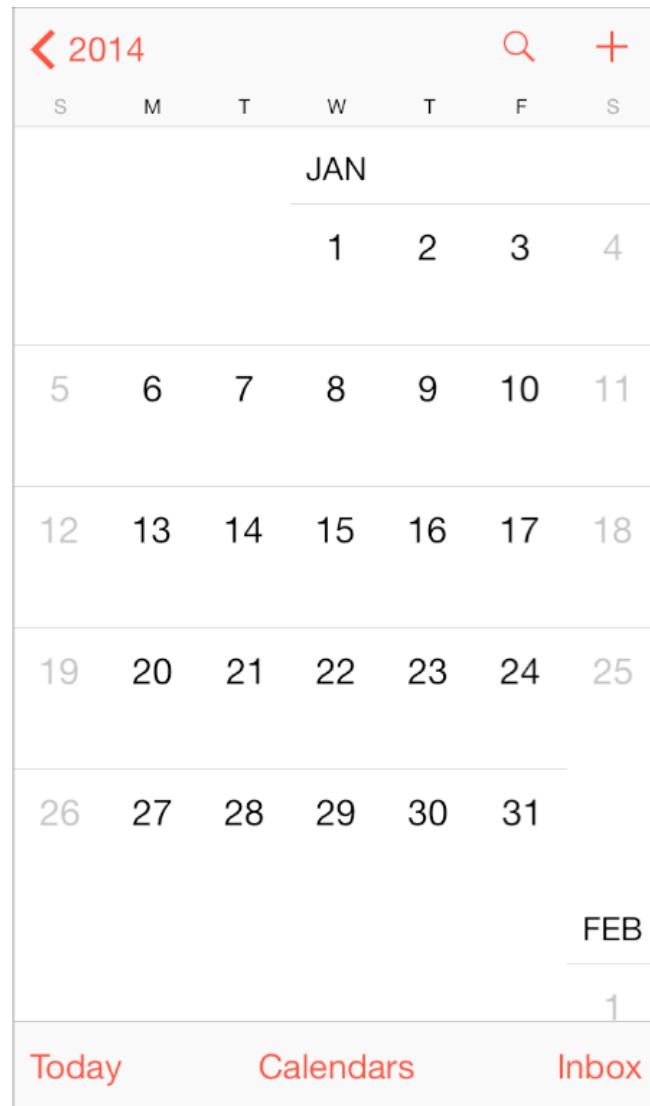
- 应用图标是产品品牌的重要组成部分。要将设计图标看成是向用户讲述产品故事以及构建情感联系的机会。
- 优秀应用图标都是独特、简洁、动人和令人难忘的。
- 应用图标在不同尺寸不同背景上都要显示良好。那些在大尺寸图标上可以增强效果的细节元素在小尺寸上可能会变得模糊不清。

条栏图标

iOS 提供了大量代表常见任务和内容类型小图标，它们可以用在标签栏、工具栏和导航栏中。最好尽可能地使用内置图标，因为用户已经熟知其含义。



如果需要表示自定义操作或自定义内容类型，你可以创建自定义条栏图标。设计这些线条流畅的图标和设计应用图标不太一样。如果你需要创建自定义条栏图标，请参阅「[条栏按钮图标](#)」（第 221 页）了解如何创建。



请注意，在导航栏或工具栏中，你可以用文本来替代图标以显示项目。例如，「日历」使用「今天」、「日历」和「收件箱」替代工具栏上的图标。

为帮助你决定在导航栏或工具栏中是使用文本还是图标，你可以算算同一时间有多少图标在屏幕上可见。屏幕上出现过多的图标会让 app 看上去很复杂。另外要注意，这一决定可能因 iPhone app 和 iPad app 而有所不同，因为 iPad app 往往在条栏文字上有更多空间。

图像

iOS app 往往包含丰富的图形元素。无论是在展示用户照片还是创造自定义作品，你都应当遵循以下这些原则：

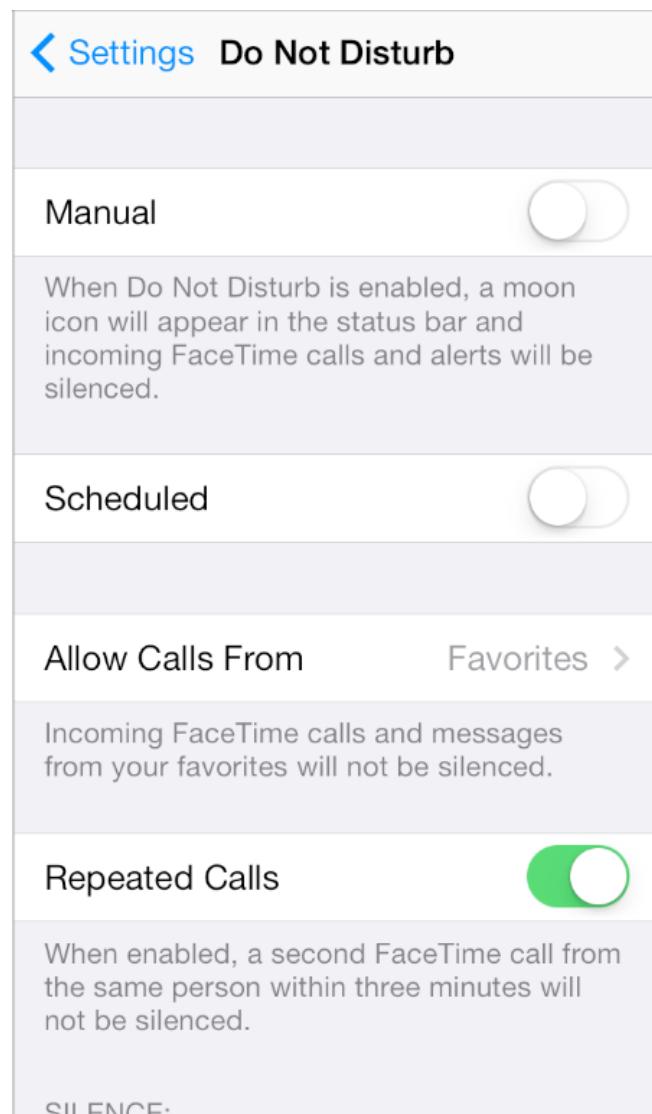
支持 Retina 显示屏。确保你的 app 中所有的插图和图形元素都有提供 @2x 资源。

以原始宽高比显示照片和图形，且放大比例不超过 100%。你不会希望自己 app 中的差图和图形元素看上去失真或过大。让用户自己选择是想要放大图像还是缩小图像。

不要在你的设计中使用 Apple 产品的复制图像。这些图像都是有版权的，而且这些产品设计会频繁变更。

术语和措辞

你在 app 中所展示的每一个字都是你与用户对话的一部分。要将这样的对话看成是在你的 app 中清晰表达和营造轻松氛围的机会。



「设置」是一个所有用户都会用到的 app，它用简单而直接的语言来描述用户能做的事。例如，「设置 > 勿扰模式」并没有使用技术术语去解释各种设置的效果，因为那对普通用户来说可能很难理解。

确保你所使用的术语能被用户理解。基于对用户的了解去判断你即将使用的词汇和短语是否恰当。例如，对于那些面向普通用户的 app，技术术语几乎没有用处，但对于那些为精通技术的用户设计的 app 来说，使用技术术语可能又会备受赞赏。

使用轻松友好的语气，但不要过分亲密。要避免语气生硬或过于正式，但也不要过于虚情假意或曲意逢迎。要记住，用户可能会频繁看到你的 UI 中的文字，起初看似精妙的表达看多了也许会变得令人生厌。

像报纸编辑那样思考，找出那些冗余和不必要的字词。如果你界面中的文本简短而直接，用户便能迅速轻松地理解。找出那些最重要的信息，简明地表述并突出显示，这样一来人们就不用在一大堆文字中寻找他们要找的信息或下一步要做什么了。

给控件加上简短的文本标签或为人熟知的图标。这样人们一眼就能知道这个控件的作用。

日期信息表述要务必准确。一般来说，在你的界面中使用像今天和明天这样友好的词汇来显示日期信息是合适的。但如果你没有考虑用户当前所在的位置，这样的表达可能就会让人困惑。例如，要考虑到某些事件正好在午夜前开始。对于同一时区的用户，该事件是发生在今天，但对那些更早时区的用户，这个事件可能已经在昨天发生。

撰写一段优秀的 App Store 产品描述文案，要尽量利用这个和潜在用户沟通的机会。除了准确描述你的 app 并突出你认为人们可能会喜欢的品质之外，还需要确保：

- **更正所有的拼写、语法和表达错误。**尽管这些错误不会被每个人发现，但在一部分人心中会对你的 app 品质留下负面印象。
- **尽量少使用全部大写的词语。**偶尔使用全部大写的词语有助于吸引人们的注意力，但当整段文字都是大写时，它会变得难以阅读，看上去像是在对用户大吼大叫。
- **考虑描述某些 bug 修复。**如果你的 app 的新版本包含一些用户所期待的 bug 修复，那在描述中提到这些修复将会是个好主意。

与 iOS 整合

与 iOS 整合可以在这个平台上给用户带来迷人而愉悦、宾至如归的体验；但这不代表要创建一个和内置 app 一模一样的 app。

领会驱动 iOS 的设计主旨——这些在「[为 iOS 7 而设计](#)」（第 9 页）中有所表述——并思考你的 app 应如何传达这些主旨，这是将你自己独一无二的 app 整合到平台的最好方式。当这样做时，请遵循本节提到的准则，这会有助于给用户以他们所期待的体验。

正确使用标准 UI 元素

尽可能使用 UIKit 提供的 UI 元素，这会是一个不错的做法。当你使用标准 UI 元素而非创建自定义元素时，你和你的用户都会受益：

- 如果 iOS 引入了一个重新设计的外观，标准 UI 元素会随之自动更新，但你自定义的元素则不会。
- 标准 UI 元素往往会提供多种自定义外观和行为的方式。例如，所有视图（即，从 UIView 中继承而来的对象）都可以改变颜色，这会让给 app 添加颜色变得容易。如需了解更多给 UI 元素添加颜色的信息，请参阅《iOS 7 UI Transition Guide》中「Using Tint Color」一节。
- 人们习惯于标准 UI 元素，所以他们会立刻了解在你的 app 中该如何使用它们。

为充分发挥使用标准 UI 元素的优势，你必须：

每一个 UI 元素都遵循设计规范。当一个 UI 元素看上去和用起来都符合用户期望时，他们便可以根据已有的经验在你的 app 中去使用它们。你可以在「[条栏](#)」（第 120 页）、「[内容视图](#)」（第 131 页）、「[控件](#)」（第 152 页）和「[临时视图](#)」（第 165 页）中找到这些 UI 元素的规范。

不要混用 iOS 不同版本的 UI 元素样式。你不会希望用户感到困惑，因为和当前设备中运行的版本相比，显示的 UI 元素看上去却属于老版本的 iOS。

通常，不要为标准交互行为创建一个自定义 UI 元素。首先问问自己，为什么要创建一个行为方式和标准元素完全一样的自定义 UI 元素。如果你仅仅是想自定义外观，请考虑通过使用 UIKit 的自定义外观 API 或者着色来改变标准元素的样子。如果你是希望行为上有细微差异，那在你调整其属性和特性时，一定要清楚标准元素是不是可以实现你想要的效果。而如果你是需要完全自定义交互行为，则最好设计一个看上去和标准元素不太近似的自定义元素。

建议：Interface Builder 会让创建标准 UI 元素变得容易，比如使用自定义外观的 API，设置属性和特性，以及在你的控件中添加自定义和系统提供的图标。如需了解更多关于 Interface Builder 的信息，请参阅《Xcode Overview》。

不要为系统定义的按钮和图标赋予其他含义。iOS 提供了很多可以在你的 app 中使用的按钮和图标。确保你理解了这些按钮和图标的文档和含义；而不是基于自己对其外观的阐释。（你可以在「工具栏和导航栏按钮」（第 124 页）和「标签栏图标」（第 127 页）中找到这些图标的含义。）

如果不能为你 app 中的功能找到一个有合适含义的系统按钮或图标，可以自己创建一个。参阅「条栏按钮图标」（第 182 页），你可以了解有助于你设计自定义图标的准则。

如果你的 app 有着沉浸式的任务或体验，那么创建完全自定义的控件可能会是合理的做法。由于你正在创建一个独特的环境，而在这样的 app 中，探索如何与环境互动也是用户所期待的一种体验。

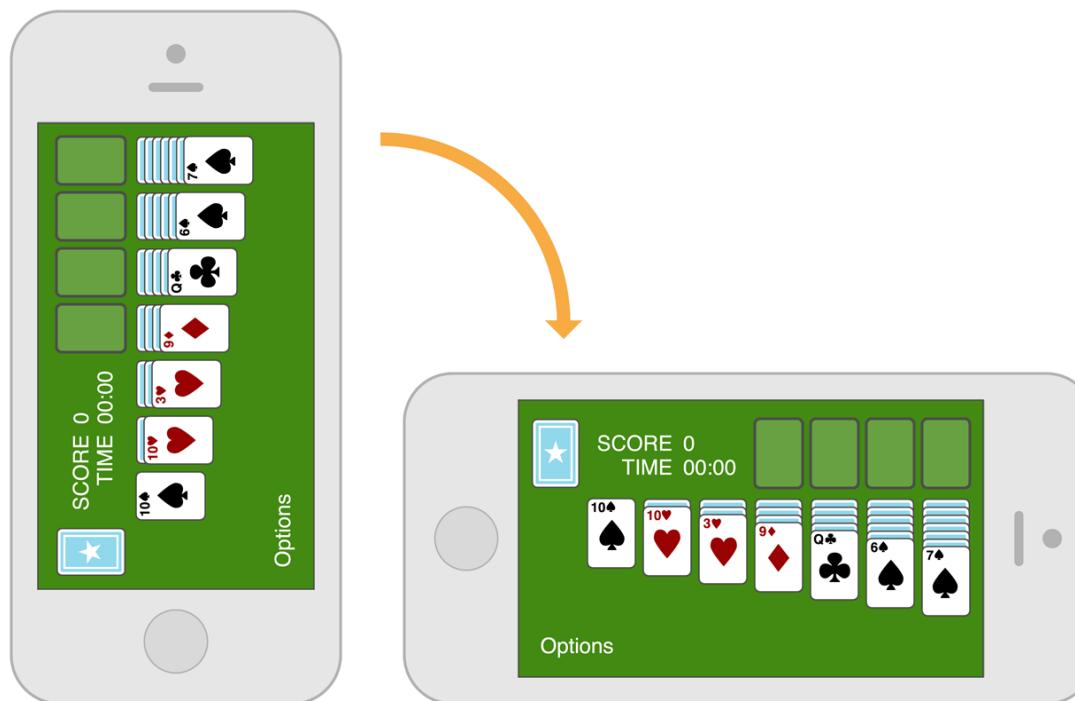
响应设备方向变化

人们通常希望能在任何方向下使用 iOS 设备，所以最好让你的 app 在那个时候作出恰当的响应。

无论是横屏还是竖屏方向，始终保证突出重要内容。这将是你最优先的目标。人们使用你的 app 来浏览并与他们所关心的内容交互。在设备发生旋转时改变焦点，这会让人们迷惑并让他们觉得已经对 app 失去控制。

一般来说，要让 app 在所有方向上都要能正常运行。人们希望在不同方向上使用你的 app，所以最好能满足他们的这一期望。特别是 iPad 用户，他们会希望能在当前握持设备的任意方向上使用你的 app。但的确有一些 app 只能在竖屏或横屏方向下运行。如果你的 app 确实需要只在一个方向上运行，你应当：

- **以默认支持的方向启动你的 app，忽略当前设备的方向。**例如，假设有一个仅支持横屏的游戏或多媒体播放 app，即便设备当前是竖屏方向，以横屏方向启动这个 app 也是合适的。这样，当人们在设备竖屏时启动 app，他们便会知道要旋转设备到横屏方向以查看内容。



- **避免显示一个提示用户旋转设备的 UI 元素。**如有需要，无需在界面中添加不必要的东西，以默认支持方向运行 app 也能清楚地提示用户需要旋转设备了。
- **支持一个方向的两种变化。**例如，如果某个 app 只能以横屏运行，无论人们是将主屏幕按钮放在右边还是左边去握持设备，他们应该都能使用这个 app。而如果用户在使用过程中 180 度旋转设备，app 最好也能 180 度旋转内容以及时响应。

如果你的 app 将设备方向的变化作为一种用户输入方式，则以 app 特定的方式处理设备旋转。例如，一款允许用户通过旋转设备移动方块的游戏，它不可能以旋转屏幕的方式响应设备旋转。在像这样的情况下，你应该支持以所需方向的两种变化形式启动，并允许人们在开始 app 主要任务前在两种变化中切换。一旦人们开始主要任务，马上以 app 特定的方式响应设备变化。

在 iPhone 上，在响应设备方向变化时考虑用户需求。用户尝尝会旋转他们的设备到横屏方向，因为他们想要「看到更多。」如果你仅仅是将内容放大，你不一定能满足用户预期。相反，重新调整文本行数，如果需要还可以调整界面布，以便可以在屏幕上看到更多内容。

在 iPad 上，力图支持所有方向以满足用户期望。iPad 的大屏幕降低了用户旋转设备至横屏以「看到更多」的欲望。而且由于人们不再留意设备的框架限制和主屏幕按钮的位置，他们不太会认为该设备有默认方向。

当你在设计你的 iPad app 该如何在所有方向上提供优秀的体验时，请遵循以下准则：

- **考虑改变辅助信息或功能的显示方式。**虽然要让最重要的内容时刻处于焦点，但你也可以通过改变次级信息的显示方式以响应设备旋转。

例如，一款以横屏方向展现矩形棋盘的 iPad 游戏，需要在竖屏方向时重新调整棋盘以良好显示。相比在竖屏方向时垂直拉伸棋盘——或者让顶部或底部留空——游戏可以在多出来的空间里显示附加信息或物体。

- **避免毫无意义的布局变化。**在所有方向上一致的体验会让用户在旋转设备时可以维持原有的使用习惯。例如，如果你的 iPad app 在横屏方向中显示一个矩阵列表，那就不需要在竖屏时以列表形式显示相同的信息（即使你可能会调整矩阵的尺寸）。
- **尽可能避免在旋转时重新定义信息和文本格式。**试着在所有方向中维持相似的格式。如果人们是在 app 中阅读文字，那尤其重要的是要在旋转设备时帮助他们停留在原来的位置。

如果某些格式的调整不可避免，那使用动画以帮助人们跟随变化。例如，如果你在旋转方向时中增加或移除了一栏文字，你可以将栏的变化隐藏并让新的布局简洁地淡入显示。为了让自己设计出合适的屏幕旋转行为，你可以想象一下如果你是在和现实世界中的这些内容发生物理性的互动，该是怎样的样子。

- **为每一个方向提供单独的启动画面。**当每个方向都有单独的启动画面后，无论当前设备是哪个方向，人们都能体验到平滑的 app 启动。iPad 的主屏幕支持切换屏幕方向，所以人们有可能在以和上一个 app 相同的屏幕方向启动你的 app。

弱化文件和文档处理

iOS app 可以帮助人们创建和管理文件，但这并不意味着人们应该在 iOS 设备中思考文件系统的问题。

在 iOS 中，不存在类似 OS X 系统中 Finder 这样的 app，因此人们也不应该像在电脑上那样被要求和文件直接交互。特别是不应让人们面对任何会让其联想到文件元数据或存储位置的东西，例如：

- 显示文件层级的打开/保存对话框
- 关于文件权限的信息

尽量让人们不需要打开电脑上的 iTunes 就能管理文档。考虑使用 iCloud 去帮助用户访问他们在所有设备上的内容。如需了解如何在你的 app 中提供更好的 iCloud 体验，请参阅「[iCloud](#)」（第 90 页）。

如果你的 app 帮助人们创建和编辑文档，那最好提供某种文件选择器来让用户打开已有文档或创建新文档。理想情况下，这个文档选择器应该：

- **高度图形化。**用户看一下屏幕上文档的视觉形式便能轻松找到他们想要的文档。
- **让人们使用尽量少的手势来完成想要的操作。**例如，人们可以水平滚动一个已有文档的缩略图列表，然后轻点一下打开想要打开的文档。
- **包含新建文档功能。**不要让人们到其他别的地方去创建一个新文档，而是在文档选择器中让他们轻点一个占位图像去创建一个新文档。

建议：你可以使用「快速预览」（Quick Look）特性让人们在你的 app 内预览文档，即便你的 app 不能打开它们。如需了解如何在你的 app 中提供这个特性，请参阅「[快速预览](#)」（第 106 页）。

让用户确信他们的工作成果始终会被保存，除非明确地取消保存或进行删除。如果你的 app 能帮助人们创建和编辑文档，那就不要再让他们采取额外的保存操作。iOS app 应该承担起保存用户的输入内容的责任，无论是在他们打开另一个文档还是切换到其他应用时。

如果你的 app 的主要功能不是创造内容，而你允许用户在查看信息和编辑信息之间切换，那么最好提醒用户保存内容变更。在这样的场景中，最好在显示信息的视图中提供一个「编辑」按钮。当人们轻点「编辑」按钮，随之替代的是一个「保存」按钮并新增一个「取消」按钮。「编辑」按钮的变化有助于提醒人们他们正处于编辑模式，可能需要保存内容变化，而「取消」按钮则提供了不保存变更直接退出的机会。

必要时提供设置方式

虽然大多数 app 可以不需要或者推迟进行设置，但有些 app 可能会需要给用户一种更改安装或设置选项的方式。优秀的 app 能让大多数人立即上手使用，并在主要界面中提供一些途径以调整用户体验。

尽量避免让用户跑到系统「设置」中去。请记住，进入「设置」需要先从你的 app 切换离开，而你并不会希望用户这样做。

当你从大多数用户的期望来设计你的 app 功能时，就会降低对设置选项的需求。如果你需要用户信息，从系统中获取而非让用户提供。如果你决定必须要提供一些用户偶尔需要更改的设置选项，请参阅《iOS App Programming Guide》中「The Settings Bundle」一节，以了解如何在代码中支持。

如果需要，让用户在你的 app 内部进行设置。将配置选项整合到你的 app 中，这样的话人们不需要离开你的 app 去进行设置，从而让你可以即时地响应设置更改。

尽可能在主界面中提供设置选项。如果设置选项对应的是应用的主要任务，或者人们可能会频繁地更改设置，那么最好将其放到主界面中。如果人们只是偶尔才会更改 app 配置，那就将它们放到一个单独的视图中。

充分利用 iOS 技术

iOS 提供了丰富的技术，以各种方式支持用户所期待的常见任务和场景。这样的期望意味着，相比设计自定义的方式，更好的做法是将这些系统支持的技术整合到你的 app 中。

有一些 iOS 技术——例如「[多任务处理](#)」（第 84 页）和「[VoiceOver](#)」（第 114 页）——是所有 app 都应当纳入的特性。其他技术则根据 app 具体功能而定，例如处理票据和礼物卡（「[Passbook](#)」（第 82 页））、支持用户在 app 内购买（「[App 内购买](#)」（第 92 页））、在 app 内显示广告（「[iAd 富媒体广告](#)」（第 99 页））、和「[Game Center](#)」（第 94 页）整合以及支持「[iCloud](#)」（第 90 页）。

设计策略

- 「设计原则」 (第 61 页)
- 「从概念到产品」 (第 65 页)
- 「案例研究：从桌面到 iOS」 (第 69 页)
- 「在 iPhone 5 中运行」 (第 74 页)

设计原则

美学完整性

美学完整性不是用来衡量 app 的艺术表现或风格特征，而是指 app 的外观与行为是否与其功能相衬一致。

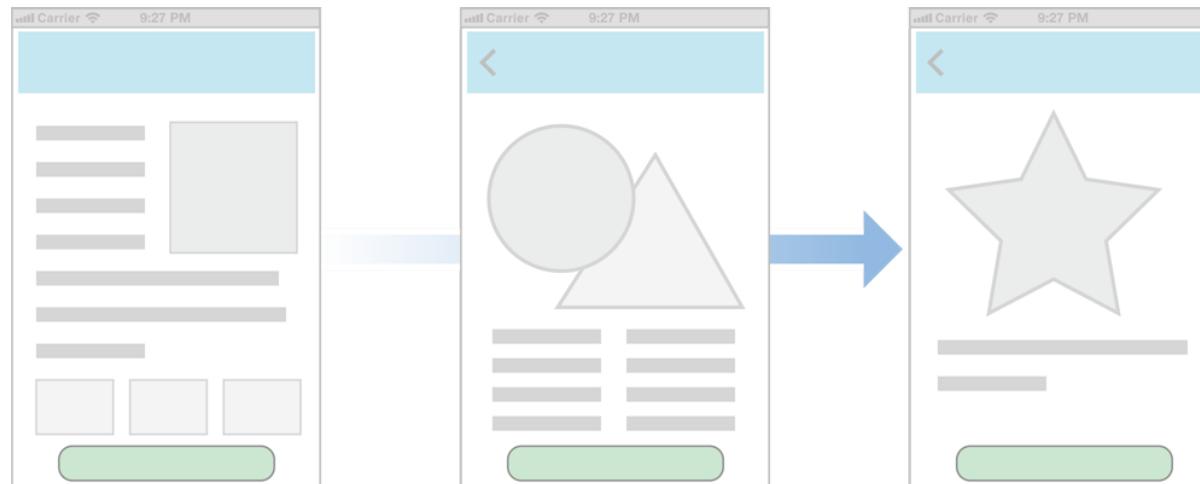


用户不止关心 app 是否实现了它所承诺的功能，他们也在很大程度上会被 app 的外观和行为所影响，尽管有时他们并未意识到这一点。例如，一款协助人们处理重要任务的 app，它会通过弱化装饰性元素并使用标准控件和可预期的行为等方式来将焦点放到任务上。这样一款清晰一致地传达出其目的和特点的 app，会让用户对其产生信任。然而，如果这个 app 使用了杂乱无章、充满干扰的 UI，用户可能会对这个 app 的可靠性和信赖度产生怀疑。

另一方面，对那些鼓励沉浸体验的 app（例如游戏），用户会期望一个充满乐趣、让人兴奋和期待探索的迷人外观。用户不希望在游戏中完成一个严肃或枯燥的任务，相反，他们期望游戏的外观和行为能够和它的目的相一致。

一致性

一致性可以让用户将 app 中的某部分界面的经验和技巧复用到其他地方，或者从一个 app 复用到另一个 app。一致性的 app 不是对其它 app 的简单复制，也不是风格上的一成不变，相反，它关注用户所习惯的方式和标准，并提供一个具有内在一致性的体验。

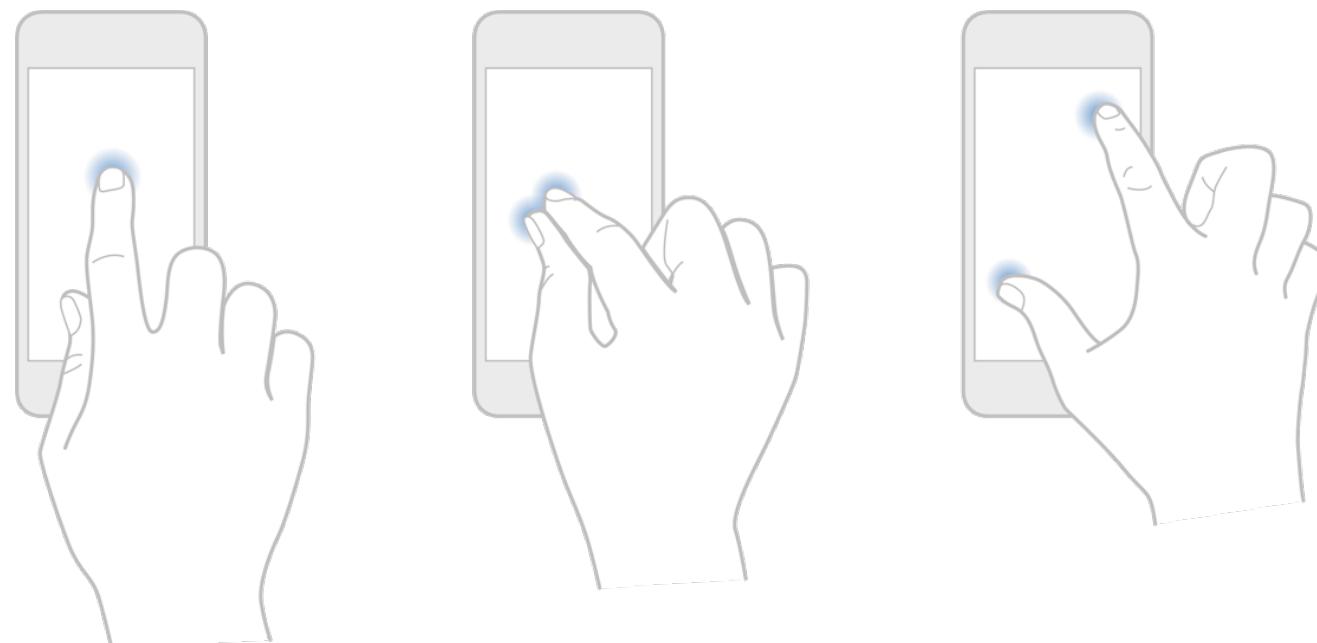


要判断一个 iOS app 是否符合一致性原则，可以通过以下几个问题来考量：

- App 是否和 iOS 标准保持一致？它是否正确地使用了系统控件、视图和图标？是否以用户所期望的方式利用了设备的特性？
- App 自身是否具有内部一致性？文本内容是否使用了统一的用辞和风格？同样的图标是不是通常意味着相同的意思？当用户在不同的位置执行同一个操作时是否符合其预期？自定义的界面元素外观和其行为是否保持一致？
- App 是否在合理范围内与之前的版本保持一致？术语和含义是不是仍然相同？基本概念和主要功能是否基本不变？

直接操控

当人们直接操控屏幕上的物体而不是使用另外的控件来操作时，他们就会更沉浸在任务当中，从而更容易地了解自己的操作会产生的结果。



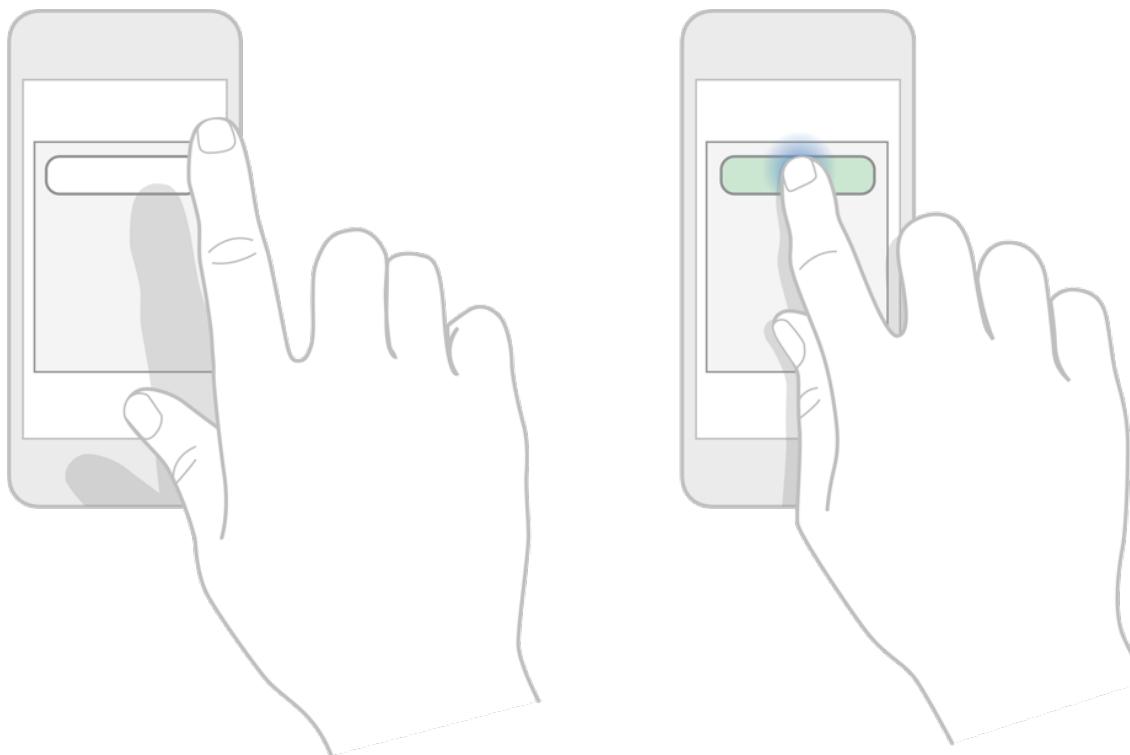
使用多点触控界面，人们可以双指开合以直接放大图片或者内容区域。在游戏中，玩家则可以直接移动屏幕上的物体并与之互动——例如，一款游戏可能会展示一个密码锁，用户可以旋转密码盘去打开。

在 iOS app 中，人们可以在如下场景里体验到直接操控：

- 旋转或以其他方式移动设备以影响屏幕上的物体
- 使用手势动作操纵屏幕上的物体
- 可以看到他们的动作有直接且可见的结果

反馈

反馈可以让人们知道系统已经收到他们的操作行为，并向其呈现操作结果，让他们了解自己的任务进程。



iOS 的内置 app 在响应用户的每一个操作行为时都提供了可感知的反馈。当用户点击列表项和控件时，它们会被短暂地高亮。而那些会持续超过几秒钟的操作，对应的控件则会显示已完成的进度。

精致的动画效果可以给用户有意义的反馈，以此帮助他们了解其行为的结果。例如，在列表中添加一个新条目时会有动画，以便让人们察觉到视觉上的变化。

声音也可以为用户提供拥有的反馈，但这不应该是唯一的反馈方式，因为用户不是所有时候都能听到他们的设备声音。

隐喻

如果 app 中的虚拟对象和操作行为是一段相似体验的隐喻，无论这些体验是基于真实世界还是数字世界，用户都会很快了解如何使用这个 app。

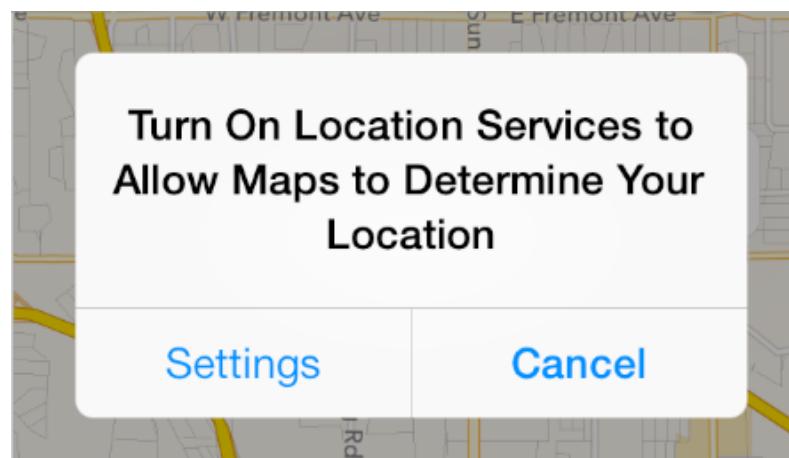
在 app 使用隐喻去展现使用方法或体验时，最好不要被它所基于的对象或行为局限。

由于用户能和屏幕直接产生互动，iOS app 为隐喻提供了广阔空间。iOS 中的隐喻包括：

- 移开图层视图，展现其下方的内容
- 在游戏中拖拽、轻敲、或扫开物体
- 点击切换开关、滑动滑块或转动选择器
- 在书或杂志中的翻页

用户控制

让用户而不是 app 去触发并控制操作。App 可以针对危险结果提出一系列建议操作或警告，但不应该剥夺用户的控制权而由 app 来做决策。优秀的 app 会找到一个恰当的平衡，在赋予用户所需要的控制权的同时，帮助其避免不希望发生的结果。



当行为和控件是熟悉且可预期时，用户会感觉 app 更为可控。如果一个操作行为简单而直接，用户就可以轻松地理解并记住它们。

人们期望在一个操作执行之前有足够的机会去取消，同时，对于那些存在潜在风险的操作也希望有机会去确认其意图。总之，人们希望能够从容地中止一个即将执行的操作。

从概念到产品

定义你的 App

App 定义陈述是一份对 app 的主要目的及其目标用户简洁而具体的阐述。

在你开发的早期就创建 app 的定义陈述，这将有助于你把一个想法和一堆功能变成一个用户梦寐以求的完整产品。在整个开发过程中，通过定义陈述来决定哪些潜在功能点和操作行为是否有合情合理。可以通过以下步骤来创建一个稳健的 app 定义陈述。

1. 列出所有你认为用户会喜欢的功能点

从这一步开始进行头脑风暴。你要尝试记下所有和你主要产品创意相关的任务。不要担心这个列表太长，稍后你会再做筛选。

假设你最初的想法是开发一个帮助用户采购食物的 app。当你进行头脑风暴时，你会想到一大堆用户可能会感兴趣的潜在特性。例如：

- 创建清单
- 获取菜谱
- 比价
- 附近的商店
- 食谱备注
- 获取和使用优惠券
- 浏览厨艺作品
- 发现不同的美食
- 查找食材成份

2. 定义你的目标用户

现在你需要弄清楚，是什么让你 app 的用户群体和其他 iOS 用户区分开来。在你主要想法的情境中，什么对他们最为重要？以采购食物为例，你可能需要问你的用户是否：

- 经常在家做饭，还是喜欢加工好的熟食

- 被认为是优惠券达人，还是认为不值得为优惠券花时间
- 热衷于搜寻特殊食材，还是很少在日常食材之外冒险
- 严格遵循菜谱，还是将菜谱用作灵感来源
- 少量多次，还是量多次少地进行采购
- 希望为不同目的维护几个正在进行的清单，还是仅仅记住几件东西在回家路上购买
- 执着于某个品牌，还是接受采购最方便的替代品
- 每次采购都倾向于购买类似的东西组合，还是按菜谱来买

经过思考这些问题，假设你确定了最符合你目标用户的三个特征：喜欢试验新菜谱、总是很忙、以及在不会太麻烦时尽量节俭。

3. 通过定义目标用户来筛选功能点

在确定了目标用户的特征之后，如果你得到了为数不多的几个特性，那就对了：优秀的 iOS app 都会精准聚焦于它们所要帮助用户完成的任务。

现在，回想你在第一步中列出的那个长长的潜在特性清单。即便是这些特性都有用，但也不是所有特性都会被你在第二步中定义的目标用户所喜欢。

当你不断在目标用户的情境中去验证特性清单，你就会领悟到你的 app 应该聚焦于三个主要特性：创建清单、获取使用优惠券和发现食谱。

现在，你可以开始定义你的 app 了，具体地概括 app 是做什么用的以及给谁用。对于这个食物采购 app，一个好的定义陈述可能是这样的：

「一个购物清单创建工具，为勤俭的美食家而生。」

4. 继续向前

在整个开发过程中，始终要用你的app 定义陈述去判断功能点、控件和用词是否妥当。例如：

当你考虑增加一个新功能时，问问自己，这对你 app 的主要目的和目标用户是不是不可或缺？如果不是，把它放在一边，它可能是构建另一个 app 的基础。例如，你已经确定你的用户喜欢探索烹饪，那么强调打包好的蛋糕和加工好的熟食可能不会被用户喜欢。

当你考虑 UI 的外观和行为时，问问自己，你的目标用户是喜欢简洁流畅的设计，还是一个更明显的主题风格？以用户对你 app 的期望为指引，例如使用 app 去完成一项重要任务，快速找到一个答案，探究内容详情，或者为了娱乐。尽管你的购物清单 app 需要易于理解而且立即可以上手，但你的用户可能会更喜欢一个展示着大量美味食材图片的界面主题。

当你在考量术语时，努力去迎合你的受众的专业认知。例如，即便你的受众不太可能由专业厨师组成，但你也应该明白，用户会希望看到配方和技术方面的专业术语。

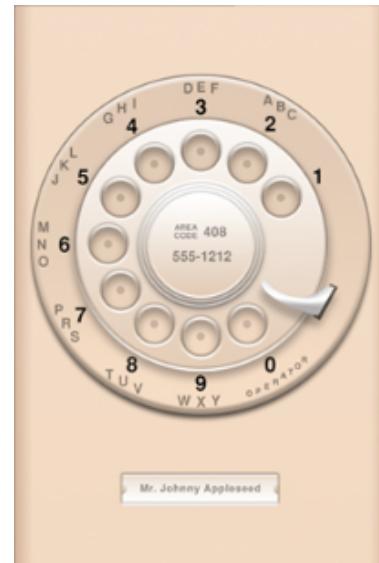
为任务量身定制

优秀的 iOS app 能够在易于使用和目的清晰之间平衡用户界面的自定义。要达到这个平衡，一定要在设计过程中尽早去考虑个性化定制。由于用户对品牌化、原创性和市场营销的关注都会影响设计决策，保持对差异化及用户体验的关注将会是一种挑战。

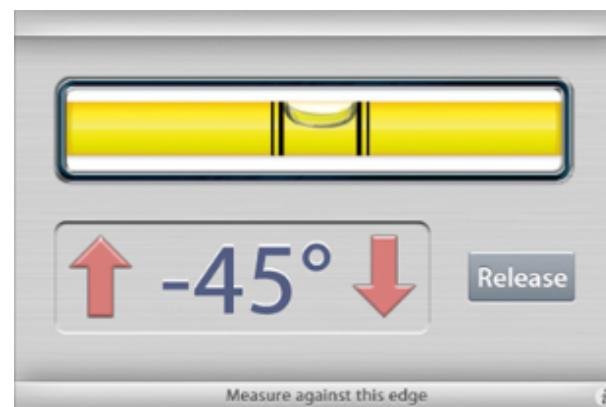
从考虑你的 app 所要完成的任务着手：用户在何种情况下会如何频繁地用到它们？

例如，假设有一个能拨打电话的 app。现在想象一下这个 app 没有使用键盘，而是呈现了一个漂亮而逼真的拨号盘。这个拨号盘被非常细致地渲染出来，用户很喜欢这样的质感。拨号方式非常真实，用户会乐于使用复古的拨号手势，还会听到那标志性的拨盘声音。

不过，对于那些经常需要拨打电话的用户，一开始的愉悦体验很快就会被挫折感所取代。因为使用旋转拨号盘远不如使用键盘有效率。对于一个设计来帮人们拨打电话的 app，这样漂亮的自定义界面其实反而是障碍。



另一方面，一个气泡水平仪的示例 app，它看上去就像是木匠使用的水平仪一样。人们知道如何使用现实中的的水平仪，所以他们很快就能上手这个 app。这个 app 或许可以不用气泡玻璃瓶来呈现信息，但这可能会让 app 不够直观。



观，甚至很难使用。在这个案例中，自定义的用户界面不止告诉用户如何使用这个 app，还让任务易于达成。

当你在思考定制化是增强还是削弱了 app 完成任务的能力时，请记住下面这些准则。

定制要行之有据。理想情况下，定制的 UI 会辅助用户想要完成的任务，并提升其体验。你需要尽可能让你 app 的任务去驱动你定制设计的决策。

尽可能避免增加用户的认知负担。用户熟悉标准 UI 元素的外观和行为，所以不需要停下来去思考如何使用它们。但当他们面对那些和标准 UI 外观和行为完全不一样的元素时，先前的经验毫无用处。除非你独特的元素可以让任务执行更容易，否则用户可能不会喜欢被迫去学习新的流程，而这些流程又不会在其他 app 中用到。

保持内部一致性。你的 UI 自定义得越多，在你的 app 中这些自定义元素的外观和行为保持一致就更为重要。如果用户需要花时间去学习如何使用你所创建的不熟悉控件，那他们会希望可以依赖这些经验混迹整个应用。

始终为内容服务。由于这些标准元素是如此常见，所以它们不会和内容有任何的不和谐。当你在自定义界面时，要确保其不会盖过人们所关注的内容。例如，如果你的 app 可以让人看视频，你可能需要选择设计一个自定义的播放控件。但是无论是使用自定义还是标准的播放控件，更重要的是控件是否在视频开始后隐藏并轻点时重新浮现。

重新设计标准控件前要三思。如果你正在计划重新设计一个标准控件，要确保你的新设计提供了和标准控件一样多的信息。例如，如果你设计的开关控件不能表现出两个相反的状态，用户可能不会意识到这是一个两态控件。

务必对自定义的 UI 元素进行充分的用户测试。在测试期间，仔细观察用户是否能够准确了解你界面元素的作用，以及他们是否可以轻易地与之交互。例如，你设计了一个点击区域小于 44*44 点的控件，那用户在点击时就会碰到麻烦。或者，如果你创建了一个会对轻扫和轻点产生不同响应的视图，那么确保这个视图所提供的功能是值得人们和它互动时耗费的注意力的。

原型&迭代

在你真正投入开发资源去完成你的设计前，创建一个原型去做用户测试会是很好的做法。即使你只能找到几个同事来测试这些原型，你仍然会从他们对 app 功能的新鲜视角和体验中获益匪浅。

在你设计的最早期，你可以使用纸面原型或者线框图来表现最主要的视图和控件，并绘制好不同界面间的流程。你可以从测试线框图中获得不少有用的反馈，但线框图的粗陋也可能会误导测试用户。这是用户很难想象当线框图被落实到真正的内容时，这个 app 的体验将是什么样子。

如果你能将组建一个可以在设备上运行的原型，你将会得到更有价值的反馈。当人们能在一台设备上和你的原型交互，他们会发现 app 中那些不符合期望或者用户体验过于复杂的地方。

创建一个可信的原型最为简单的办法就是使用一个基于**故事板 (Storyboard-based)** 的 Xcode 模板去创建一个基本 app，并填充一些合适的占位内容。（故事板文件能够获取整个 app 的界面，包括不同界面之间的切换。）然后，将这个原型安装到一台设备上，以便你的测试用户拥有一个尽可能真实的体验。

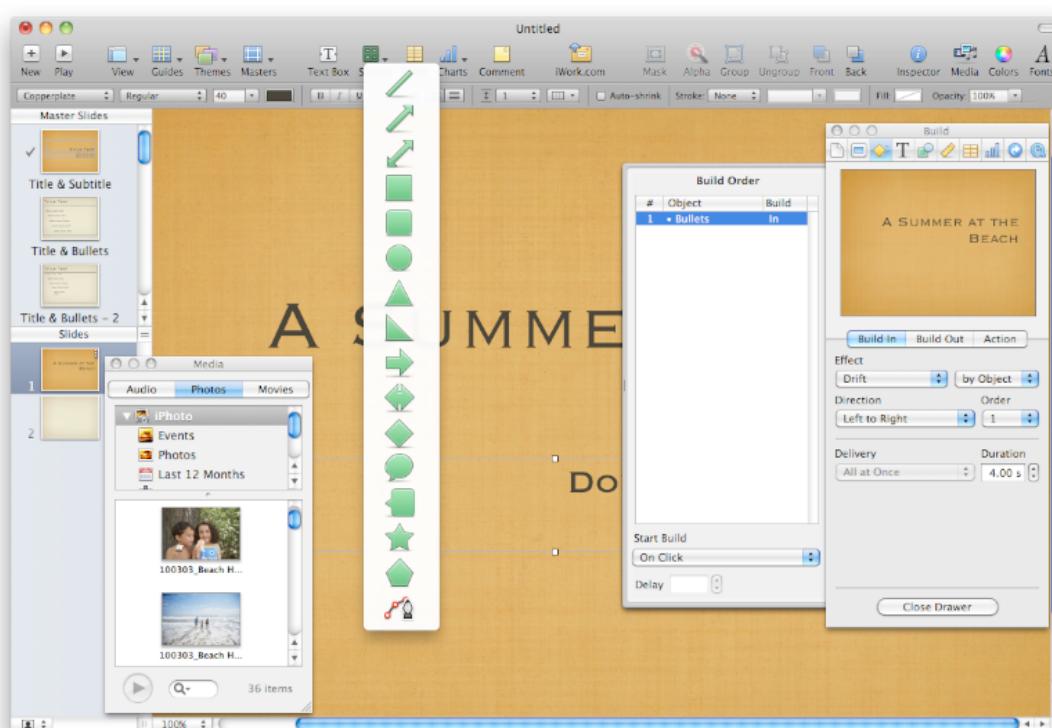
你不需要在你的原型 app 中放太多内容或者支持每一种控件，但你需要提供足够的情境去营造一种真实的体验。要努力在典型的用户体验和非常规边缘案例之间达成平衡。例如，如果你的 app 将要处理很长的列表项，那你不应该创建一个只能显示一两行条目的原型。对于用户交互性测试，只要测试者轻点屏幕中的某个区域能进入下一个逻辑视图或者操作一个主要任务，他们就可以提供建设性的反馈了。

如果基于 Xcode 的 app 模板创建原型，你可以免费获得很多功能，而且根据用户反馈做出设计调整也会相对容易。在你最终确定设计并提交开发资源之前的这段时间内，你可以对你的原型做几次测试迭代。如需了解更多关于 Xcode 的知识，请参阅《Xcode Overview》。

案例研究：从桌面到 iOS

iPad 中的 Keynote

桌面版的 Keynote 是一个强大且灵活的 app，可以用来创造世界级的幻灯演示。人们喜欢 Keynote，因为它将易用性和无数精确的细节很好地融合在一起，例如动画和文本属性。



iPad 版的 Keynote 继承了桌面版 Keynote 的精髓，通过创造如下这些体验，使它在 iPad 让人觉得宾至如归：

- 聚焦于用户内容
- 减少复杂度，但不减弱性能
- 提供强大而令人愉悦的快捷方式
- 微调桌面上常见的特性
- 通过动人的动画提供反馈和交流

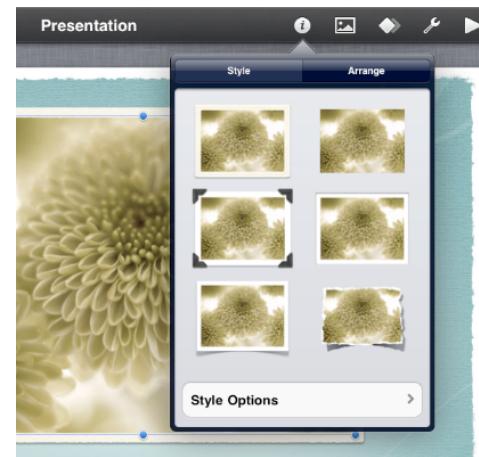
Keynote 用户马上就能明白如何在 iPad 上使用这个 app，因为它使用了原生的 iPad 范式并传达出符合预期的功能。新用户也能轻松学会如何使用 iPad 版的 Keynote，因为他们可以用一种简单自然的方式去直接操控内容。

从桌面版进化到 iPad 版 Keynote，经过了大量由浅入深的修改和重新设计。以下是最明显的一些变化：

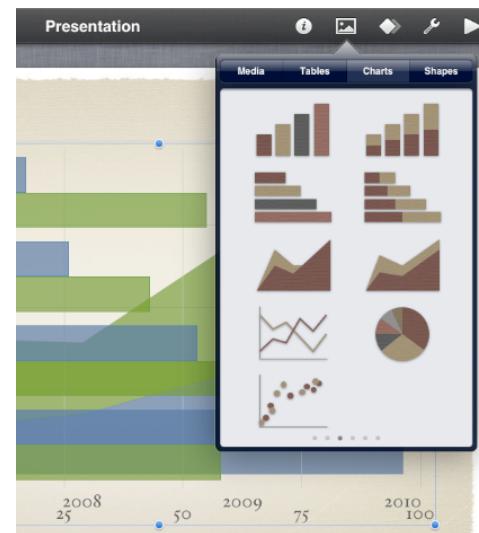
简洁流畅的工具栏。在工具栏上仅有 7 个项目，但这为用户提供了一致的入口，用来访问创建内容所需的所有功能和工具。



简洁而有序的检查器，随时响应用户需求。iPad 版 Keynote 的检查器会根据人们选中的修改对象而自动包含相应的工具和属性。通常，人们可以在检查器的第一个视图中进行所需的所有改动。如果他们需要修改那些很少被改动的属性，则可以深入到检查器的其他视图中寻找。



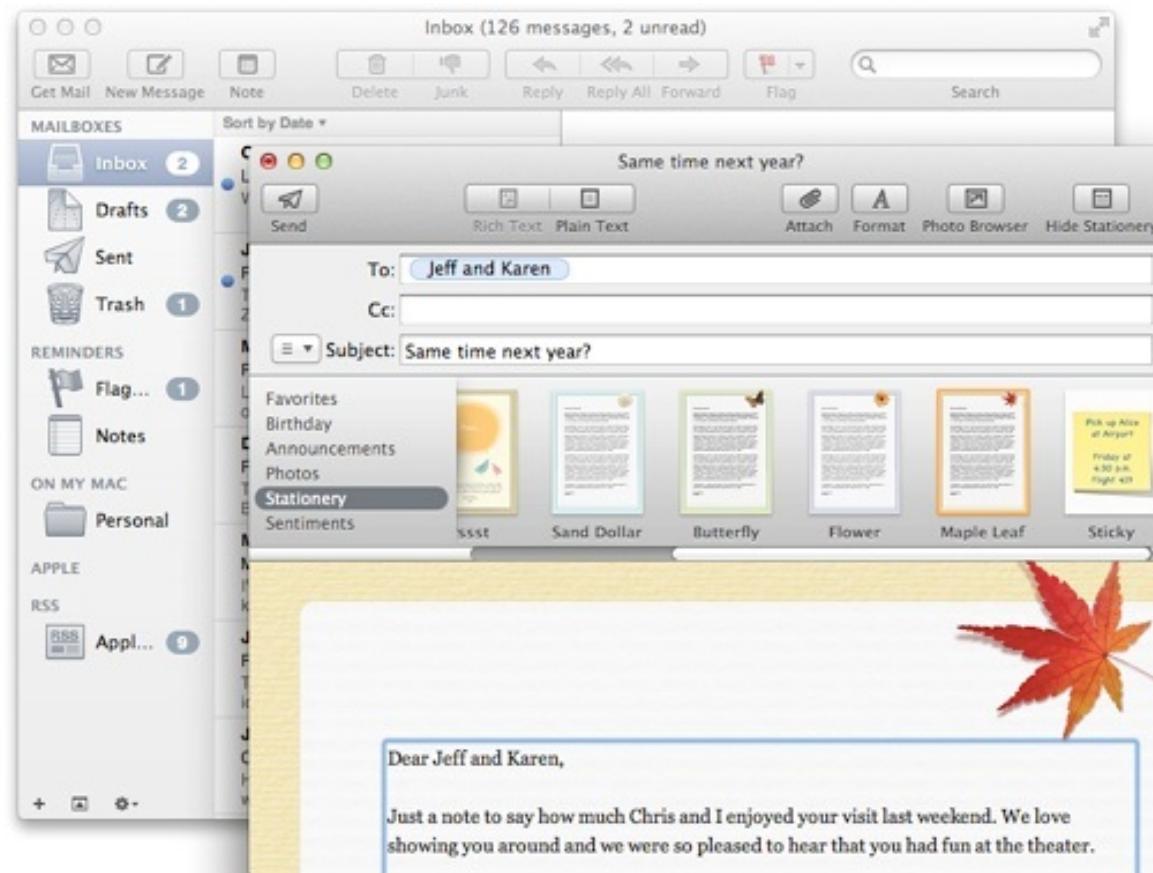
大量预设的样式精选。利用预设样式，人们可以很轻松地改变诸如图表和表格等对象的外观和感觉。除配色方案外，每一个模板还包括了表格标题和分栏符号等预设属性，这些都被设计来配合整个主题。



直接对内容进行操控，并赋予有意义的动画。在 iPad 版 Keynote 中，用户可以拖拽一张幻灯片到一个新的位置，扭转手指可旋转对象，或是轻触一张图片来选中。随时响应的动画效果增强了直接的操控感受。例如，在用户移动一个幻灯片时，它会轻轻抖动，当用户把它放到新位置时，周围的幻灯片会如涟漪般散开以为它腾出空间。

iPhone 中的「邮件」

「邮件」是 OS X 中最为瞩目、最常用和最受赞赏的 app 之一。同时，它也是一个非常强大的程序，能让用户新建、接收、处理并存储电子邮件，追踪行动项目和事件，以及新建笔记和邀请函。桌面版的「邮件」用几个窗口就提供了这些强大的功能。



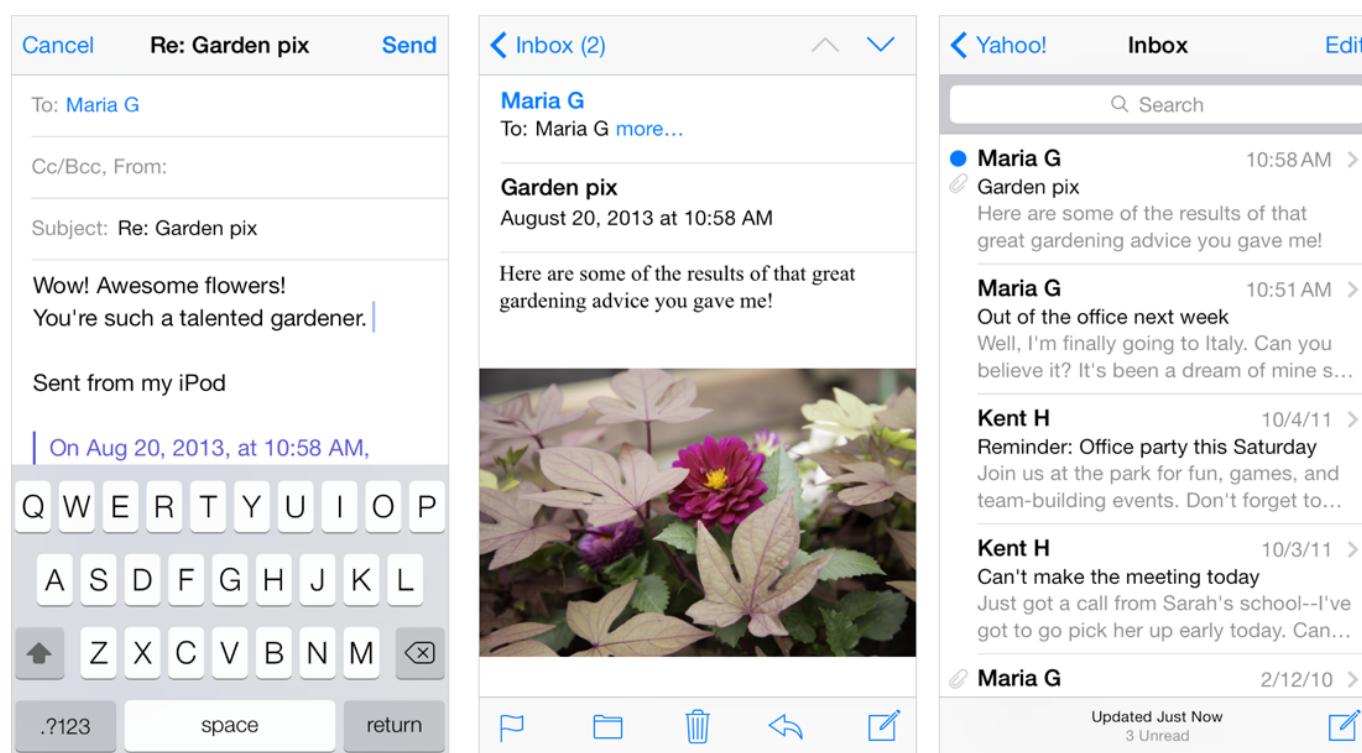
iPhone 上的「邮件」聚焦于桌面版的核心功能，帮助用户接收、创建、发送和整理他们的邮件。iPhone 版「邮件」在为移动体验量身打造的 UI 中实现了大量功能，其中包括：

- 简洁流畅、将用户的内容置于中心的外观
- 为满足不同任务需求而设计的不同视图
- 直观且易于理解的信息架构
- 在用户需要时，提供强大的编辑和组织工具
- 精妙但富有表现力的动画，与操作行为互动并提供反馈

必须要意识到，iPhone 上的「邮件」不一定是一个比桌面版「邮件」更好的 app，相反，它是为移动用户重新设计的「邮件」。通过聚焦于桌面版的一部分特性，并用一个简洁而富有吸引力的 UI 去呈现，iPhone 版「邮件」在人们移动时为其提供了最为核心的「邮件」体验。

为了让「邮件」体验适应移动情境，iPhone 版「邮件」在几个关键地方对 UI 进行了革新：

独特而高度聚焦的页面。每个页面展示「邮件」体验的一个方面：账户列表，邮箱列表，邮件列表，邮件视图和撰写视图。在一个页面中，人们可以滚动以阅读全部内容。



简洁、符合预期的导航。轻点每个页面，用户可以从概要（账户列表）进入具体内容（一封邮件）。每个页面都展示一个可以告诉用户位置的标题，和一个让他们可以很容易原路返回的后退按钮。

简明、可点击的控件，在需要时保持可用。由于撰写邮件和检查新邮件在任何情况下对人们来说都可能是最想要进行的基本操作，iPhone 版「邮件」让它们在很多页面都保持可见。当用户在阅读一封邮件时，保持诸如回复、移动和删除等功能可用，因为用户会对其进行操作。

为不同任务提供不同类型的反馈。当人们删除一封信件时，它会有一个扔进垃圾桶图标的动画效果。当人们发送一封邮件时，他们能看到进度；当发送完成，还能听到一个特别的音效。透过邮件列表中工具栏上精致的文字，用户一眼就能看到邮箱上次更新的时间。

iOS 中的网页内容

iOS 版 Safari 在 iOS 设备上提供了卓越的移动网页浏览体验。用户喜欢它清晰的文字，明锐的图像，以及通过旋转设备、双指开合及轻触屏幕以调整视图的能力。

标准的网站在 iOS 设备上会显示良好，尤其是那些可以检测设备且没有使用插件的网站，在 iPhone 和 iPad 上看起来都非常棒，即便有细小的改动。

此外，最为成功的网站通常：

- 如果页面宽度需要匹配设备宽度，为 iOS 设备设置合适的显示范围
- 避免使用 CSS 绝对定位，这样用户在缩放页面时内容不会错位
- 使用基于触控的 UI，而非依赖基于指针的交互

有时候，可以在其他方面进行适当的修改。例如，Web app 通常会设置合适宽度的显示范围，并在 iOS 中隐藏 Safari 的 UI。如需了解更多关于如何做出这些修改的信息，请参阅《Safari Web Content Guide》中「Configuring the Viewport」和「Configuring Web Applications」两节。

网站还可以通过其他方式将桌面 web 体验迁移到 iOS 中：

适配 iOS 版 Safari 中的键盘。当键盘和表单助手出现时，iPhone 版 Safari 会在 URL 地址栏下方、键盘和表单助手上方的区域里显示你的网页。

适配 iOS 版 Safari 中的弹出菜单控件。在桌面版的 Safari 中，一个包含大量项目的弹出菜单，会像在一个 OS X 应用中那样呈现，也就是在菜单展开后显示所有项目，在必要时甚至还拓展到窗口边缘以外。而在 iOS 版 Safari 中，弹出菜单会使用原生组件呈现，这样可以提供更好的用户体验。例如，在 iPhone 中，弹出菜单是以选择器的方式呈现，用户可以从一系列选项中选择一个。（如需了解更多关于选择器的信息，请参阅「[选择器](#)」（第 157 页）。）

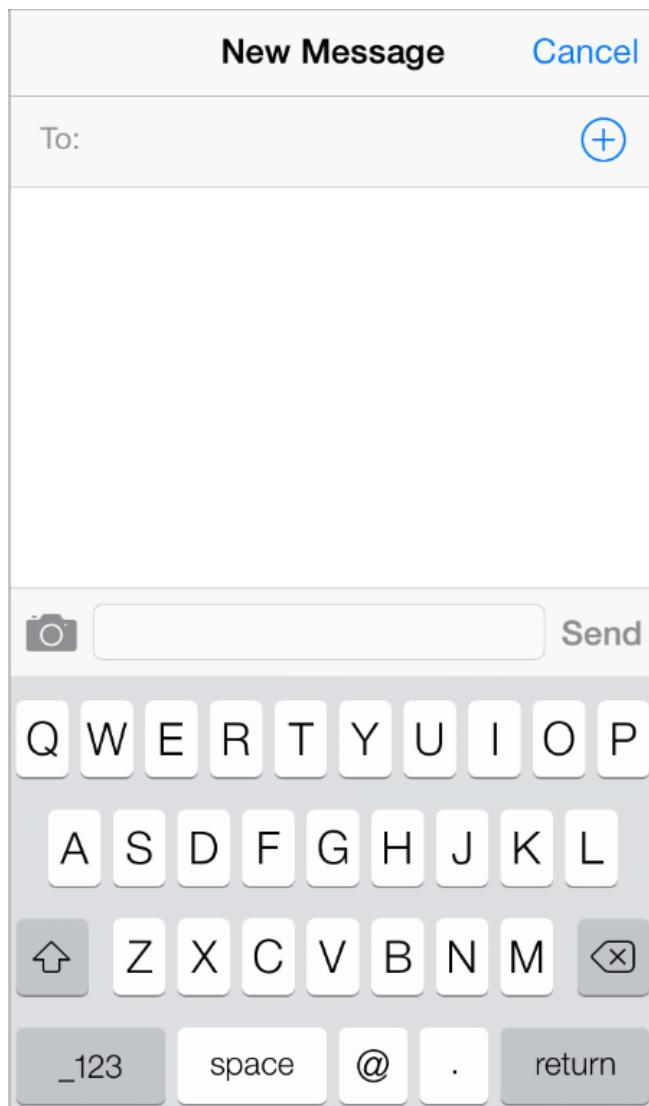
在 iPhone 5 上运行

你不需要重新设计你的 app 才能让其在 iPhone 5 上显示良好。通过展现更多已有的界面，大部分的 app 看上去都很简洁优雅；其他的可能需要拉伸内容或背景区域。只有一些 app——例如游戏或展示大量自定义界面的 app——才需要一些额外的工作才能让它们显示良好。

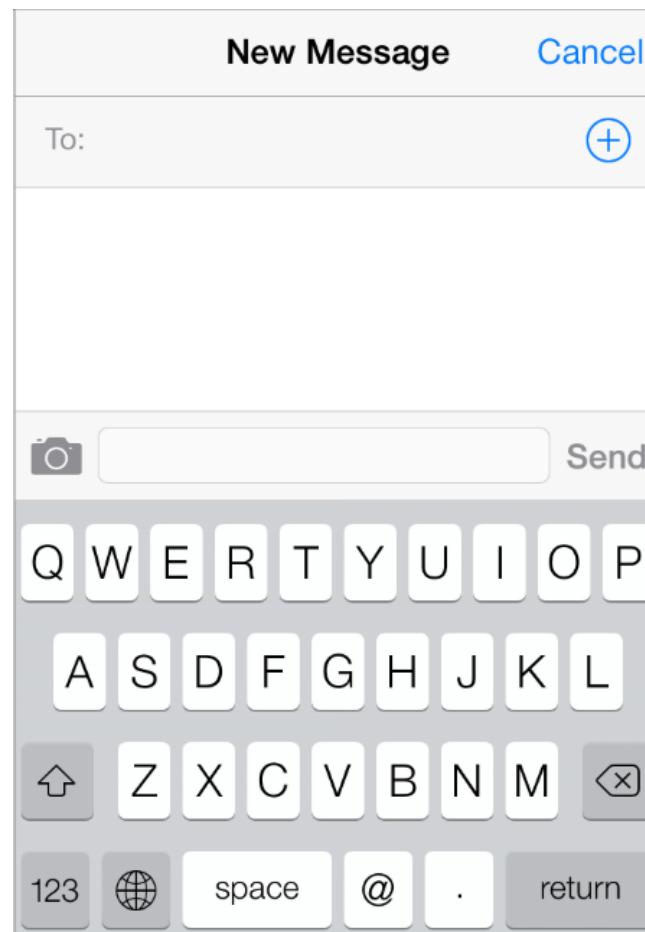
注意：所有运行在 iPhone 5 中的 app 必须包含正确尺寸的启动画面。如需了解如何创建 iOS 设备的启动画面，请参阅「[启动画面](#)」（第 180 页）。

iPhone 5 的屏幕比其他 iPhone 和 iPod touch 设备屏幕高了 176 像素。在竖屏方向中，多出来的高度大约等于标准表格视图中两行的高度。例如，运行在 iPhone 5 中的「信息」有着比 iPhone 4S 中的「信息」更高的对话区域。

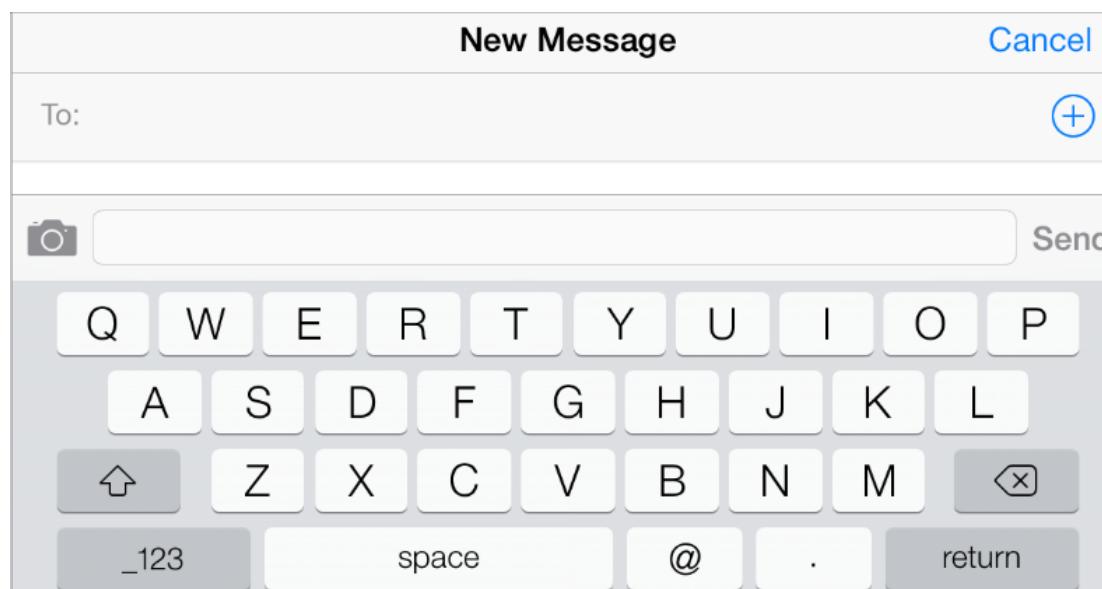
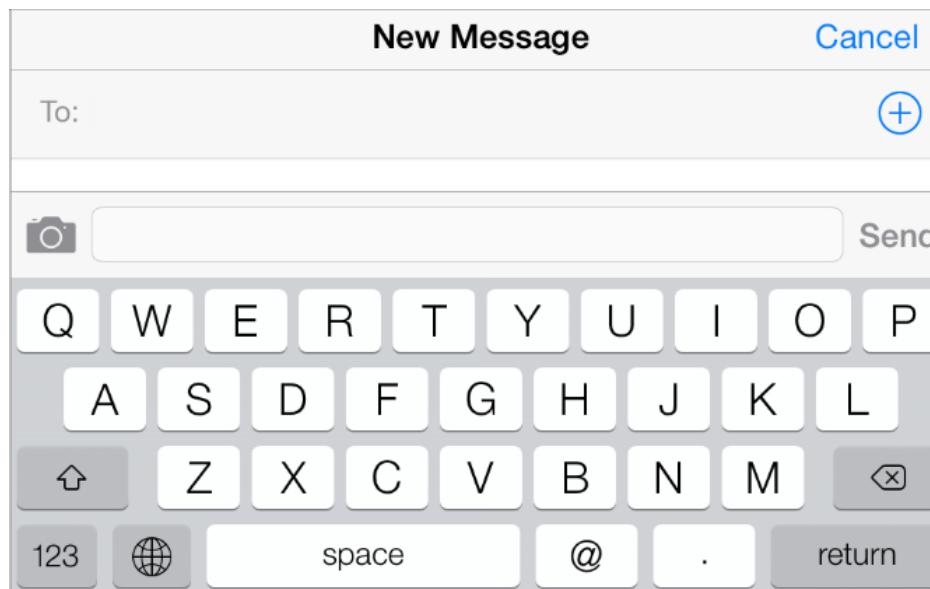
iPhone 5 中的「信息」



iPhone 4S 中的「信息」



当然，当设备处于横屏模式时，多出来的 176 像素高度会变为额外的侧面空间。例如，在 iPhone 5 中的「信息」（如下所示是 iPhone 4S 中的「信息」）中，界面伸展填满了多出来的宽度。



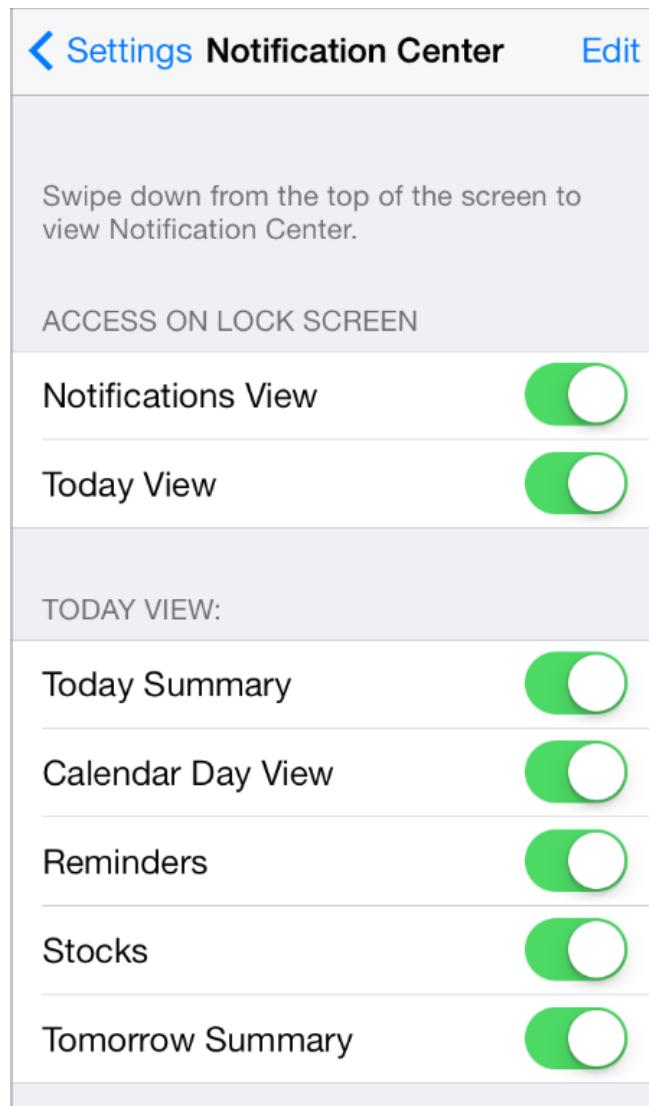
如上显示的「信息」页面传达了一个关键点：为适应 iPhone 5 显示而做的少量界面调整在任何方面都不会改变 app 的功能性。用户体验的一致性——在本节中提到的所有 app 中尤其明显——遵循 iPhone 5 的一个关键设计原则：更大的设备屏幕能让用户看到更多他们所关心的内容；它并不提供获得更多 app 功能的机会。

由于你不会改变 app 的功能，你通常只需要做一点事情就能让你的 app 在 iPhone 5 上看上去很棒。如果你使用 Auto Layout 去涉及你的 app 界面，你甚至可能需要做得更少。下面这些准则可以帮助你决定改变哪些界面对你的 app 最适合。

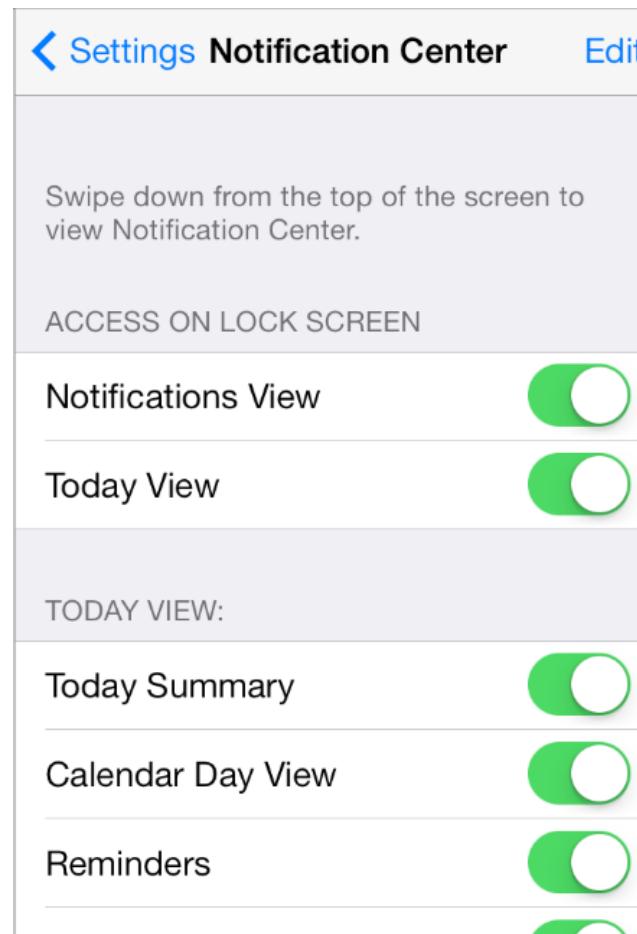
注意：如果你的 app 不作任何改变，它会以兼容模式在 iPhone 5 上运行。当一个 app 以兼容模式运行，iOS 会通过在上下两侧增加黑边来让 app 界面自动居中显示。

允许更多内容自动展现。如果你的部分界面隐藏在屏幕底部之下——需要用户滚动——你不用做任何改变去增加 iPhone 5 中的显示高度，来展现你的更多界面。例如，在「设置」中用户可以在 iPhone 5 上看到比在 iPhone 4S 中更多的表格行。

iPhone 5 中的「设置」

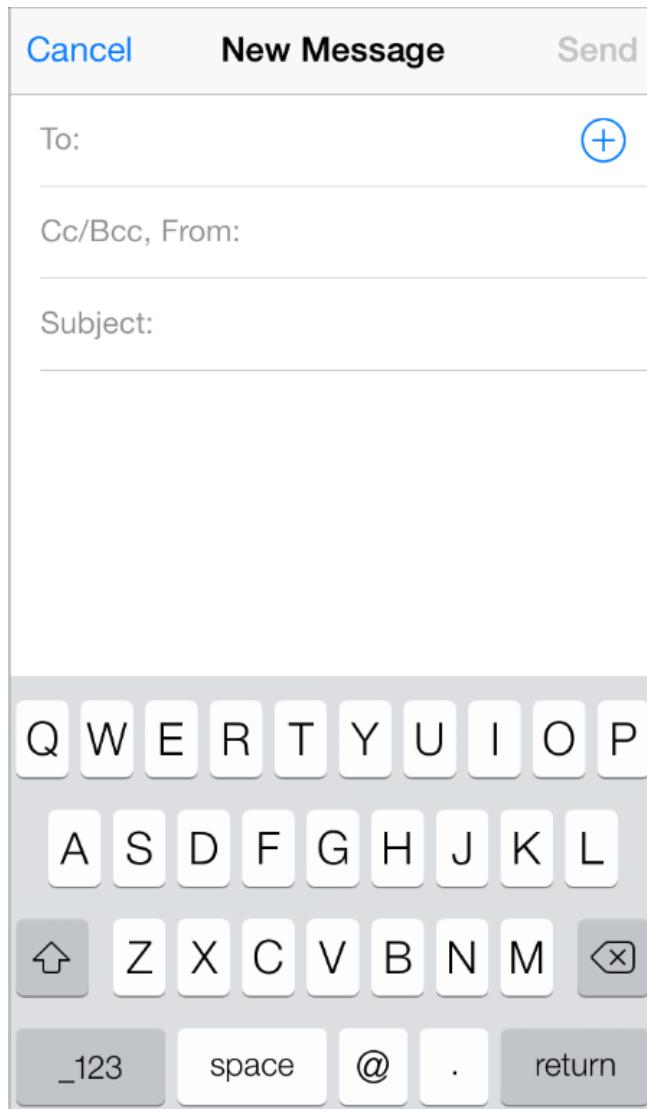


iPhone 4S 中的「设置」

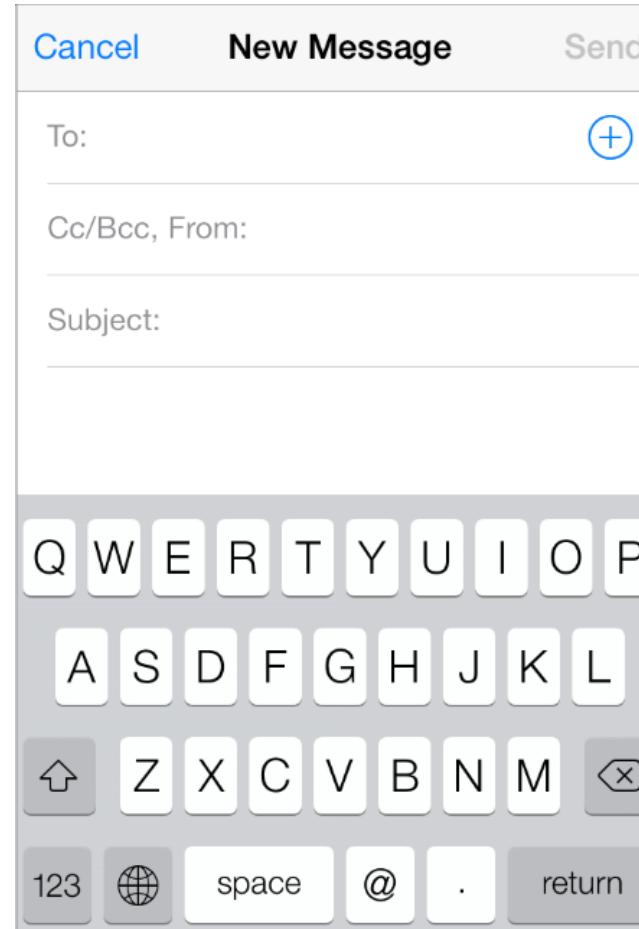


伸展内容区域。如果你在一个视图中显示内容，考虑垂直拓展视图以看到更多内容或者在视图中插入更多留白。例如，用户在 iPhone 5 中的「邮件」中有着比 iPhone 4S 更大的信息正文区域。

iPhone 5 中的「邮件」



iPhone 4S 中的「邮件」



伸展内容区域之间的背景区域。你可以通过拓宽内容视图之间的垂直空间来让你的布局看起来更舒适。例如，iPhone 5 中的「天气」在页面上部显示了比其在 iPhone 4S 上更大的背景。

iPhone 5 中的「天气」

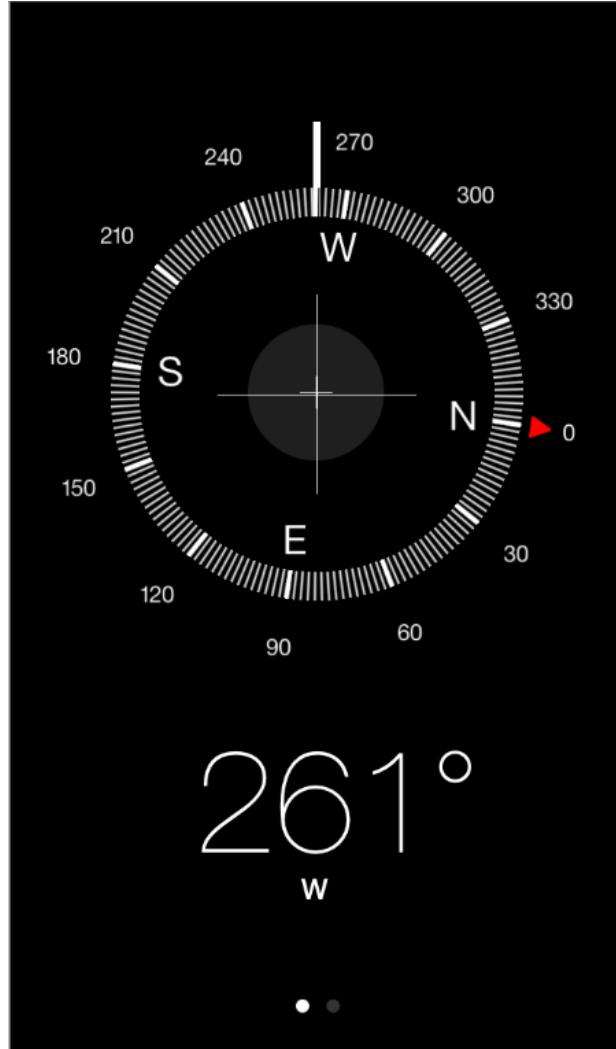


iPhone 4S 中的「天气」

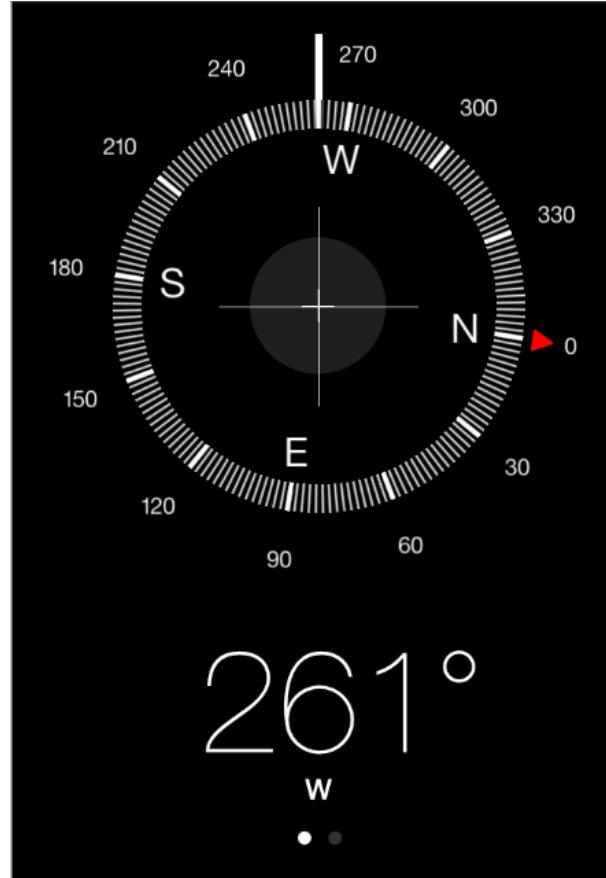


如果需要，重新居中对齐主要视觉要素。在调整背景或视图以适配更多空间后，你可能想要确保你界面中最重要的元素仍然是合适的居中对齐。这样做的一种方式是，首先伸展靠近页面顶部的区域再调整其他元素的居中对齐。例如，iPhone 5 中的「指南针」通过保持元素在拉伸背景上的居中对齐让用户始终聚焦于指南针和头部。

iPhone 5 中的「指南针」



iPhone 4S 中的「指南针」



通常来说，**避免增大控件尺寸**。如果你在 app 中使用了最小点击尺寸——即，最小 44×44 点——你不需要再为 iPhone 5 调整它们。但是，你可以选择在现有控件之间增加一点空间。

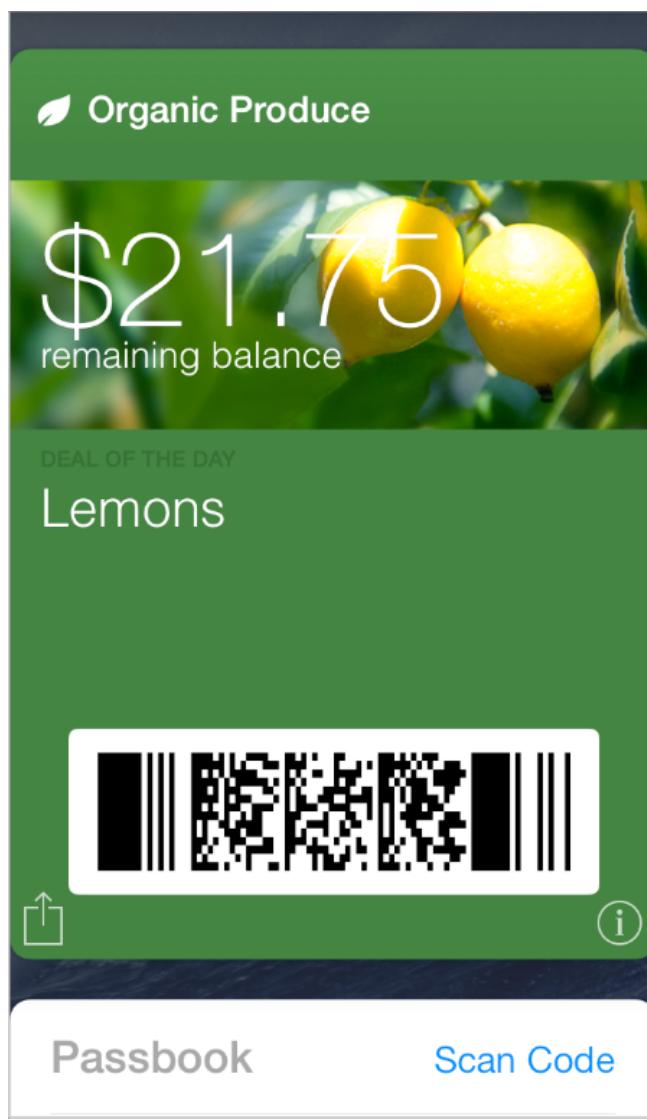
不要使用多出来的空间去显示额外的条栏或横幅。人们期望在 iPhone 5 上看到更多内容。为避免你的用户失望，要抵制使用额外的垂直控件去显示自定义按钮条栏或横幅的诱惑。

iOS 技术

- 「Passbook」 (第 81 页)
- 「多任务处理」 (第 83 页)
- 「路线导航」 (第 85 页)
- 「社交媒体」 (第 87 页)
- 「iCloud」 (第 89 页)
- 「App 内购买」 (第 91 页)
- 「Game Center」 (第 93 页)
- 「通知中心」 (第 95 页)
- 「iAd 富媒体广告」 (第 98 页)
- 「AirPrint」 (第 101 页)
- 「位置服务」 (第 103 页)
- 「快速预览」 (第 105 页)
- 「声音」 (第 106 页)
- 「VoiceOver」 (第 113 页)
- 「编辑菜单」 (第 114 页)
- 「撤销和重做」 (第 116 页)
- 「键盘和输入视图」 (第 117 页)

Passbook

Passbook 可以帮助用户查看和管理票券，并以数字形式呈现诸如登机牌，优惠券，会员卡和票据等物理实体。在你的 app 中，你可以创建发放给用户的票券并在有变化时更新它。



Pass Kit 框架让使用自定义内容制作一个票券变得容易，并支持在将其放置在用户的票券夹中时读取它。（如需了解更多 Passbook 的关键点和如何在你的 app 中使用 Pass Kit API，请参阅《Passbook Programming Guide》。）以下这些办法能够帮你创建一个用户喜爱且乐于使用的票券：

尽可能不要简单复制现有的物理票券。Passbook 有一套既定的设计美学，符合这套美学的票券往往看上去会很好。放弃对物理实体外观的临摹，趁此机会追随 Passbook 的形式与功能去设计一个干净简洁的票券。

有选择地在票券正面放置信息。用户希望看一眼票券就能立刻找到他们所需要的信息，因此票据正面应当是整洁且易于阅读的。如果你认为有用户可能会需要的额外信息，相比挤在票券正面，更好的办法是将其放到背面。

通常情况下，避免使用一片空白作为背景。票券的背景是一个生动干净的颜色，或是一张颜色丰富活力的图片时，这会看起来很棒。设计背景时，要确认其没有对内容的可读性产生干扰。

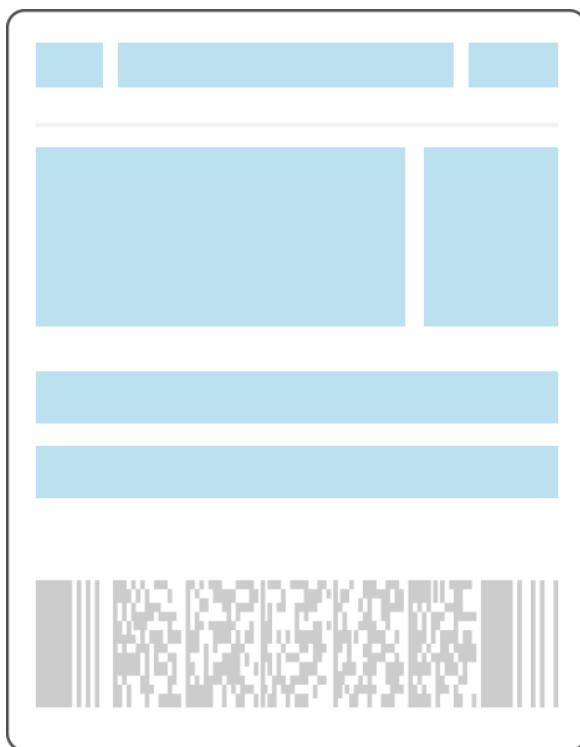
使用标识文本框填写你的公司名字。在所有票券中，标识文本框中的文字都会用同一个字体。为了避免和用户票券夹中的其他票券冲突，比起使用一个自定义字体，在标识文本框中输入文字是更推荐的方式。

注意：最好在你的票券中使用合适的票券文本框来输入文本，并避免在文本中插入图像或使用自定义字体。使用文本框对你有两个很重要的好处：它允许 VoiceOver 用户获取票券信息并能让你的票券有着一致的外观。

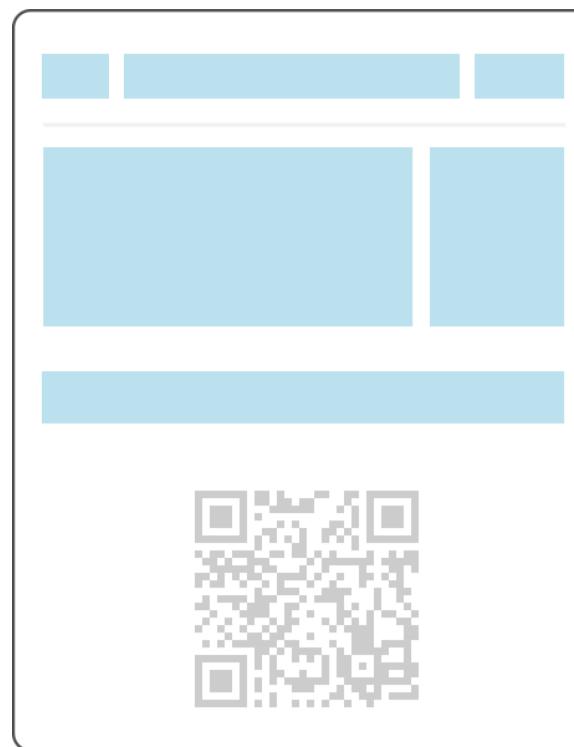
使用白色的公司标识。标识图像会被放置到票券的左上角，紧挨着公司名字。使用一个不包含文本、白色版本的标识可以呈现出最佳效果。如果你想让标识和渲染的标识文字的雕刻效果相衬，添加一个 1 像素垂直偏移、1 像素大小、35% 不透明度的黑色阴影。

尽可能使用矩形条码。由于票券的布局设计，矩形条码（例如 PDF417）看上去会比正方形条码更好。如右下图所示，正方形条码会造成两边有很大空隙，又垂直挤占了上下两边的区域。

矩形条码能很好适应布局



正方形条码会挤占其他空间

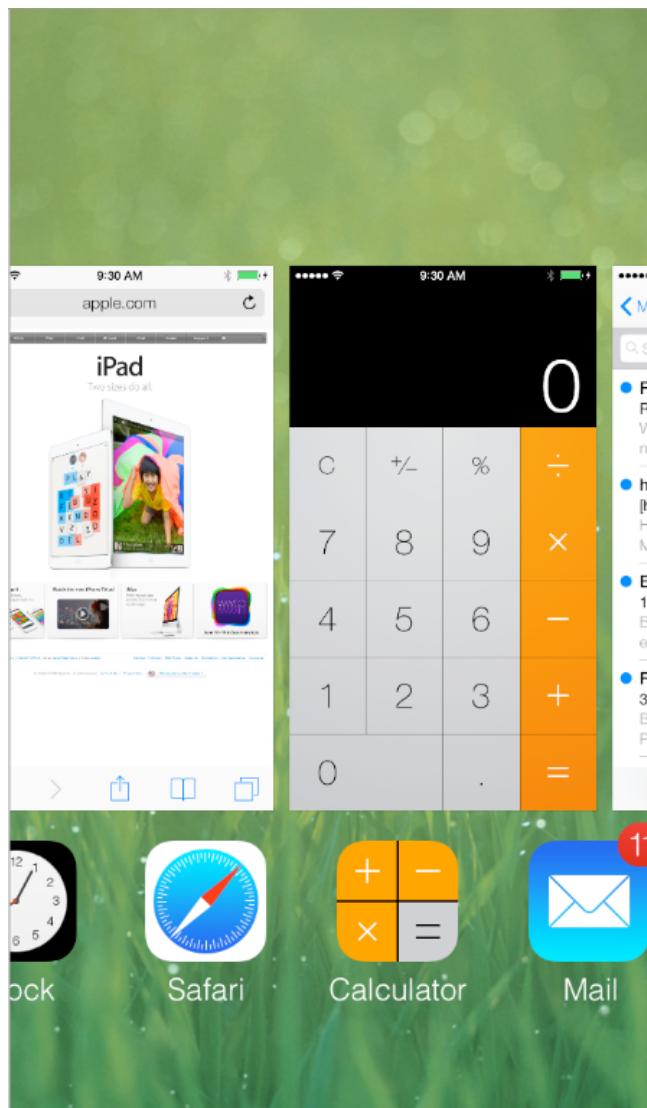


优化图片性能。由于用户会经常通过电子邮件或 Safari 接收票券，所以尽可能很快地下载票券很重要。为了提升用户体验，你可以用最小的图片文件来实现所需的视觉效果。

适时更新票券，以提高其效用。尽管票券代表着某个通常不会发生变化的物理实体，但你的数字票券可以通过反映真实世界的事件来提供更好的体验。例如，当航班延误时，更新你的航班登机牌以便用户在检查票券时总能获取最新信息。

多任务处理

多任务处理允许用户在最近使用过的 app 之间快速切换。



为了提供这样的体验，在用户从一个程序中切换离开时，多任务处理会让其在后台进入暂停状态。当用户切换回来时，由于无需重新加载界面，因此 app 能够快速恢复。人们会用**多任务处理界面**（如下所示）来选择最近使用的 app。

API 备注：如需了解如何在代码中实现多任务处理，请参阅「App States and Multitasking」。

纷繁的多任务环境能让设备上的其他应用实现和谐相处。从一个很高的层面来说，这意味着所有的 app 应当：

- 优雅地处理从其他应用过来时的中断和音效
- 快速、平滑地停止和重启（即进入和退出后台）

- 负责任地处理后台行为

以下这些特定的准则会有助于在多任务环境中成功运行你的 app。

做好随时中断和恢复的准备。多任务处理增加了后台程序打断你的 app 的可能。其他的特性，例如广告和快捷切换的存在，可能会导致更频繁的打断。越快越准确地保存你程序当前的状态，用户就能更快地从他们中断的地方重启。为了给用户一个无缝的重启体验，使用 UIKit 的状态保存和功能恢复（如需了解更多，请参阅「State Preservation and Restoration」）。

确保你的界面能支持双倍高度的状态栏。双倍高度状态栏会在正在进行的通话、录音和放在底座上时出现。没有处理这个双倍栏高的应用可能会导致布局错乱。例如，界面可以被压缩或者被盖住。由于在多任务环境中要处理好双栏高度的状态栏，这尤为重要，因为越来越多的 app 会导致其出现。

做好随时停止那些需要用户关注或参与的进程的准备。例如，如果你的应用是一个游戏或者多媒体程序，确保用户从你的程序切换离开时不会遗漏任何内容或事件。当用户切换回来时，他们想像从未离开过一样继续体验。

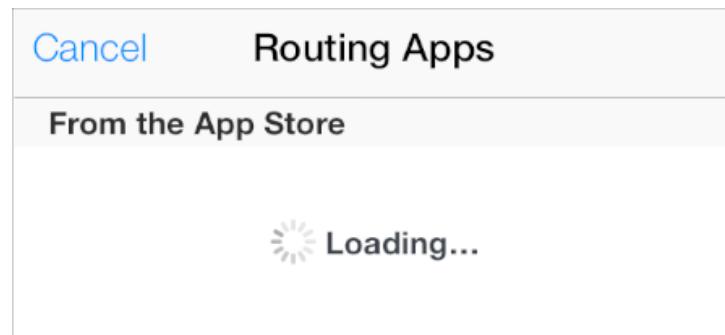
确保恰当地处理程序声音。多任务处理使得在你的程序运行时很有可能同时运行着其他程序。这也使得你的程序声音要能暂停和恢复以应对随时可能发生的中断。关于帮助你确保程序声音符合用户预期并和设备上的其他声音和谐共处的指南，查看「[声音](#)」（第 107 页）。

恰当地使用本地通知。无论 app 在后台暂停、运行还是根本没有运行，程序都要能在特定时间发送本地通知。为了获得最佳用户体验，避免给用户呈现过多通知，并遵循在「[通知中心](#)」（第 96 页）中创建通知内容的准则。

适时地在后台完成用户发起的任务。在用户触发一个任务后，他们常常希望即便在离开你的程序后，程序也保持运行直至完成任务。如果你的 app 执行一个用户任务到一半，不需要用户做出额外操作，你也应当在后台将其暂停前完成它。

路线导航

在 iOS 6 及更高版本中，当用户想要获取到一条路线的交通信息时，「地图」会展示一列导航应用（包括安装在设备上和 App Store 中的 app）。



路线 app 能提供当前选中路线的交通信息。用户期望路线程序能快捷易用以及——最为重要的——准确。遵循下面这些准则，有助于你向用户提供的他们可以信任的交通信息和深受赞赏的体验。

重要：「地图」会给用户的线路提供驾车和步行方向。路线程序提供交通信息，其重点是使用替代交通工具，如巴士、火车、地铁、渡轮、自行车、徒步和班车等一步一步的方向。如果你的程序不能提供用户指定路线的交通信息，就不要将其认证为一个路线程序。

实现 app 所承诺的功能。当用户在交通列表中看到你的 app，便会假定你能帮他们到达目的地。但如果你并不能为选中路线提供信息，或者不包括其显示的交通类型信息，用户不会再用第二次。准确的表述应用的功能是很重要的，否则可以认为你的 app 是在故意误导用户。

这里有两个可以在路线程序中给予用户信心的方法：

- 尽可能精确地定义你所支持的地理区域。例如，如果你的 app 能让用户获取巴黎的巴士路线，那你所支持的区域应该是巴黎，而不是法兰西岛或者法国。
- 明确你所支持的交通类型。例如，如果你专注于地铁信息，那这意味着你不需要提供所有轨道交通工具的信息。

注意：尽管准确告知你所支持的区域意味着你的 app 在交通列表中会减少出现，但这会让用户对你更加信任。

流畅易用的 UI。操作简便对路线程序尤为重要，因为用户常常会在极端情况下使用它们，在明亮或昏暗的火车上，在颠簸的骑行中，在行路匆匆时。确保在任何亮度下你的文本都易于阅读，甚至在颠簸骑行时按钮都能准确点触。

关注路线。尽管辅助信息会很有用，但你的 app 应当着眼于给予用户一步一步的方向，以便他们可以按图索骥。特别是，你想让用户知道他们在哪一步，下一步怎么去。你可以提供一些额外数据，例如时刻表和系统地图，但不要让这些数据比交通信息更显眼。

给路线中的每一步都提供信息。在你的 app 中，用户不应该感到被抛弃。即便你准确告知了你所支持的区域，你也不能假设用户已经处于某条路线中的起点或终点，或者终点和他们的目的地是同一个低点。为了应对这种情况，首先需要检查路线起点和终点的距离。如果距离足够短，提供从当前位置到第一个中转站，再从最后一个中转站到终点的步行路线。如果步行不是一个合适的选择，试着向用户提供其他选项。如果需要，你可以给用户一种打开「地图」的途径以获得部分路线的步行或驾车路线。

当用户从「地图」切换到你的程序时，不要再次询问相关信息。如果用户从「地图」进入，你已经知道他们感兴趣路线的起点和终点，所以在 app 打开时你可以呈现相应的交通信息。如果用户从主屏幕启动 app，为他们提供一个进入路线细节的简便方式。

图形化、文字化地显示交通信息。地图视图让用户从一个更大的环境脉络看到整个路线，一个列表的步骤有助于用户关注他们到达目的地所必须要采取的行动。当你同时支持它们时，让用户能轻松在两者之间切换是最好不过的了。

注意：不考虑格式，对于用户的路线，很重要的是显示相同的交通信息。例如，如果一个路线包含五个步骤，那路线的地图和列表视图都应该描述同样的五个步骤。

当你的 app 从交通列表中被打开后，在其启动时就应在地图视图中展现一条完整路线——如果合适，应包含每个中转站之间的步行路径。地图视图应该让用户对整个旅程中各个步骤有一个概览，并向其展示他们的路线是如何和周边地理环境相协调的。

用附加信息丰富地图视图。用户期望你 app 中地图的表现与他们用过的地图类似。此了让用户放大和平移之外，你应当显示一些点，以呈现用户当前位置、目的地、中转站和沿途的兴趣点（POI）。务必要避免只显示一个点，因为如果没有附加信息会很难告诉用户它代表什么。如需了解更多在 app 中使用地图视图的信息，请参阅「[地图视图](#)」（第 136 页）。

在地图视图中尽可能整合地图数据，例如地铁路线图。将这些静态图像覆盖在地图视图上是一个不错的办法，这样可以方便用户了解他们的路线和当前的位置是如何和更大范围的交通系统相联系的。

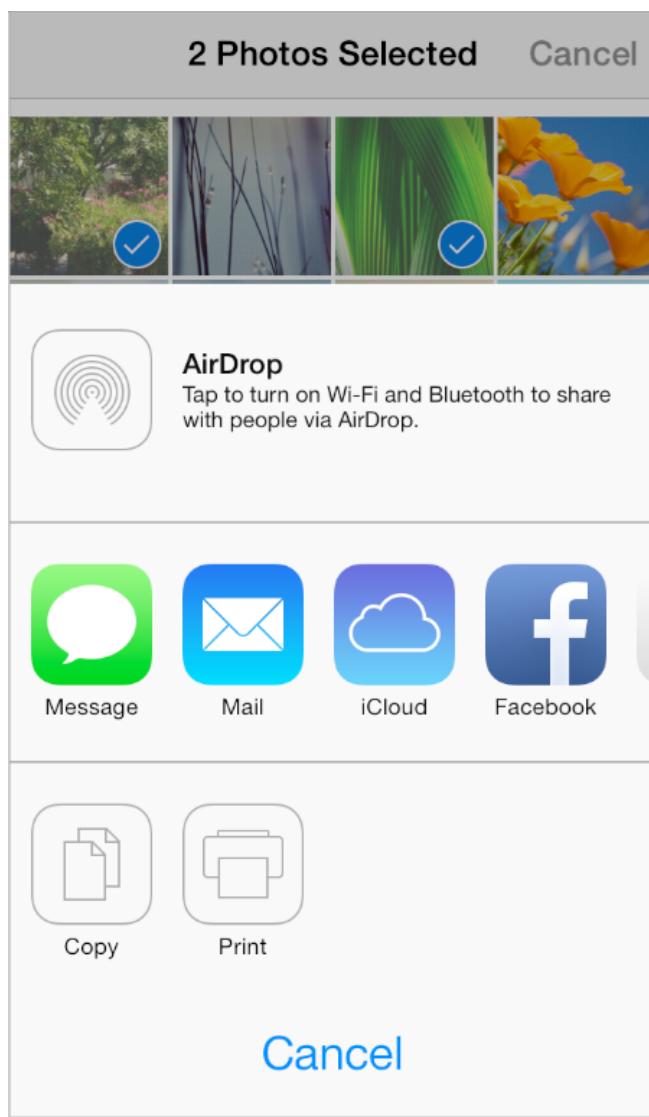
注意：如果你决定显示静态地图图像本身，务必使用一个高分辨率的图像以在用户放大它们时也能保持很好的显示品质。

给用户不同的方式对多种交通方式排序。很多因素会影响用户的交通决策（例如时间、天气和他们有多赶），所以能很容易地比较交通方式非常重要。例如，你可以让用户通过开始或结束时间、所需的步行距离、沿途站点数量、或者不同交通方式的换乘次数来对交通方式排序。无论你如何显示多种交通方式的排序，确保用户一眼就能分辨不同方式之间的区别。

考虑使用推送通知向用户告知其路线的重要信息。当交通状况发生变化时，尽可能的让用户知道，以便他们能够调整计划。例如，如果一趟火车延误或一班巴士暂时停开，用户可能需要选择其他路线前往目的地。另外，那些在两个步骤间包含很多站点的路线，用户可能希望在他们快要抵达行程的下一站时获得通知。

社交媒体

无论人们处于何种情境，他们总是期望能够读取所钟爱的社交媒体账户。iOS 可以很容易地以用户欣赏的方式，将社交媒体互动整合进你的 app 中。



给用户提供一种无需退出你的应用就能撰写信息的便利方式。你最好尽可能在你的程序中整合社交媒体支持，以便用户无需切换到另一个程序就能发布内容到他们的账户。Social framework 提供了一个撰写视图控件，让你能给用户展现一个他们可以编辑信息的视图。某些时候，在你向用户显示撰写视图以供编辑前，你可以在其中预置的自定义内容（当你向用户显示此视图后，只有他们能够编辑内容）。如需了解社交框架的程序接口——包括 `SILComposeViewController` 类——请参阅《Social Framework Reference》。

如果可能，避免请求用户登录社交媒体账户。社交框架与账户框架相配合以支持单点登录模式，因此无需请求用户再次授权你也能获取到访问账户号的授权。如果用户还没有登录账户，你可以显示允许他们这么做的 UI。

考虑使用活动视图控件，以帮助用户选择一个社交媒体账户。默认情况下，活动视图控件（即 `UIActivityViewController` 对象）会列出一些对当前所选内容生效的系统服务，包括通过「邮件」或者「信息」发送内容和张贴内容到社交媒体账户。当你在使用活动视图控件时，受益于用户对显示一系列服务的「分享」按钮的熟悉，你无需提供一个和社交媒体账户交互的自定义服务。如需了解如何在 app 中使用活动视图控件，请参阅 [「活动视图控制器」](#)（第 132 页）。

iCloud

无论用户正在使用哪台设备，iCloud 让他们可以读取他们所想要的内容。当你在 app 中整合 iCloud 时，无需特意进行同步，用户就能在不同设备中使用你的程序查看和编辑他们的私人内容。



为了提供这样的用户体验，你最好需要再次检查你的程序中存储、读取和呈现信息（尤其是用户创造的内容）的方式。如需了解如何在你的程序中支持 iCloud，请参阅《iCloud Design Guide》。

透明，是 iCloud 用户体验的基石：通常来说，用户不需要知道他们的内容存储在哪里，他们也很少去思考当前查看的内容是哪个版本。以下准则有助于为用户提供他们所期望的 iCloud 体验。

可以的话，让用户能轻易地为你的程序启用 iCloud。 在 iOS 设备上，用户在 iCloud 「设置」页面中登录他们的 iCloud 账户，最重要的是，他们期望你的程序能够自动地配合 iCloud 运作。但如果你认为用户可能想要在程序中选择是否使用 iCloud，在他们首次打开你的程序时，提供一个简单的选项以供设置。在大多数情况下，这个选项应当提供一个选择，让用户在你的程序中读取并使用所有 iCloud 的内容或者完全关闭。

尊重用户的 iCloud 空间。 iCloud 是用户花钱购买的有限资源，牢记这一点很重要。你应当使用 iCloud 去存储那些用户生成和知悉的信息，避免用其去存储 app 资源或者你能够重新生成的内容。同时需要注意的是，当用户的 iCloud 账户处于活跃时，iCloud 会自动备份程序文档文件夹下的内容。避免占用用户太多的空间，最好对你存储在文档文件夹中内容有所选择。

避免要求用户选择哪些文件存储在 iCloud。 通常来说，用户期望所有他们所关心的内容在 iCloud 中都是可用的。大多数用户不需要对单独的文档进行管理，所以你也不需要假设你的程序需要提供这样的体验。为了提供一个好的体验，你可能要重新构建你的程序的处理方式和公开内容，以便你能为用户执行更多文件管理任务。

确定哪些类型的信息会存储在 iCloud 中。除存储用户生成的文档和其他内容之外，你也可以存储少量的数据，例如用户在程序中的状态或偏好设置。存储这类信息你需要使用 iCloud key-value storage。例如，如果用户使用你的应用阅读一本杂志，你可能要用 iCloud key-value storage 去存储他们翻过的上一页，以便当他们在其他设备中再次打开这本杂志时可以从中断的地方继续阅读。

如果你使用 iCloud key-value storage 去存储偏好设置，确认其是用户希望在他所有设备上生效的设置。例如，一些设置在家里比在工作环境中更有用。在某些情况下，比起在用户的 iCloud 账户中存储偏好设置，放到你 app 的服务器上会更有意义，这样，无论 iCloud 是否启用，这些设置都能生效。

确保你的程序在 iCloud 不可用时的行为合理。例如，如果用户退出了 iCloud 账户，你应用不能使用 iCloud；或进入飞行模式，iCloud 变得不可用。在这些情况下，用户执行了一个切断读取 iCloud 的操作，所以你的程序不需要告诉他们这一点。然而，在这些用户做出的改变在其他设备上不可见时，可以适当地告诉用户，直至 iCloud 恢复访问。

避免给用户一个创建「本地」文件的选项。无论你的程序是否支持 iCloud，你都不应当鼓励用户去关注设备专用的文件系统。相反，你要让用户关注他们的内容在 iCloud 中的普遍可用性。

适时地自动更新内容。最好是用户不必做出任何操作就能在你的应用中读取最新内容。然而，你也需要在尊重用户设备空间和带宽限制与这样的体验之间把握平衡。如果你的用户使用大体积文档，可以适当地让他们控制是否从 iCloud 下载更新。如果你需要这样做，设计一种方式以显示文档在 iCloud 上可用的新版本。当用户选择更新文档时，如果下载耗时超过几秒，务必提供适当的反馈。

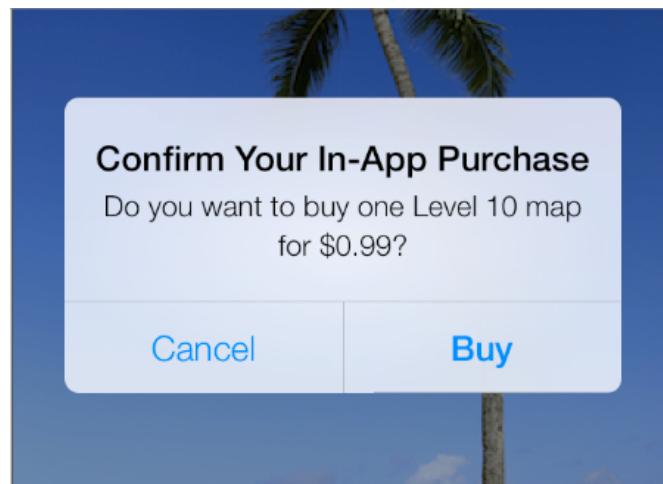
向用户警告删除文档的后果。当用户在一个支持 iCloud 的 app 中删除文档时，文档会从用户的 iCloud 账户和其他所有设备中移除。在你执行删除之前，恰当的做法是展示一个警告框描述其后果并获得确认。

将冲突尽可能快地告诉用户，但只在必要时这样做。使用 iCloud 程序接口，你应当无需打扰用户就能解决大部分不同版本之间的冲突。尽管这不太可能，但尽可能发现冲突以便你能帮助用户避免在错误的版本上浪费时间。你需要构想一种不显眼的方式向用户表明冲突的存在，接着，让用户轻易地发现版本间的不同从而做出决定。

确保搜索中包含了用户的 iCloud 内容。使用 iCloud 账户的用户常常认为他们的内容是普遍可用的，他们希望搜索结果会体现这个想法。如果你的程序允许用户搜索内容，确保你使用了合适的 API 以拓展对 iCloud 账户的搜索。

App 内购买

「App 内购买」（In-App Purchase）能让用户在你的程序中一个你所设计的商店里购买数字产品。



例如，用户可能会：

- 升级应用的基础版到高级版
- 续订每月新内容
- 购买虚拟项目，例如游戏中一个新等级或一件武器
- 购买下载新书

在你的程序中使用 Store Kit 框架嵌入一个商店，以支持「App 内购买」。当用户决定购买时，Store Kit 连接到 App Store，安全地处理付款，然后通知你的 app 可以提供所购买的项目。

重要：「App 内购买」只针对付款——你需要提供额外的功能，例如向用户展现你的商店、解锁内置特性和从你自己的服务器上下载内容。此外，所有你通过「App 内购买」销售的产品都必须在 App Store 注册。如需了解关于在程序中添加商店的技术条件，请参阅《In-App Purchase Programming Guide》。更多关于使用「App 内购买」的商务条件，请访问 [App Store Resource Center](#)。你也应该阅读授权协议，它可以告诉你关于可销售物品的确切信息和如何在你的程序中提供这些产品。

以下准则有助于设计一个用户称赞的支付体验。

优雅地在你的程序中整合商店体验。在你的程序中展示商品和处理用户交易时，营造一种宾至如归的感觉。你一定不想用户访问你的商店时让他们感觉进入了另一个 app。

使用简洁明了的标题和描述。最好是用户扫一眼就能马上发现他们感兴趣的项目。当你使用直接了当的语言和简炼的标题时，用户更容易理解你所出售的项目。

不要替换默认的确认警告框。在用户订购一个产品时，Store Kit 会在屏幕之上弹出一个确认警告框。你不应更改这个警告框，因为这会帮助用户避免意外的购买。

Game Center

Game Center 可供人们玩游戏、组织在线多人游戏、享受更多乐趣。玩家登录内置的 Game Center 便可以发现新游戏、添加新朋友以及浏览排行榜和成就。



作为一个游戏开发者，你可以使用 Game Kit API 向 Game Center 服务器告知分数和成就、在游戏界面中显示排行榜，并帮助用户发现其他玩家。如需了解如何将 Game Center 整合进你的程序，请参阅「Game Center Programming Guide」。

以下准则可以帮助你在程序中给用户以极致的 Game Center 体验。

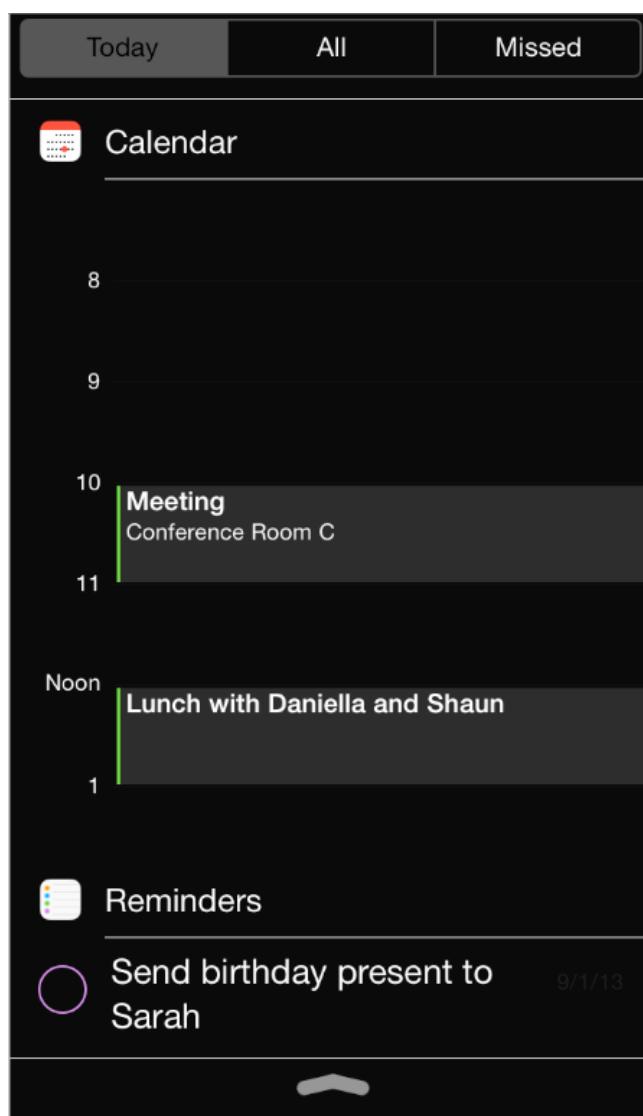
不要创建自定义界面来引导用户登录 Game Center。如果人们还没有在其设备上登录 Game Center，那当他们启动支持 Game Center 的程序时，系统会自动提示登录。没有必要去展示自定义的登录界面，而且还会干扰用户。

通常来说，使用标准的 Game Center 界面。在少数情况下，自定义游戏的 Game Center 界面可能很有意义，但这样做也有可能让人们感到迷惑。iOS 和 OS X 用户都很熟悉标准的 Game Center 界面，这会有一种同属一个游戏社区的归属感。

让用户能够关闭语音聊天。一些用户可能不想在启动游戏时自动运行语音聊天，大部分用户都会认可在具体场景下能够关闭语音聊天。

通知中心

「通知中心」能让用户有一个独立而方便的地方去查看 app 的通知。用户喜欢「通知中心」朴实低调的界面设计，也很看重其自定义每个 app 的通知方式的能力。



「通知中心」使用分段式列表依次展示用户感兴趣的 app 通知项目。此外，用户也可以在「通知中心」中查看来自内置 app 的信息，例如「天气」、「日历」、「提醒事项」和「股票」。

在一些有趣的事情发生时，iOS app 能够使用本地通知或推送通知让用户获知，例如：

- 收到一条信息
- 一个事件即将发生
- 新的数据可供下载
- 某些东西的状态改变了

本地通知由 app 安排并由 iOS 系统在同一设备上发布，无论程序当前是不是在后台运行。例如，某个日历或者待办事项应用可以安排一个本地通知来提醒用户一个即将到来的会议或者截止时间。

推送通知由 app 的远程服务器发送到 Apple 推送通知服务（Apple Push Notification service）上，再由其推送通知到所有安装了这个程序的设备上。例如，一款用户能和他人对战的游戏可以更新所有玩家的最新动向。

当你的 app 在后台运行时，你仍然可以接受本地和推送通知，但要以一种特定的方式向用户传达。

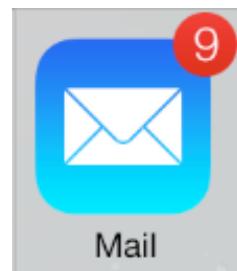
那些支持本地或推送通知的 iOS 应用能以多种方式融入「通知中心」，这取决于用户的偏好。为了确保用户能够自定义通知体验，你应当尽可能多地支持以下这些通知样式：

- 横幅 (banner)
- 提醒 (alert)
- 徽标 (badge)
- 声音 (sound)

横幅是一个小的透明区域，出现在屏幕顶部并在几秒后消失。除了推送信息，iOS 还在横幅中显示你应用的小图标，以便用户一眼就能知道哪个应用在通知他们（如需了解应用小图标，请参阅「[应用图标](#)」（第 175 页））。

提醒是一个标准的警告框视图，出现在屏幕之上并需要用户交互才会消失。在提醒中你可以展现通知信息和一个有标题的操作按钮（可选），但你无法控制提醒的背景或按钮样式。

徽标是一个出现在应用图标右上角的红色小圆点，上面显示着未读通知数。你无法控制徽标的大小和颜色。



这三种通知样式可以和自定义或者系统内置的**声音**配合使用。

注意：使用本地通知的 app 可以展现横幅、提醒、徽标和声音，但使用推送通知的程序的通知样式需要只能是此应用注册的推送分类。例如，如果一个推送通知的程序只注册了提醒，在收到通知时，用户是不能选择接收徽标和声音的。

当你设计你的通知可以提供的内容时，请务必遵循以下准则。

随时更新徽标上的数字。用户一浏览信息就更新徽标显得尤为重要，这样他们才不会认为有新通知抵达。在徽标上的数字归零的同时，也记得从「通知中心」移除相关的通知项目。

重要：不要将徽标用于除通知外的其他用途。要记得用户可以关闭你的程序的徽标，因此你不能肯定他们将看到徽标数字。

不要为同一事件重复发送通知。在选择通过一些方式对通知项目进行操作前，用户会一直看到它们。如果你对同一时间重复发送通知，你会塞满整个「通知中心」列表，很可能使得用户关掉你的通知。

放置自定义信息，但不包含你 app 的名字。你自定义的信息会展现在提醒、横幅和「通知中心」的列表项中。在你自定义的信息中不应包含你 app 的名字，因为 iOS 会自动在其中显示名字。

一条有用的本地通知或推送通知应当：

- 聚焦于信息，而非用户操作。避免告诉用户点击哪个警告框按钮或者如何打开你的应用。
- 足够简短，以一或两行显示。冗长的信息对用户来说很难浏览，他们不得不滚动提醒。
- 使用句式大写样式和合适的结束标点。如果可能，使用一个完整的句子。

注意：通常来说，「通知中心」中的项目比横幅可以展示更多的通知信息。如果需要，iOS 会截断你的信息以便其使用每一种通知样式。但为获得最佳效果，你不应该让信息被截断。

为提醒中的操作按钮提供一个自定义标题（可选）。提醒可以包含一两个按钮。在双按钮提醒中，「关闭」按钮在左边，操作按钮（默认标题视图）在右边。如果只有一个按钮，提醒中会显示「好」按钮。

轻点操作按钮，提醒框消失并启动你的程序。无需打开你的 app，轻点「关闭」或「好」按钮都能让提醒消失。

如果你想为操作按钮使用自定义标题，确保创建的标题清晰描述程序启动后会发生的操作。例如，游戏中可能会使用「开始游戏」作为标题以暗示轻点按钮会打开应用到一个用户可以开始玩的场景。要确保标题：

- 使用标题大写样式
- 足够简短，没有截断地适应按钮（也一定要测试本地化标题的长度）

注意：当设备在锁屏时收到一条通知，你自定义的按钮标题也会被展示在用户看到的「滑动来查看」消息中。当这种情况发生，你自定义的标题会自动转成小写并替代消息中的「查看」这个词。

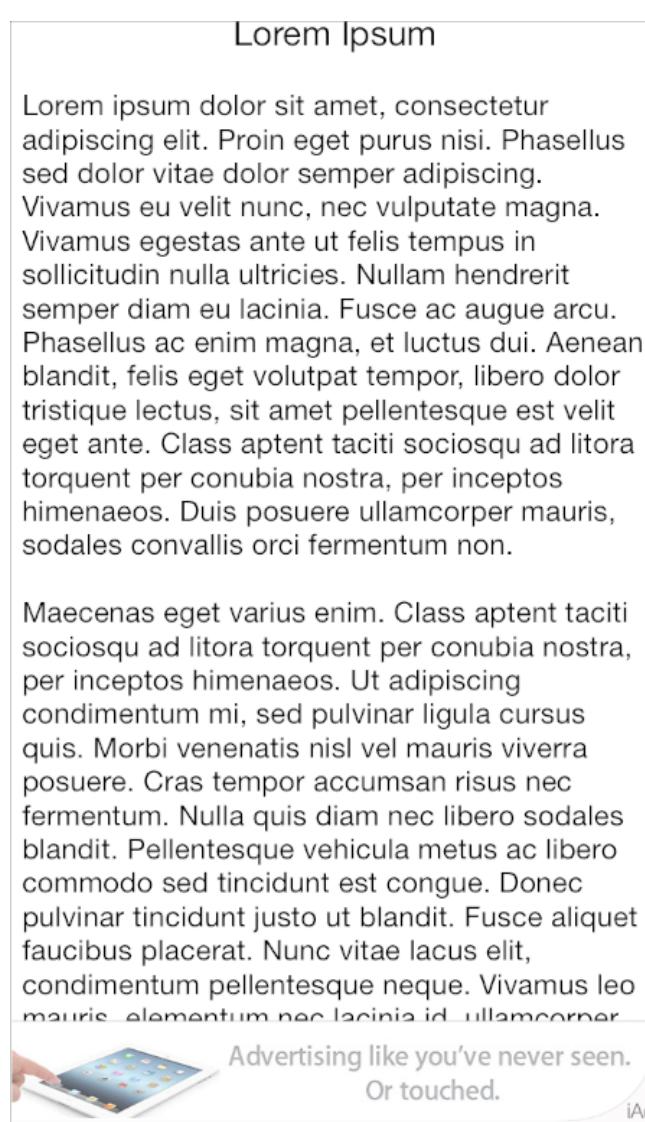
提供一个用户可以在收到通知时选择听到的声音。在用户没有看着设备屏幕时，声音可以吸引用户注意。当用户认为一个通知很重要时，他们可能想要打开声音。例如，日历应用可能会为一个提醒播放声音以提醒用户即将到来的事件。或者，一个协同任务管理的应用可能随着新徽标播放声音，以提醒一个远程同事已经完成了一项任务。

可以使用内置的提醒声音或者自定义一个。如果你要自定义一个声音，确保其简短、独特且制作精良。（如需了解更多声音的技术要求，请参阅《Local and Push Notification Programming Guide》中的「Preparing Custom Alert Sounds」部分。）请注意，在收到通知时你不能让设备自动振动，因为是用户控制提醒是否伴随振动。

可以考虑提供一个启动画面。除了已有的启动画面，你还可以在用户从通知中启动你的程序时提供一个不同的启动画面。例如，游戏可以指定一个和游戏过程中的截图相似的启动画面，而非和开场菜单页面相似的图像。（如需了解如何创建启动画面，请参阅「启动画面」（第 180 页）。）

iAd 富媒体广告

当用户看到或者点击你在你的 app 中放置的广告时，你可以收到收入。（下面的示例项目中你可以看到一个 iAD 横幅的占位符。）



你可以在你界面的某个视图中展示一幅由 iAd Network 支持的广告。最初，这个视图可以包含广告横幅，作为进入完整 iAd 体验的入口。当人们轻点横幅，广告会执行预先设置的动作，例如播放一段影片、显示交互性内容或者启动 Safari 去打开一个网页。这个动作可以覆盖你的界面来显示内容或可能会导致你的 app 切换到后台。

这里有三种你可以在你的 app 中展示的横幅类型：标准、中等矩形和全屏。所有类型的横幅都有同样的目的——即，吸引用户进入广告——但它们在外观和功能上略有不同。

标准横幅会占据页面中某个小区域，通常随页面出现而出现。你可以挑选应该显示标准横幅的 app 页面并在布局中为横幅视图留下空间。

所有的 iOS app 都可以显示标准横幅。你可以使用由 `ADBannerView` 类提供的视图来在你的 app 中显示标准横幅。

中等矩形横幅的行为和标准横幅类似——和标准横幅一样——你可以选择中等矩形横幅出现的位置。中等矩形横幅只在 iOS 6.0 及更高版本的 iPad app 中可用。你可以使用由 `ADBannerView` 类提供的视图来在你的 app 中显示中等矩形横幅。

全屏横幅会占据大部分或者全部页面，通常只在 app 流程中的某些特定时段或者特定位置出现。你可以选择是以模态形式还是作为滚动的单独页面显示。

使用由 `ADInterstitialAd` 类提供的视图来在你的 app 中显示全屏横幅。

所有类型的横幅都在 iAd 框架内出现，并会在右下角显示 iAd 标识。iAd 框架被设计为出现在你的 app 页面底部时效果最佳。

条栏	标准横幅视图位置
屏幕底部没有条栏	在屏幕底部
屏幕中任何地方都没有条栏	在屏幕底部
工具栏或标签栏	底部栏之上

为确保与横幅广告的无缝整合并提供最佳用户体验，请遵循以下准则。

在屏幕底部或附近放置标准横幅视图。这个位置可能会稍有不同，这取决于屏幕底部是否存在条栏。

将中等矩形横幅放置到不会干扰用户内容的地方。和标准横幅视图一样，中等矩形横幅视图出现在屏幕底部或附近时效果最佳。将横幅放置到屏幕底部也会降低吸引用户注意的可能性。

在用户体验的间隙呈现全屏横幅。如果在你 iPad app 的流程中有一些自然的间断或情境变化，那么模态的呈现方式会很合适。当你模态地显示一个全屏横幅时（通过使用 `presentFromViewController:`），用户要么进入广告要么关闭广告。基于这个原因，在用户期望一个体验变化时使用模态的呈现方式会是一个不错的主意，比如当他们完成某个任务后。

当 app 视图之间存在过渡时则非模态地呈现全屏横幅。如果用户会通过频繁的页面切换来体验你的 app，比如翻一本杂志或者滚动浏览相册，那么非模态的呈现方式会非常合适。当你非模态地呈现一个全屏横幅时（通过使用 `presentInView:`），你可以在界面中保留条栏以便用户可以使用 app 控件来回到过去或者返回广告。和所有的横幅一样，全屏横幅会在用户点击时提供 iAd 体验，但如果合适，你的 app 可以在横幅区域内响应其他手势（例如拖拽或轻扫）。

确保使用合适的动画来展现和隐藏非模态的全屏横幅视图。例如，一款杂志阅读器 app 可能会使用和展现其他内容页面一样的翻页动画来展现一个横幅。

确保所有你 app 的横幅在正确的时间出现在正确的地方。当人们没有感觉像是在任务流程中被打断时，他们会更乐意进入 iAd 体验。这对于那些沉浸式 app 尤其重要，比如游戏。你不会想把横幅视图放到与玩游戏相冲突的地方。

避免在用户只看一眼的页面中显示横幅。如果你的 app 包含用户在浏览他们所关注的内容时会很快扫过的页面，那最好避免在这些页面中显示横幅。用户倾向于点击那些在页面上停留不止两三秒的横幅。

尽可能在横屏和竖屏两个方向上显示横幅。最好让用户无需改变设备方向就能在使用 app 和观看广告之间切换。同样，支持两种方向能让你支持更宽尺寸的广告。如需了解如何确保横幅视图能响应方向变化，请参阅《iAd Programming Guide》。

不要让标准或中等矩形横幅在页面中滚动。如果你的 app 在页面中显示了滚动内容，确保横幅视图在其位置始终显示。

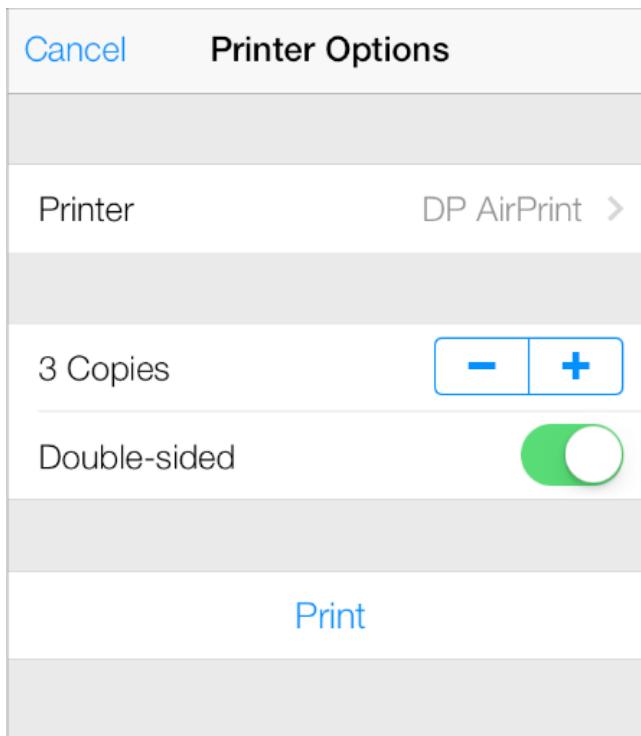
当用户观看或与广告交互时，暂停其他需要用户关注或交互的活动。当用户选择观看广告时，他们不会想感觉到 app 中有他们错过的事件，并且他们也不希望你的 app 打断广告体验。一个好的经验法则是暂停和你在切换到后台时所停止的相同活动。

除少数情况外，不要暂停一个广告暂停。通常来说，当用户在观看和与广告交互时，你的 app 会继续运行并接受事件，因此很有可能会发生迫切需要用户立即关注的事件。然后，在少数一些场景下需要授权取消正在播放的广告。一个例子是一款提供 VoIP 服务的 app，在这样一个 app 中，当收到一个电话时取消正在播放的广告是必要的。

注意：取消广告可能会对你的 app 能接受的广告类型和你可以收到的收入产生影响。

AirPrint

使用 AirPrint，用户可以无线打印在你程序中的内容和使用「打印中心」程序来检查打印任务。



你可以利用对图像和 PDF 内容的内置打印支持，或者使用打印专用的程序接口来自定义格式和渲染。iOS 会找到打印机，在选中的打印机上安排并执行打印任务。

通常，当用户想要打印什么东西时，在你的应用中轻点标准的「分享」按钮即可。在出现的视图中选择「打印」后，他们可以选中一台打印机，设好打印选项，然后轻点打印按钮开始工作。在 iPhone 上，这个视图以操作列表的方式从屏幕底部上滑出现；在 iPad 上，这个视图出现在从打印按钮弹出的菜单中。

用户可以在「打印中心」中检查提交的打印任务，这是一个只在进行打印工作时才出现的后台系统程序。在「打印中心」里，用户可以查看当前打印队列，获得某个打印任务的详细信息，甚至取消任务。

只需要一丁点的代码，你就能在你的 app 中支持最基本的打印（如需了解如何在你的代码中添加打印支持，请参阅《Drawing and Printing Guide for iOS》）。为了确保用户会喜欢你程序的打印体验，遵循以下准则：

使用系统自带的「分享」按钮。用户对次按钮的意义和功能都很熟悉，所以可能的话尽可能使用它。唯一的例外是，如果你的 app 没有工具栏或者导航栏。碰到这种情况，由于系统「分享」按钮只能用在工具栏或导航栏中，因此你需要设计一个自定义的打印按钮，来显示在你的 app 主要的界面中。

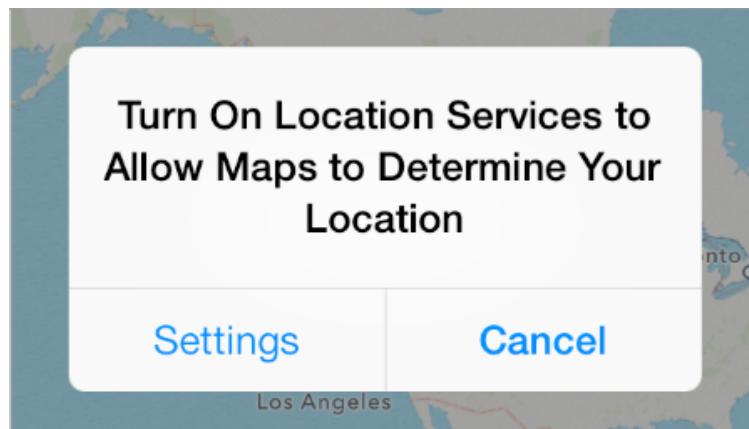
如果打印在当前情境下是首要功能，则显示「打印」项目。如果在当前情境下打印是不适当的，或者用户不想要打印，在「分享」按钮相关的视图中不要出现「打印」项目。

合适的话，给用户提供额外的打印选项。例如，你可能允许用户选择页码范围或者要求打印多份。

如果用户不能打印，不要展示打印专用界面。在你展示打印界面前，一定要检查用户的设备是否支持打印。如需了解如何在你的代码实现，请参阅《UI Print Interaction Controller Class Reference》。

位置服务

「位置服务」（Location Services）允许程序确定用户大致的地理位置，设备所指的方向和所移动的方向。在 iOS 6 以后，诸如「通讯录」、「日历」、「提醒事项」和「相册」等系统服务页允许 app 访问用户存储在其中的数据。



尽管用户赞赏使用一个很了解他们的应用所带来的便利，但仍然希望有选项去控制数据隐私。例如，用户很喜欢能自动给内容打上位置标签，或者发现附近的朋友，但也希望能够在不想选择和他人分享位置时禁用这些特性。（如需了解如何让你的程序感知位置，请参阅《Location Awareness Programming Guide》。）

下面这些准则可以帮助你以用户感觉舒适的方式向用户请求数据。

确保用户理解之所以请求他们分享个人数据的原因。对一个用户没有看到明显需要就获取个人数据的行为心存怀疑，这很正常。为了避免让用户感到不舒服，确保只在用户尝试使用一个明确需要获取位置信息的功能时弹出警告框。例如，当「位置服务」关闭时用户也能使用「地图」，但当使用发现和追踪当前位置的功能时，他们会看到一个警告框。

如果看起来不太需要，请描述你的 app 为什么需要这些信息。你可以提供文案，显示在警告框中「『应用名字』想访问您的通讯录」等系统标题下方。为了让用户能理解你为什么要要求访问他们的信息并且不会感到压力，你需要让这个文案详尽但礼貌。你的解释文案应当：

- 不要包括你的程序名称。系统自带的警告框标题已经包含了程序名称。
- 清楚地描述为什么你的程序需要这些数据。如果合适，你也可以解释你的程序不会将数据用于哪些用途。
- 使用以用户为中心的术语并保持本地化。
- 尽可能简短，同时还易于理解。尽量避免出现一个以上的句子。
- 使用句子大写样式。（句子大写样式意味着首字大写，其余的话都是小写，除非是专业名字或形容词）

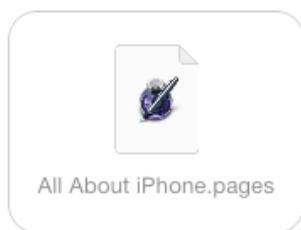
在应用启动时请求授权，但只在你的程序没有用户数据就无法操作主要功能时这样做。如果 app 的主要功能明显需要获取他们的个人信息，用户并不会介意这一点。

避免在用户确切地选择一个需要位置数据的功能前自动触发警告框。这样的话，你不会让用户感到好奇，为什么你的程序会在他们做一些看上去和位置毫不相干的事情时需要他们的个人数据。（请注意，获取用户的「位置服务」偏好设置不会触发警告框。）

对于位置数据，检查「位置服务」的偏好设置以避免触发不必要的警告框。你可以使用 Core Location 的程序接口去获取这个设置（如需了解如何做到这点，请参阅《Core Location Framework Reference》）。通过这种方法，你可以尽可能只在需要获取位置信息时触发警告框，或者可以完全避免弹出警告框。

快速预览

使用「快速预览」（Quick Look），用户可以在你的程序内部预览一个文档，即使你的程序不能打开文档。例如，你可以允许用户预览从网络下载或者从其他渠道接收到的文档。



如需了解如何在你的程序中支持「快速预览」，请参阅《Document Interaction Programming Topics for iOS》。



在用户在你的程序中预览一份文档前，他们可以在你所定制的界面中看到关于文档的信息。例如，在用户从一封邮件中下载一个附件文档后，iPad 中的「邮件」应用会在邮件的特定界面中展示文档图标、标题和大小。用户可以点击这个区域以预览文档。

在你的程序中，你可以在一个新视图、全屏界面或者模态视图中展现一个文档预览。你所选择的展现方式取决于你的程序所运行的设备。

在 iPad 上，模态地呈现文件预览。 iPad 的大屏幕很适合在一个沉浸的环境中呈现文件预览，用户也可以很容易离开。放大这样的转场效果特别适合显示预览。

在 iPhone 上，最好是在一个有导航栏的专用视图中显示文档预览。 这样做会让用户能够通过导航来到文档预览并离开，不至于在你的程序中迷路。虽然在 iPhone app 中以模态方式展现文档预览是可能的，但并不推荐这样做。（注意，放大这个转场效果在 iPhone 上不可用。）

另外请注意，「快速预览」的导航控件放置在展现文档预览的导航视图中的导航栏上。（如果你的视图已经包含了工具栏，相应地，「快速预览」的导航控件应该放在工具栏中。）

声音

无论声音是你 app 体验不可或缺的部分，还是仅仅只是偶尔用来增强效果，你都需要了解用户对声音行为有何期望并如何满足他们的期望。

理解用户期望

人们可以使用设备控件调节声音，也可能会使用有线或无线的耳机。他们也对用户行为如何影响其所听到的声音有各种期望。虽然你可能感觉其中一些期望很奇怪，但它们都遵循用户控制的原则——是用户而不是设备来决定什么时候听到声音是合适的。

当用户想要以下结果时会将设备切换到静音：

- 避免被不符合期望的声音打断，例如电话铃声和新消息的提醒声音
- 避免听到由用户操作所产生的声音，例如键盘声音，其他反馈声音，故障声音或者应用启动声音
- 避免在玩游戏时听到不相关的声音，例如偶然的故障声音

注意：在 iPhone 上，人们可以通过切换「响铃/静音模式」将设备转为静音；在 iPad 上，则可以开启「静音模式」。

例如，在剧院中用户想将设备调成静音，以避免打扰剧院中的其他听众。在这种情况下，用户仍然想要能够在他们的设备上使用 app，但不想被突然的声音所吓到，例如铃声或新消息声音。

对于用户特别设置的声音，「响铃/静音模式」（或「静音模式」）并不能关闭它们。例如：

- 在多媒体播放器中的媒体播放不能被静音，因为它是由用户主动明确请求的。
- 「时钟」中的闹钟不会静音，因为它是由用户明确设置的。
- 在语言学习应用中的声音片段不应被静音，因为用户做出了明确操作想听到它。
- 在语音聊天应用中的对话不应被静音，因为用户启动它的目的就是为了能够语音聊天。

用户用系统的音量键可以调整设备播放的所有声音，包括歌曲、应用声音和设备声音。无论处于「响铃/静音模式」的哪一挡上，用户都可以使用音量键静音。使用音量键调节 app 当前播放的音频，也可以调节整个系统的声音，铃声音量除外。

iPhone: 在没有音频播放时，使用音量键是调节铃声音量。

用户使用耳机和耳塞来私密地收听声音，这还解放了他们的双手。无论这些配件是有线还是无线连接，用户对其用户体验都有特殊的期望。

当用户插上耳机或者连接上一个无线音频设备，他们是想私密地继续听当前播放的音频。基于这个原因，他们期望正在播放音频的 app 能够不被打断继续播放。

当用户拔出耳机，断开无线设备（又或者，设备超出范围或关机），他们不想把刚才听到的声音自动地共享给其他人。所以，用户期望正在播放音频的程序可以暂停，等到准备好了允许它们重新播放。基于这个原因，他们期望正在播放音频的 app 能够暂停，并在他们准备就绪时允许他们重新播放。

定义应用的声音行为

如果需要，你可以调节相对独立的音量水平以使 app 输出最好的混响声音。无论你是通过音量按键还是音量滑块调节音量，最终音频输出的音量应该受系统音量管制。这意味着对 app 音频输出的控制权仍然在用户手中。

如果合适的话，确保你的 app 能够展示音频线路选择器。音频线路 (**audio route**) 是音频信号的电子通道，例如从一台设备到耳机或者从一台设备到麦克风。即使用户不能插入或者拔出一个无线音频设备，他们仍然期望能够选择一个不同的音频路线。为了解决这个问题，iOS 会自动地展示一个控件允许用户选择一个音频输出路线（使用 MPVolumeView 类以在你的 app 中展现这个控件）。由于选择不同的音频路线是用户发起的行为，用户期望当前播放的音频无需暂停就能继续播放。

如果你需要展示音量滑块，确保你在使用 MPVolumeView 类时使用系统所提供的音量滑块。需要注意的是，在当前激活的音频输出设备不支持音量控制时，音量滑块会被替换为合适的设备名字。

如果你的 app 只是在界面的某些功能中附带音效，请使用 System Sound Services。System Sound Services 是一种在 iOS 中呈现警告和 UI 音效并调用振动的技术，但并不适用于其他目的。当你使用 System Sound Services 来呈现音效时，你不能决定你的声音会如何与设备中其他的声音交互，或它会如何响应中断和设备配置的变化。如需了解如何使用这种技术的示例项目，请参阅《Audio UI Sounds (SysSound)》。

如果声音在你的 app 中扮演一个很重要的角色，使用 Audio Session Services 或者 AVAudionSession 类。这些程序接口不会直接呈现声音，相反，它们帮助你的声音如何与设备中的声音配合，并响应中断和设备配置的变化。

iPhone: 无论你使用何种技术呈现声音，如何定义它的行为，电话总是能够打断当前运行的 app。这是因为没有 app 应当阻止人们接听来电。

在 Audio Session Services 中，**音频会话 (audio session)** 是你的 app 和系统之间的调节者。音频会话中最重要的一个方面是**类别 (category)**，其定义了你 app 的声音行为。

为获得 Audio Session Services 的好处和提供用户所期待的声音体验，你需要选择和你的 app 行为最相符的分类。这取决于你的 app 是只在前台播放声音还是也能在后台播放。请遵循以下准则做出选择：

- 选择音频会话类别要基于其语意上的意义。**选择一个目的明确的类别，这会确保你的 app 行为符合用户期望。此外，如果将来重新定义了具体的行为，这也给你的 app 一个绝佳机会使其正常运转。
- 在少数情况下，给音频会话添加一个属性以修改类别的标准行为。**一个类别的标准行为代表着大部分用户所期望的样子，因此在修改其行为前你应当认真考量。例如，为了确保你的声音比其他声音大（电话铃声除外），你可能会添加一个下沉（ducking）属性，以便符合用户对你 app 的预期。（如需了解更多关于音频会话属性的内容，请参阅《Audio Session Programming Guide》中「Fine-Tuning the Category」一节。）
- 考虑基于当前设备的音频环境来选择你的类别。**这可能会有用，例如，用户可以一边听其他音频一边使用你的 app，而不会被你的音轨打断。如果你这样做，请确保在你的 app 启动时不要强制用户暂停他们的音乐或者做出明确的音轨选择。
- 通常来说，避免在你的 app 运行时改变类别。**可能改变类别的最大原因是你的 app 需要在不同时候支持录制和回放。在这种情况下，比起选择播放和录音（Play and Record）类别，根据需要在录音（Record）和回放（Playback）类别中切换确实会更好。这是因为选择录音（Record）类别能确保在录制过程中不会有提醒（如新消息提醒）会发出声音。

表 30-1 列出了你可以使用的音频会话类别。不同的类别允许使用「响铃/静音」模式或者「静音」模式（或设备锁屏）将声音静音，和其他音频混响，或者在 app 处于后台时播放。（如需了解在程序接口中出现的具体分类和属性名称，请参阅《Audio Session Programming Guide》。）

表 30-1 音频会话类别及其相关行为

类别	含义	是否静音	混响	是否在后台
独奏环境（Solo Ambient）	声音会提升 app 的功能效果，并应把其他音频静音。	是	否	否
环境（Ambient）	声音会提升 app 的功能效果，但并不把其他音频静音。	是	是	否
回放（Playback）	声音作为 app 功能必不可少的部分，并且可能会和其他音频混响。	否	否（默认），是（在 Mix With Others 属性被添加时）	是
录音（Record）	音频是由用户录制的。	否	否	是
播放和录音（Play and Record）	声音呈现为音频输入和输出，两者循序或同时进行。	否	否（默认），是（在 Mix With Others 属性被添加时）	是
音频处理（Audio Processing）	App 执行硬件辅助的音频解码（并非播放和录音）。	是/否	否	是*

* 如果你选择了 Audio Processing 类别并在后台执行音频处理，你需要在完成音频处理之前防止你的 app 停止运行。如需了解如何实现，请参阅《iOS App Programming Guide》中「Implementing Long-Running Background Tasks」一节。

这里有一些场景，描述了如何选择音频会话类别以提供用户所赞赏的声音体验：

场景 1：一款帮助用户学习新语言的教育类 app。你需要：

- 在用户轻点特殊控件时，播放反馈声音
- 在用户想要听正确的发音示例时，播放事先录制的单词和词汇

在这个 app 中，声音作为主要功能必不可少。用户使用这个 app 来听所学语言的单词和词汇，因此即使是在设备锁屏或者切换为静音时都应当发出声音。由于用户需要听到清晰的发音，他们期望其他正在播放的声音都静下来。

为了在这个 app 中提供用户期待的声音体验，你需要使用回放（Playback）类别。尽管这个类别稍加修改后可以支持和其他音频混响，但这个 app 应当使用默认行为，以确保其他音频不会和用户确切选择的教育内容相冲突。

场景 2：一款 VoIP 电话 app。你需要提供：

- 接收音频输入的能力
- 播放音频的能力

在这个 app 中，声音作为主要功能必不可少。用户在用这个 app 和其他人通话时，往往正在使用其他 app。他们期望在切换设备到静音或者锁屏时仍然能够接听电话，并希望在通话过程中其他音频保持安静。用户同样希望当 app 在后台运行时仍然能够接听电话。

为了在 app 中提供所期待的用户体验，你应当使用播放和录音（Play and Record）类别。另外，你需要确保只在需要时激活音频会话，以便用户可以在不通话时使用其他声音。

场景 3：一款允许用户操作角色完成不同任务的游戏。你需要提供：

- 各种游戏音效
- 一个音乐音轨

在这个 app 中，声音会显著提升用户体验，但对于主要任务并不是必须的。同时，用户很容易欣赏可以将游戏静音，或是在听他们乐库中的音乐时替换游戏原声的做法。

最好的策略是在你的 app 启动时看看用户是不是在听其他音频。不要让用户选择他们是想听其他音频还是你的原声。反而，调用 Audio Session Services 的函数 `AudioSessionGetProperty` 去读取 `kAudioSessionProperty_OtherAudioIsPlaying` 的值。基于这个值，你可以选择环境（Ambient）或者独奏环境（Solo Ambient）类别（两者均允许用户让游戏静音）：

- 若用户正在听其他音频，你应该假设他们更希望继续听，对强制播放游戏原声以替代的做法并不喜欢。这种情况下，你可以选择环境（Ambient）类别。
- 若用户在你的 app 启动时没有听其他音频，你可以选择独奏环境（Solo Ambient）类别。

场景 4：一款提供准确实时导航，指引用户到达目的地的 app。你需要提供：

- 为行程中每一步语音提示方向
- 一些反馈声音
- 可以让用户继续听他们音频的能力

在这个 app 中，无论它是否在后台，语音导航指令都是其最主要的任务。基于此，你应当使用回放（Playback）类别，它允许你在设备锁屏、切换为静音或者你的 app 在后台运行时播放声音。

为了让用户在使用你的 app 时听其他音频，你可以添加

`kAudioSessionProperty_OVERRIDECATEGORYMIXWITHOTHERS` 属性。无论如何，你也想要确保用户在他们当前播放的声音智商听到语音指令。为了做到这样，你可以在音频会话中使用 `kAudioSessionProperty_OtherMixableAudioShouldDuck` 属性，以确保你的音频比所有当前播放的音频都大声（iPhone 上的电话声音除外）。这些设置允许 app 在后台运行时重新激活音频会话，这确保用户可以实时获知导航变更。

场景 5：一款允许用户往网站上传文字和图片的博客 app。你需要提供：

- 简短的启动音乐
- 各种伴随用户操作的简短音效（例如在成功上传一篇日志后播放的音效）
- 在发布失败时的警告音

在这个 app 中，声音可以提升用户体验，但不是必需的。主要任务与声音无关，用户页不需要听到任何声音就能成功使用 app。在这种情况下，你可以使用 System Sound Services 以生成声音。由于 app 中所有的声音内容都要和此 app 的既有目的相符，播放的 UI 音效和警告音需要如用户期待那样，服从于设备锁屏和「响铃/静音」（或「静音」）模式。

管理音频中断

某些时候，正在播放的声音会被其他 app 的声音打断。例如，在 iPhone 中接听一个来电的时间段里会中断当前 app 的声音。在多任务环境中，这样的声音中断频率会很高。

为提供用户所喜欢的声音体验，iOS 需要你：

- 确定你的 app 会导致的音频中断类型
- 在音频中断结束后合适地响应

每个 app 都需要确定其可能导致的音频中断类型，但不是每一个 app 都需要决定在一段音频中断结束后如何响应。因为大部的 app 都应当在一段音频中断后恢复声音。只有以多媒体回放为主的 app，那些提供多媒体回放控件的 app 需要采取额外的步骤以决定恰当的响应。

从概念上来说，基于中断的声音类型和用户对中断结束后的期望，这里有两种音频中断类型：

- **可恢复的中断**，是由一段用户认为是其主要聆听体验的插曲的音频导致的。

在一段可恢复的中断结束后，有回放控件的 app 应该恢复到中断发生时的状态，无论当时是正在播放还是处于暂停。没有回放控件的 app 应该恢复到播放状态。

例如，设想一位用户正用 iPhone 听音乐，突然在听到一半时来了一个 VoIP 电话。他接了电话，期望播放 app 在通话期间保持静音。通话结束后，用户希望播放 app 能自动恢复到正在播放的歌曲，因为音乐是他们首要的聆听体验，在来电时不得不将音乐停止。从另一方面来说，如果用户在收到来电前已经将音乐停止，通话结束后他们仍然希望音乐继续保持暂停。

其他可能会导致可恢复中断的 app 方面的例子，例如播放闹铃、声音指示（例如语音驾驶导航）或其他间歇音频方面的 app。

- **不可恢复的中断**，由用户视为主要聆听体验的音频引起，例如一个多媒体播放 app 的声音。

在一段不可恢复的中断结束后，有回放控件的 app 不应当回复音频播放。没有回放控件的 app 应当恢复音频播放。

例如，设想一位用户正在使用一个音乐播放 app（第 1 个音乐 app）时，被另一个音乐播放 app（第 2 个音乐 app）打断了。作为响应，用户决定继续听一会儿第 2 个音乐 app。在退出第 2 个音乐 app 后，用户不会希望自动恢复第 1 个音乐 app 的播放，因为用户已主动将第 1 个音乐 app 视为他们的主要聆听体验。

以下准则会帮助你决定提供哪些信息以及在一段音频中断结束后如何继续：

确定 app 导致的音频中断类型。你在你的音频播放完毕后通过以下两种方式结束你的音频：

- 如果你的 app 导致了可恢复的中断，通过 `AVAudioSessionSetActiveFlags_NotifyOthersOnDeactivation` 标记来结束音频会话。
- 如果你的 app 导致了不可恢复的中断，不要用任何标记（flag）来结束音频会话。

无论提供与否，在合适情况下，标记（flag）会让 iOS 赋予被打断的 app 以自动恢复音频播放的能力。

在音频中断结束后决定是否应该恢复播放。你可以根据在你的 app 中所提供的音频体验做出决定。

- 如果你的 app 有回放控件可供用户用于播放或暂停音频，你需要在音频中断结束后检查 `AVAudioSessionInterruptionFlags_ShouldResume` 标记。

如果你的 app 收到了 Should Resume 标记，你应当：

- 若你的 app 在播放音频时被打断，恢复音频播放
- 在你的 app 没有播放音频时被打断，无需回复音频播放
- 如果你的 app 没有显示任何用户可以用来播放或暂停的多媒体播放控件，在音频中断结束后你通常应当恢复音频播放，无论 Should Resume 标记是否出现。

例如，一个会播放背景音乐的游戏应当在中断结束后自动地回复之前正在播放的背景音乐。

合适地处理远程多媒体控制事件

在人们使用 iOS 多媒体控件或辅助控件——例如耳机控件时，app 可以接受远程控制事件。这让你的 app 可以接受用户没有经由你界面的指令，无论你的 app 是正在前台还是后台播放音频。

App 可以发送视频到支持 AirPlay 的设备——例如 Apple TV——并在挂到后台的同时继续播放。这样一个可以接受用户通过远程控制事件输入的 app，在其处于后台时，用户也可以控制视频播放。此外，这种类型的 app 还可以在中断结束后在后台重新激活音频会话。

多媒体播放 app，尤其需要适当响应多媒体远程控制事件，特别实在其播放音频或处于后台时。

为满足和你的 app 可以在后台播放多媒体内容这一特权相符的责任，请确保遵循以下准则：

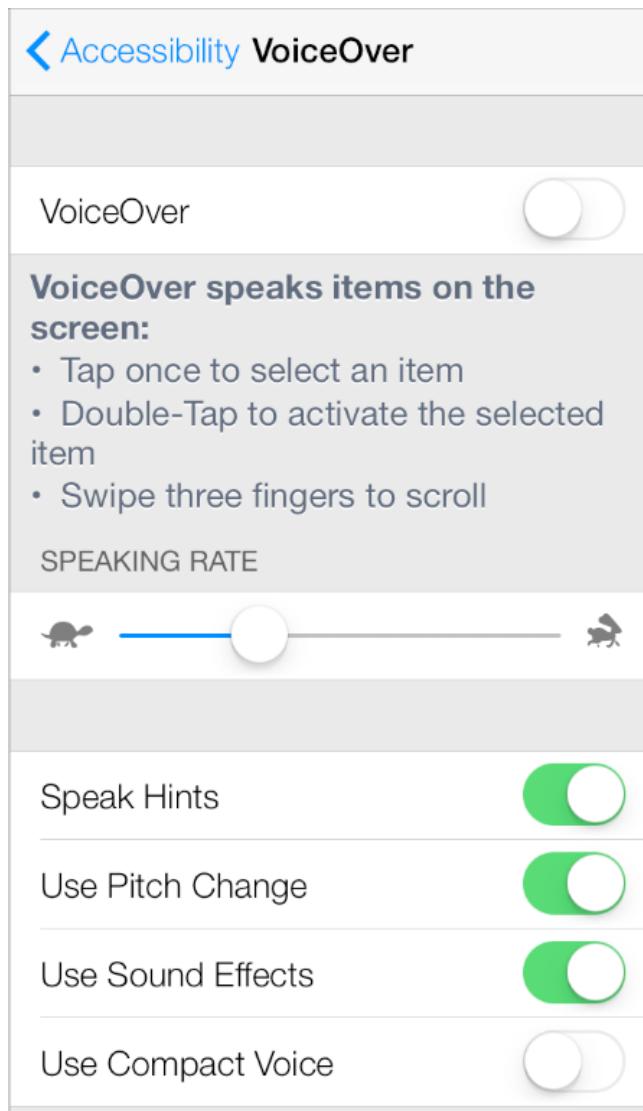
让你的 app 只在适当的时候接受远程控制事件。例如，如果你的 app 是帮助用户阅读内容，搜索信息，或者聆听音频，它应当只在用户处于音频情境时才接受远程控制事件。当用户离开音频情境，你应当放弃接受事件的能力。若你的 app 让用户可以在支持 AirPlay 的设备上播放音频或视频，它应只在播放期间接受远程控制事件。当用户在你的 app 中处于非多媒体情境时，遵循这些准则可以让他们消费另一个 app 的多媒体内容——并用耳机操控。

尽可能使用系统控件以提供 AirPlay 支持。当你在使用 `MPMoviePlayerController` 类去支持 AirPlay 播放时，你可以利用标准控件的优势让用户选择一个当前范围内支持 AirPlay 的设备。或者你可以使用 `MPVolumeView` 类以显示用户可以选择的支持 AirPlay 的音频或视频设备。用户习惯了标准控件的样式和行为，所以他们会知道在你的 app 中如何使用它们。

不要改变一个事件的用途，即使它在你的 app 中没有任何意义。用户期望 iOS 的多媒体控件和辅助控件在所有 app 中保持功能上的一致性。你不需要处理你的 app 用不上的事件，但你处理的事件一定要达到用户所期望的结果。如果你要重新定义事件的含义，你会混淆用户并冒险地讲他们带到一种未知的境地，除了退出你的 app，他们没有其他办法。

VoiceOver

VoiceOver 为盲人、低视力用户和有学习障碍的用户提升了可达性。



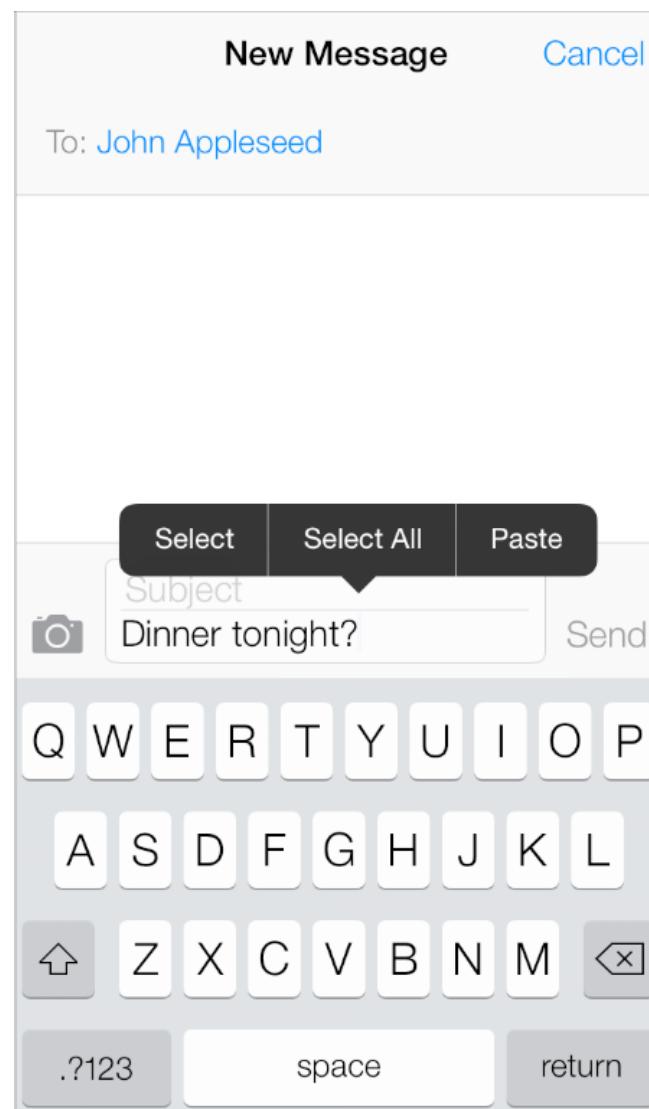
为确保 VoiceOver 用户可以使用你的 app，你可能需要在你的界面中提供一些关于视图和控件的描述信息。支持 VoiceOver 并不需要你对用户界面的视觉设计做任何改变。

只要你以完全标准的方式使用标准的 UI 元素，就没有多少额外工作要做。越多地自定义你的界面，你需要提供的自定义信息就越多，以便 VoiceOver 可以准确描述你的 app。

让 VoiceOver 用户可以使用你的 iOS app，这会增加你的用户基数并帮助你进入一个新的市场。支持 VoiceOver 还有助于你遵循由不同的理事机构提出的无障碍准则。

编辑菜单

用户可以在文本视图、web 视图或者图片视图中呼出编辑菜单，以执行诸如「剪切」、「粘贴」和「选择」等操作。



你可以在你的 app 中调整一些菜单的行为，以给予用户对内容更多的的控制权。例如，你可以：

- 设定适合于当前情境的标准菜单命令
- 在菜单出现前定义好它的位置，避免 UI 的重要部分被遮挡
- 当用户双击呼出菜单时，定义好默认被选中的对象

你不能改变菜单本身的颜色或形状。

如需了解如何在代码中执行这些行为，请参阅《iOS App Programming Guide》中的「Copy and Paste Operation」部分。

为确保你 app 的编辑菜单行为符合用户期望，你应当：

显示与当前情境相符的命令。例如，如果什么都没选中，菜单就不应包含「复制」或「粘贴」，因为这些命令只在选中区域中生效。同样，如果选中了一些东西，菜单里就不应包含「选择」。如果你要在一个自定义视图中支持编辑菜单，要确保菜单中显示的命令和当前情境相符。

让菜单和你的界面布局相适应。iOS 在插入点或选中区域的上方或下方（取决于可用空间）显示编辑菜单，并在此显示菜单指针，以便用户可以看到菜单命令是如何与内容联系起来的。如果需要，你可以在菜单出现前由程序决定好位置，避免 UI 的重要部分被遮挡。

支持两种呼出菜单的手势。尽管触碰并按住的手势是用户呼出编辑菜单最主要的方式，但他们也会在文本视图中双击一个单词去选中它，同时呼出菜单。如果你要在自定义视图中支持菜单，确保能响应这两种手势。此外，你可以定义用户双击时默认选中的对象。

避免在界面中放置一个在编辑菜单中已经存在的命令按钮。例如，比起提供一个「复制」按钮，让用户使用编辑菜单执行复制操作会更好，因为用户会想为什么在你的 app 中会有两种方式去做同样的事情。

如果对用户有益，考虑支持选中静态文本。例如，用户可能想要复制一张图片的标题，但他们不见得想复制条栏标题标签或页面标题，例如「账户」。在文本视图中，应当默认选中一个词。

不要让按钮标题可以选中。一个可选中的按钮标题会让用户很难在不触发按钮的情况下呼出编辑菜单。一般来说，像按钮这样的元素都不应该被选中。

支持复制和粘贴的同时，支持撤销和重做。在用户改变想法时，他们通常期望可以撤销最近的操作。由于编辑菜单在操作被执行前不需要确认，你需要给用户以机会去撤销和重做他们的操作。

如果你需要自定义编辑菜单项，遵循以下这些准则：

创建的编辑菜单要可以编辑，修改或以其他方式直接作用于用户的选中区域。用户希望编辑菜单项可以在当前环境里直接操作被选中的对象，这时你自定义的菜单项最好能有类似的行为。



自定义项目列在所有系统项目之后。不要在系统项目中插入你自定义制的项目。

保持合理的自定义菜单项目数。不要用太多的选择让你的用户崩溃。

为你自定义的菜单项选用简单明了的名字，并确保其准确描述了该命令的作用。通常来说，项目名字应当是描述了所执行动作的动词。通常你应当使用一个大写单词作为项目名字，但如果你必需使用短句，使用标题大写样式。（简言之，标题大写样式是指大写每一个单词，冠词、并列连词和四个字以内的介词除外。）

撤销和重做

用户通过摇晃设备可以发起「撤销」操作，这会显示一个警告框来让用户：

- 撤销他们刚刚输入的内容
- 重做之前未完成的输入
- 取消撤销操作

通过定义以下内容，在你的 app 中以更通用的方式支持撤销操作：

- 用户可以撤销或重做的操作
- 什么时候你的 app 应当把一个摇晃事件看成是摇一摇撤销手势
- 要支持多少步的撤销

如需了解如何在代码中执行这一行为，请参阅「Undo Architecture」。如果你要在你的 app 中支持撤销和重做，遵循以下准则以提供好的用户体验。

用简短的描述性短语告知用户正在撤销或重做的具体内容。 iOS 会自动在撤销警告框的按钮标题上显示「撤销」和「重做」（字词后包含一个空格），但你需要用一两个词来描述用户可以撤销或重做的行为。例如，你可能会有「删除姓名」或「修改地址」这样的文本，那使用诸如「撤销删除姓名」或「重做修改地址」这样的按钮标题。（注意在警告框中的「取消」按钮不可以被修改或者移除。）

避免使用过长的文本。 过长的按钮标题会被截断并且让用户难于理解。由于这是一个按钮标题文本，使用标题大写样式并且不要增加标点。

避免让摇晃手势过于复杂。 虽然你可以在代码中设置何时你的 app 会将一次摇晃事件识别为摇一摇撤销，但如果用户还可以用摇晃执行一个完全不同的操作，这很可能会让用户迷惑。分析用户在你的 app 中的交互行为，避免造成用户不能准确预测摇一摇手势的效果。

只有当撤销和重做是你的 app 中的基本任务时，才使用系统自带的「撤销」和「重做」按钮。 记住，摇晃手势是用户触发撤销和重做的首要方式。如果为执行同一个任务提供两种方式，这会让用户迷惑。如果你认为撤销和重做提供一个明确专用的按钮很重要，你可以在导航栏中放置系统自带的按钮。（了解更多关于按钮的信息，请参阅「[工具栏和导航栏按钮](#)」（第 124 页）。）

将撤销重做和用户当下的情境清晰明确地联系起来，而非之前的情境。 仔细考量那些允许被撤销或被重做的行为场景。通常来说，用户期望他们的改变和操作可以立即生效。

键盘和输入视图

如果合适，你可以设计一个自定义输入视图以取代系统自带的屏幕键盘。

如果你提供了一个自定义输入视图，确保其功能对用户显而易见。

你可以提供一个自定义输入辅助视图，这是分离视图会在键盘（或你的自定义输入视图）上方显示。

当用户在你的输入视图中轻击自定义控件时，请使用标准的键盘点击声音以提供听觉上的反馈。如需了解如何在你的代码中启用这样的音效，请参阅《[UIDevice Class Reference](#)》中 `playInputClick` 相关的文档。

注意：标准的点击声音只在自定义输入视图处于当前页面时可用。用户可以在「设置 > 声音」中关闭所有的按键音（包括你的自定义输入视图中的）。

用户界面元素

- 「条栏」 (第 119 页)
- 「内容视图」 (第 130 页)
- 「控件」 (第 151 页)
- 「临时视图」 (第 164 页)

条栏

状态栏

状态栏 (status bar) 显示了设备和当前环境的重要信息（在 iPhone 上显示如下）。



状态栏：

- 是透明的
- 总是在设备屏幕顶部边缘出现

API 备注：你可以为整个 app 设置统一的状态栏样式，也可以为单独视图设置合适样式。如需了解更多关于 `UIStatusBarStyle` 常量和 `preferredStatusBarStyle` 属性的信息，参阅《UIApplication Class Reference》及《UIViewController Class Reference》。

不要创建自定义状态栏。用户会依赖于系统状态栏的一致性。就算你可能会在 app 中隐藏状态栏，但也不适宜在其出现的地方使用自定义的 UI。

防止让滚动内容透过状态栏显示。当用户滚动页面时，你不会希望他们在状态栏区域将 app 内容和状态栏本身混淆。为了让用户感觉宽敞的同时仍然保证最佳的可读性，要确保状态栏背景会模糊其背后的内容。这里有一些办法让滚动内容可以透过状态栏隐约显现：

- 使用导航控制器去显示内容。导航控制器会自动显示状态栏背景，并确保其内容视图不会在状态栏背后出现。（如需了解更多关于导航控制器的信息，参阅「Navigation Controllers」。）
- 创建一个不醒目的自定义图片——例如渐变——并放到状态栏背后显示。为确保这张图片显示在状态栏背后，你可以使用视图控制器来保证这张图片在滚动视图之上，或者你可以使用滚动视图并让其固定在顶部。
- 定位内容，以避开状态栏区域（即，由 app 的 `statusBarFrame` 定义的区域）。如果你这样做了，你要使用窗口的背景颜色以在状态栏背后提供一个固定的颜色。

避免在状态栏背后放置干扰性内容。特别是，你不能让用户误认为轻点状态栏可以获取内容或触发你 的 app 中的控件。

隐藏状态栏时要慎重考虑。由于状态栏是透明的，所以你通常不需要隐藏它。始终隐藏状态栏，意味着用户必须从你的 app 中切换出去才能看到当前时间，或者知道是否有 Wi-Fi 连接。

在用户全屏观看多媒体内容时，考虑隐藏状态栏——和其他所有界面元素。如果你这样做了，请确保用户再轻点一次即可恢复状态栏（和其他所有界面元素）。如果没有非常充分的理由，则要避免重新定义一个手势来唤起状态栏，因为用户会很难发现并记住这样一个手势。

选择一个和你的 app 相协调的状态栏颜色。默认样式以白底黑字显示，适合用在浅色内容的 app 的顶部。而黑底白字的状态栏则适合放在深色内容的 app 顶部。

适当的时候，显示网络活动指示器。网络活动指示器可以出现在状态栏中，以向用户显示长时间的网络接入状态。如需了解如何在代码中执行实现这样的指示器，请参阅「[网络活动指示器](#)」（第 156 页）。

导航栏

导航栏（navigation bar）能实现在不同信息层级结构之间的导航，有时候也可以管理屏幕内容。



导航栏：

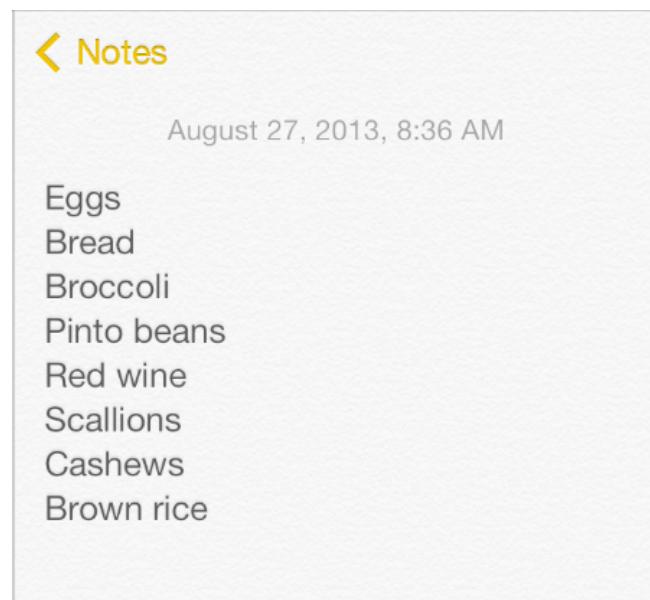
- 是半透明的
- 一般出现在 app 页面顶部，就在状态栏下面一点
- 在 iPad 上，导航栏也可以在视图内部显示而不用贯穿整个屏幕，例如分栏视图控制器的一个窗格。
- 在 iPhone 切换设备方向时，可以自动调整导航栏高度
- 在 iPad 上，导航栏的高度不随设备方向改变而改变

API 备注： 导航栏包含于导航控制器（一个管理显示自定义视图层级结构的程序对象）中。如需了解更多关于如何在代码中定义一个导航栏的信息，请参阅「[Navigation Controllers](#)」和「[Navigation Bars](#)」。

使用导航栏可以在各个视图之间导航，并能提供管理视图中条目的控件（如果合适的话）。如果你需要提供更多的控件而且你不需要启用导航，考虑用工具栏作为替代（如需了解更多信息，请参阅「[工具栏](#)」（第 123 页））。

当用户在导航层级中进入一个新的层级，会发生两个变化：

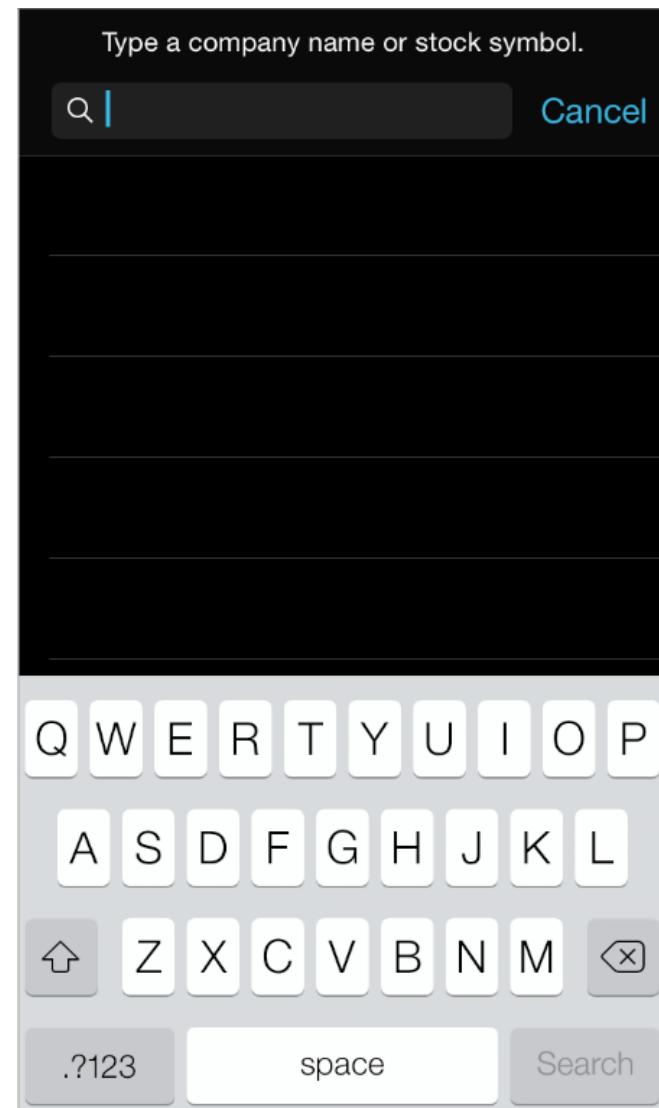
- 导航栏标题可以适时变成新层级的标题
- 在导航栏左侧显示返回按钮，并以上一级页面的标题命名



如果导航栏需要一个标题，那就使用当前视图的标题作为导航栏的标题。如果觉得给导航栏增加标题毫无必要，你也可以让标题留空。例如，「便签」并没有当前便签的标题，因为内容的第一行已经给用户所需的所有情境。

可以考虑在 app 的最顶层导航栏中放置分段控件。这样做会非常有助于扁平化你的信息层级，让人们更容易找到他们想要的东西。如果在导航栏中使用了分段控件，请确保选择一个准确的返回按钮标题。（请参阅「分段控件」（159 页）以了解使用准则。）

必要时，可以考虑用提示语去告诉用户在当前页面可以做些什么。
提示语是一句紧靠着导航栏顶部的简短句子。例如，「股票」用了一个提示语以确保用户知道该如何找到他们想要的信息。



如果你需要使用提示语，那就写一句足够简短并以合适的标点结尾的话。

避免用过多的控件将导航栏挤满，即使看上去还有大量空间。一般来说，导航栏最多应包含当前视图的标题、返回按钮和一个管理视图内容的控件。如果你在导航栏中使用了分段控件，那就不应该显示标题，而且也不应包含除了分段控件之外的任何控件。

确保在文字按钮之间有足够的间隔。如果在导航栏中那些或左或右的按钮之间没有足够间隔，按钮上的文字就会被挤到一块，而这会让用户很难区分它们。如果导航栏的按钮标题看上去太过接近，可以使用 `UIBarButtonSystemItemFixedSpace` 在它们之间增加适当的间距。（如需了解更多信息，请参阅《`UIBarButtonItem` Class Reference》。）

尽可能确保自定义的导航栏外观在整个你的 app 中保持一致。例如，不要将半透明的工具栏和不透明的导航栏混在一起。同样，最好要避免在同一方向的不同页面中使用不同的导航栏图像、颜色或者透明度。

确保自定义的返回按钮看上去仍然是一个返回按钮。用户知道，标准的返回按钮可以让他们在信息层级之间原路返回。如果你决定要用自定义图像替代掉系统自带的返回箭头，请确保这是一个自定义的蒙版图层。iOS 7 会使用蒙版让按钮标题在过渡期为返回箭头期间渐入或渐出。

重要：不要创建多节的返回按钮。通常，返回按钮会将用户带到当前页面的父级页面。如果你认为用户只有在显示一个面包屑路径式的多节控件时才不会迷路，那这多半意味着你需要扁平化你的信息层级。

在 iPhone 上，导航栏高度要随时准备随设备旋转而变化。尤其，要确保你所自定义的导航栏图标可以很好地适应横屏方向上变窄的条栏。不要在代码中写死导航栏的高度，相反，你可以利用 `UIBarMetrics` 常量来确保你的内容可以很好应对变化。

工具栏

工具栏（`toolbar`）包含了对页面或视图中对象进行操作的控件。



工具栏：

- 是半透明的
- 在 iPhone 上，始终在屏幕或视图底部出现
- 在 iPad 上，也可以在屏幕或视图顶部出现
- 在 iPhone 改变设备方向时，可以自动调整工具栏高度

- 在 iPad 上，工具栏的高度不随设备方向改变而改变

API 备注：工具栏通常包含于导航控制器（一个管理一系列自定义视图显示的程序对象）中。如需了解关于如何在代码中定义一个工具栏的更多信息，请参阅《View Controller Catalog for iOS》中的「Displaying a Navigation Toolbar」和「Toolbar」部分。

使用工具栏可以向用户提供一系列当前情境下可用的操作。

包含当前情境下最常用的指令。尽可能避免让工具栏提供一些只会偶尔用到的指令。

考虑使用分段控件，用来切换当前情境中的不同视图或模式。最好不要在工具栏中使用分段控件来显示应用级任务或模式，因为工具栏对应的就是当前页面或视图。如果你需要让用户在你的 app 中接触基本任务、视图或者模态窗口，那使用标签栏作为替代。如需了解关于分段控件的更多信息，请参阅「[分段控件](#)」（第 159 页）；如需了解标签栏的更多信息，请参阅「[标签栏](#)」（第 126 页）。

如果需要在工具栏中放置超过三个项目，请使用图标。由于文字按钮通常会比图标更占空间，因此很难做到让文字标题不挤到一块去。

确保文本标题按钮之间有足够的间隔。如果工具栏中那些或左或右的按钮之间没有足够间隔，按钮上的文字就会被挤到一块，而这会让用户很难区分它们。如果工具栏中的按钮标题看上去太过接近，可以使用 `UIBarButtonSystemItemFixedSpace` 在它们之间增加适当的间距。（如需了解更多，参阅《[UIBarButtonItem Class Reference](#)》。）

在 iPhone 上，工具栏高度要随时准备随设备方向而变化。尤其，要确保你所自定义的工具栏图标可以很好地适应横屏方向上变窄的条栏。不要在代码中写死工具栏的高度，相反，你可以利用 `UIBarMetrics` 常量来确保你的内容可以很好应对变化。

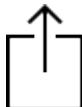
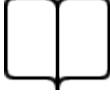
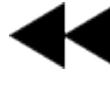
工具栏和导航栏按钮

iOS 提供了大量的标准工具栏和导航栏按钮，这些按钮被用在了内置应用当中。如需了解如何设计自定义的条栏图标，请参阅「[条栏按钮图标](#)」（第 182 页）。工具栏和导航栏上的项目可以使用 `tintColor` 属性着色。

如需了解表 35-1 中符号名称和按键的对应关系，请参阅《[UIBarButtonItem Class Reference](#)》中关于 `UIBarButtonSystemItem` 的文档。

重要：和所有标准的按钮和图标一样，应当根据按钮的含义而不是外观来决定其用途。这样，即便对应着特定含义的按钮改变了样子，你 app 的 UI 仍然可以继续使用。

表 35-1 用于工具栏和导航栏的标准按钮

按钮	名称	含义
	分享 (Share)	打开一个操作菜单，上面列出了针对当前内容、由系统提供或是由 app 自定义的操作服务。
	相机 (Camera)	打开一个操作菜单，在相机模式下显示了一个图片选择器。
	撰写 (Compose)	打开一个处于编辑模式的新消息视图。
	书签 (Bookmarks)	显示针对此 app 的书签。
	搜索 (Search)	显示一个搜索框。
	新增 (Add)	创建一个新项目。
	回收站 (Trash)	删除当前项。
	整理 (Organize)	移动一个项到 app 内的目标位置，例如文件夹。
	回复 (Reply)	发送一个项到另一个位置。
	刷新 (Refresh)	刷新内容（只在必要时使用，尽量自动刷新）。
	播放 (Play)	开始播放多媒体或幻灯片。
	快进 (FastForward)	在多媒体或幻灯片播放过程中快进。
	暂停 (Pause)	暂停多媒体或幻灯片播放（注意，这意味着要保留所处情境）
	快退 (Rewind)	在多媒体或幻灯片播放过程中后退

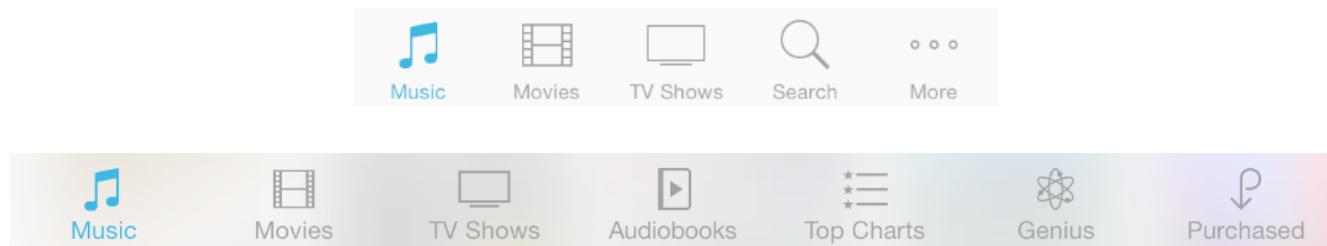
除了以上在表 35-1 中显示的按钮外，你也可以在 app 中使用系统自带的「编辑」、「取消」、「保存」、「完成」、「重做」和「撤销」等按钮，用来支持编辑或其他类型的内容操作。每个按钮的外观由其标题决定。如需了解符号名称和按钮的对应关系，请参阅《UIBarButtonItem Class Reference》中的 UIBarButtonItemSystemItem。

最后，你也可以在工具栏中使用系统自带的信息按钮：



标签栏

标签栏 (tab bar) 赋予了用户在不同子任务、视图和模态之间切换的能力。



API 备注：标签栏包含于标签栏控制器中，这是一个可以管理一系列自定义视图显示的程序对象。如需了解关于如何在代码中定义标签栏的更多信息，参阅「Tab Bar Controllers」和「Tab Bars」部分。

标签栏：

- 是半透明的
- 通常出现在屏幕底部
- 在 iPhone 上，同时显示不超过 5 个标签（如果有更多标签，标签栏会显示其中 4 个并增加一个「更多」标签，再将其他的标签以列表形式收纳在其中）
- 标签栏高度不随设备方向改变而改变
- 可以用徽标——一个红底白字并显示数字或感叹号的椭圆形——来显示和 app 有关的特有信息

使用标签栏可以让用户在同一组数据的不同视图中切换，或是在和 app 整体功能相关的不同子任务中切换。

一般来说，标签栏可以用来在应用层面上组织信息。标签栏非常适合用于 app 的主界面中，因为它可以很好地扁平化信息层级，并同时提供了进入多个不同信息分类或模态的入口。

不要用标签栏来让用户对当前模态或页面上的元素进行操作。如果你需要为用户提供这些操作，你可以使用工具栏作为替代（如需了解使用准则，请参阅「工具栏」（第 123 页））。

在某个标签的功能不可用时，不要移除它。如果某个标签所代表的部分功能在当前情境下不可用了，相比移除这个标签，更好的做法是显示一个禁用的标签。如果你在某些情况下移除了标签栏而其他时候又没有，这会让你 app 的界面变得不稳定且不可预期。最好的解决方案是确保所有标签都可用，但解释标签内容不可用的原因。比如，如果用户的 iOS 设备中没有歌曲，在「音乐」app 的「歌曲」标签中就会显示一个页面去说明如何下载歌曲。

考虑在标签栏上使用徽标以低调地传达信息。你可以在标签栏图标上显示徽标来暗示该视图或模态中有新信息，

在 iPad 上，你可能会在分栏视图窗格或弹出窗口中用到标签栏。如果这些标签是用于切换或过滤视图内容，你就可以这样做。不过，在弹出窗口或分栏视图窗格的底部使用分段控件效果往往更好，因为分段控件的外观和分栏视图或弹出窗口的外观更协调。（如需了解使用分段控件的更多信息，请参阅「[分段控件](#)」（第 159 页）。）

在 iPad 上，避免让过多的标签挤满标签栏。在标签栏放置过多的标签会让用户很难点中他们需要的标签。每多一个标签，你的 app 又多了一分的复杂。通常，尝试将主视图或分栏视图右侧窗格中的标签数量限制在 7 个以内。而在弹出窗口或分栏视图左侧窗格中，不要超过 5 个。

在 iPad 上，避免使用「更多」这一标签。在 iPad app 中，专门用一个页面显示一列多余的标签是很浪费空间的。

在 iPad 上，横屏竖屏都显示一样的标签，以增强 app 的视觉稳定感。竖屏方向时，推荐使用 7 个等分屏幕宽度的标签。横屏方向时，你应该将同样的标签沿屏幕宽度居中。这个准则同样也适用于分栏视图或弹出窗口中的标签栏。例如，如果你在竖屏方向的弹出窗口中使用了一个标签栏，那在切换成横屏时，分栏视图左侧的窗格中也要显示同样的标签。

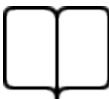
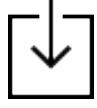
标签栏图标

iOS 为标签栏提供了标准图标，见表 35-2。如需了解如何设计自定义标签栏图标，请参阅「[条栏按钮图标](#)」（第 182 页）。标签栏图标可以使用 `tintColor` 属性着色。

要了解符号名称和按键的对应关系，请参阅《[UIBarButtonItem Class Reference](#)》中的关于 `UIBarButtonSystemItem` 的文档。

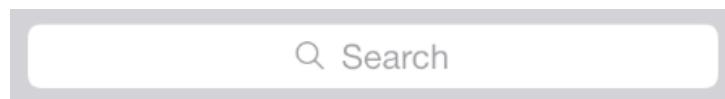
重要：和所有标准按钮和图标一样，你需要根据按钮的语义含义去决定其用途，而不是外观。这会让你 app 的界面符合直觉，即便是相关含义的图标外观被改变了。

表 35-2 用于标签栏的标准标签图标

按钮	名称	含义
	书签 (Bookmarks)	显示此 app 中的书签。
	联系人 (Contacts)	显示联系人。
	下载 (Downloads)	显示下载项。
	收藏 (Favorites)	显示用户的收藏。
	推荐 (Featured)	显示该 app 的推荐内容。
	历史 (History)	显示用户的操作历史。
	更多 (More)	显示额外的标签栏项目。
	最新 (MostRecent)	显示最新的项目。
	最多浏览 (MostViewed)	显示所有用户中最流行的项目。
	最近 (Recents)	显示在 app 规定时间内用户访问的项目
	搜索 (Search)	进入搜索模式。
	最高评分 (TopRated)	显示由用户产生的最高评分项目。

搜索栏

搜索栏 (search bar) 可以接收用户输入的文本并将其作为一次搜索输入（如下图所示）。



API 备注：如需了解如何在代码中定义搜索栏，请参阅「Search Bar」。

搜索栏可以显示一些可选的元素，如：

- 占位符文本。此文本可能会说明这个控件的功能——例如，「搜索」——或者提醒用户在何种搜索情境之下——例如，「Google」。
- 「书签」按钮。此按钮可以给用户提供一种快捷方式，再次抵达他们想要轻松找到的信息。例如，「地图」搜索模式中的「书签」按钮，可以让人访问收藏的位置、最近搜索和联系人。
「书签」按钮只在搜索栏中没有用户输入的或非占位符文本时显示。当搜索栏包含这样的文本时，则显示「清除」按钮以便用户可以清除文本。
- 「清除」按钮。大多数搜索栏都包含清除按钮，用户轻点一下就能清除搜索栏中的内容。

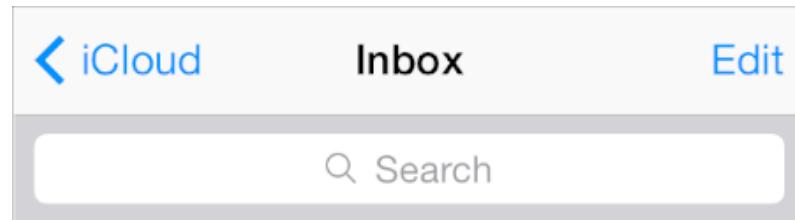
当搜索栏中包含任意文本时，显示「清除」按钮以便用户清除文本。如果搜索栏中没有任何用户输入的或非占位符的文本时，则隐藏「清除」按钮。

- 结果列表图标。此图标用来表示搜索结果。当用户点击这个图标时，app 就显示他们最近搜索的结果。
- 提示语。一个放置在搜索栏中的描述性标题，被称之为提示语。通常来说，提示语是一个简短的句子，以提供引导或搜索栏在 app 中所处的情境。

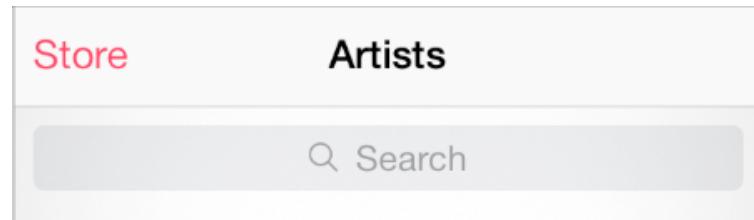
在你的 app 中使用搜索栏以启用搜索。不要将文本框用作搜索，因为它没有用户所期望的标准搜索栏样式。

在 iOS 7 和更高版本中，使用 `UISearchDisplayController` 会在导航栏中放一个搜索栏变得容易。请注意，搜索视图控制器包含于导航控制器中，当用户进入搜索时，搜索栏会自动在导航栏中渐变显现——如同「邮件」这个例子一样。

突显的搜索栏风格（在「邮件」中显示）



精简的搜索栏风格（在「音乐」中显示）



为你的 app 选择符合其重要性的搜索栏风格。如果搜索是你 app 的主要功能，你可能需要使用明显的风格；若用户不会经常用到搜索，你可能需要使用精简风格。

范围栏

范围栏 (scope bar) 随搜索栏一起出现，它允许用户定义搜索范围。



API 备注：如需了解如何在代码中定义搜索栏和范围栏，请参阅「Search Bars」。

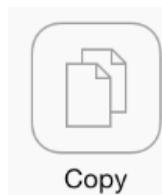
当搜索栏出现时，范围栏也会在其附近出现。范围栏的外观会和搜索栏保持一致。

当用户想要在明确定义或清晰分类的范围内搜索时，范围栏会非常有用。当然，更好的做法是提升搜索结果质量，让用户不需要对搜索进行筛选。

内容视图

活动

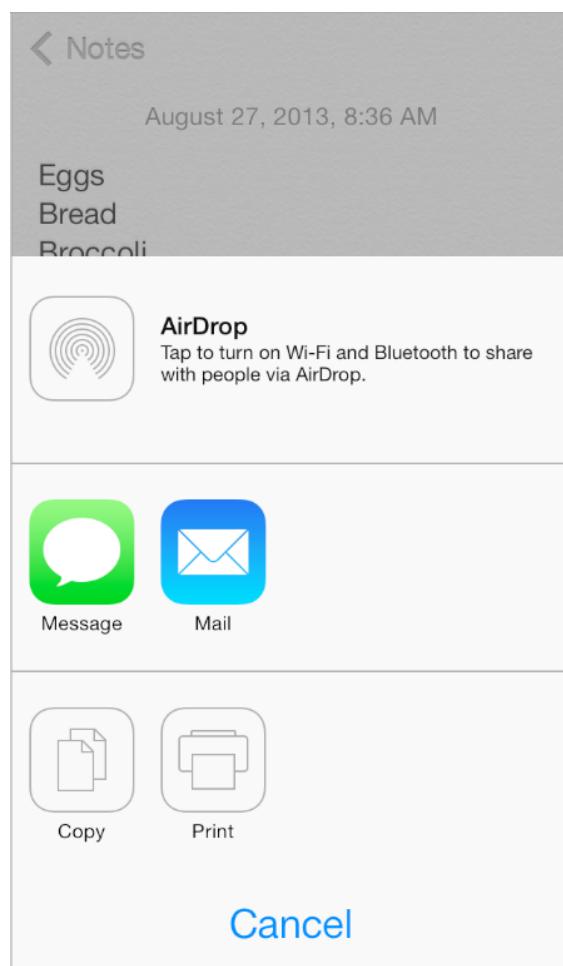
一个**活动** (Activity, 从活动视图控制器中启动) 代表着某个系统提供或自定义的服务操作，它可以作用于某些特定的内容。



API 备注：如需了解如何在代码中定义活动菜单，请参阅《UIActivity Class Reference》；如需了解如何将活动视图控制器融入你的 app，请参阅「[活动视图控制器](#)」（第 132 页）。

活动：

- 是一个可自定义的对象，代表着某个可以让用户在 app 中执行操作的服务
- 以图标形式呈现，类似于一个条栏按钮图标



用户轻点活动视图控制器中的活动图标便可以执行一个服务操作。之后，活动会立即执行该服务，而在服务比较复杂时，它也可以请求请求更多信息再执行该服务。

使用活动能让用户执行你的 app 所提供的自定义服务。请注意，iOS 本身提供了一些内置服务，例如「打印」、「Twitter」、「消息」和「AirPlay」。你不需要再创建一些执行内置服务的自定义活动。

为你的服务创建一个线条流畅的模板图像（template image）。模板图像是一张图片，iOS 会用来生成一个用户最终看到的图标。为了创建一个在最终图标中看上去不错的图像模板，请遵循以下准则：

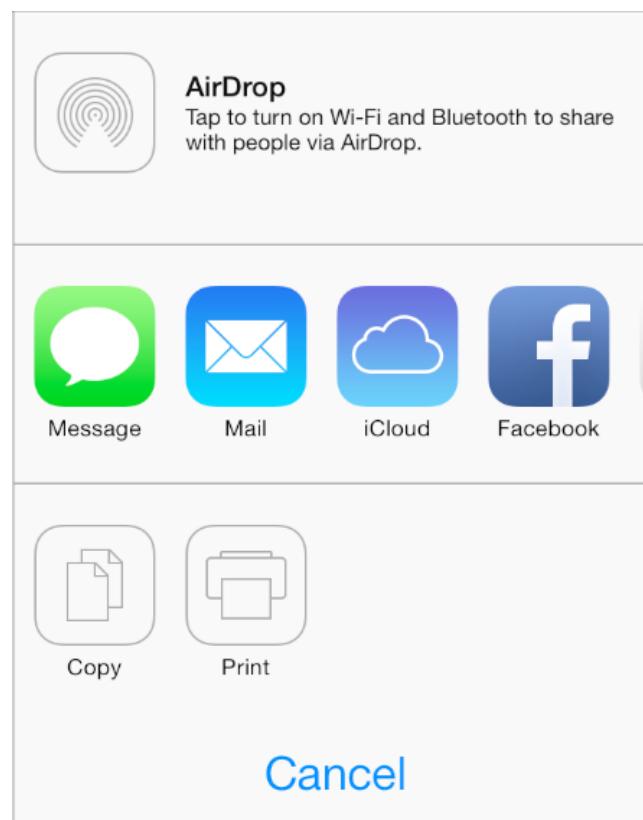
- 使用 alpha 透明度适当的纯黑色或者纯白色
- 不要包含阴影
- 使用抗锯齿处理

一个活动的模板图应在 70 x 70 像素左右（高分辨率下），且在区域内居中显示。

为你的服务撰写一个简练的活动标题。这个标题会在活动视图控制器中活动图标的下方显示。简短的标题通常效果最好，因为它在屏幕上显示效果更好且更容易本地化。如果标题过长，iOS 首先会缩小文本，仍然太长的话则会对其进行截断。一般来说，最好在活动标题中不要提及你的公司或产品名称。

活动视图控制器

活动视图控制器（activity view controller）是一个临时视图，其中列出了针对某些特定内容的系统服务和自定义服务。



API 备注：如需了解更多在代码中定义活动视图控制器的信息，请参阅《UIActivityViewController Class Reference》；如需了解如何设计一个提供自定义服务的活动，请参阅「活动」（第 131 页）。

活动视图控制器：

- 显示了一系列可配置的服务，让用户可以针对指定内容执行操作
- 在 iPhone 上，会在操作列表中出现；在 iPad 上，则会在弹出窗口中出现

使用活动视图控制器可以为用户提供一系列服务，让他们可以以某些方式对特定内容进行操作。这些服务可以是系统自带的（例如「复制」、「Twitter」和「打印」），也可以是自定义的。一种常见的活动视图控制器使用方式是，允许用户将选中的内容发布到某个社交媒体账号中去。

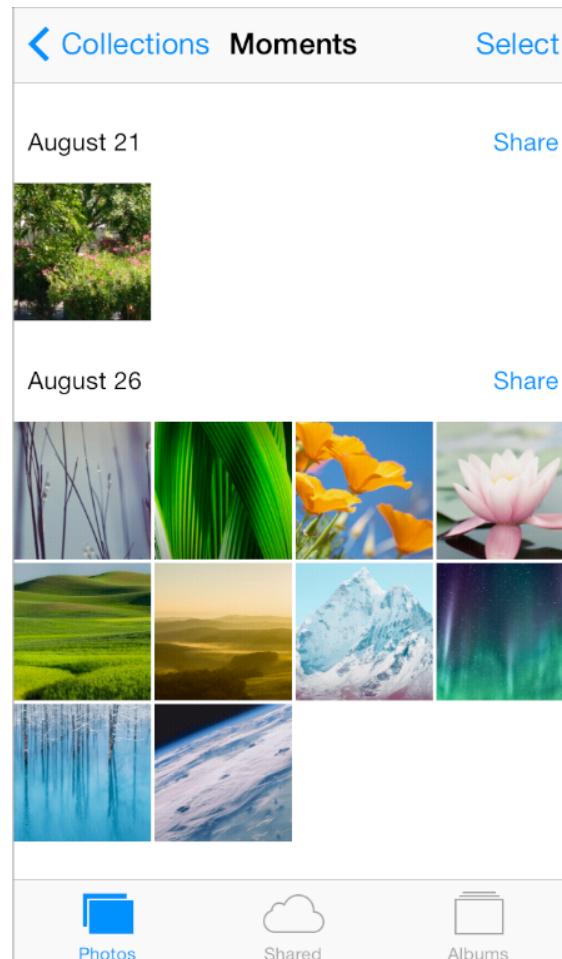
不要创建一个自定义按钮来触发活动视图控制器。人们更习惯于轻点「分享」按钮后直接进入系统自带的分享服务。你可以直接利用这一习惯，而不是为完成同样事情提供另一种方法来混淆用户。

确保列出的服务适用于当前情境。你可以更改活动视图控制器中所列出的服务，删减一些系统提供或增加一些自定义服务。例如，为了防止用户打印图像，你可以将「打印」活动从活动视图控制器中删除。

注意：你不能更改活动视图控制器中系统服务的顺序。同样，所有的系统服务都会在自定义服务的上方。

精选视图

精选视图 (collection view) 可以管理项目的有序集合，并以一种可自定义的布局呈现。



API 备注：如需了解在代码中如何定义精选视图，请参阅《Collection View Programming Guide for iOS》。

精选视图：

- 可以包含可选视图，以从视觉上区分项的子集或者提供装饰性项目，例如一些自定义背景
- 在布局之间支持自定义的转场动画效果（默认情况下，当用户在精选视图中插入、移动或者删除项目时，都会有动画效果）
- 支持额外的手势识别以执行自定义行为。默认情况下，精选视图可以识别轻点（以选中一个项目）和长按（以编辑一个项目）。

使用精选视图来让用户查看和操作一系列不适合以列表呈现的项目。由于精选视图并不是一个严格的线性布局，因此它特别适合显示那些尺寸不一的项目。

精选视图支持广泛的定制，因此必须要保持专注，避免被你创造全新设计的能力引入歧途。你要让精选视图来提高用户的任务体验；而非让视图本身成为用户体验的焦点。下面这些准则有助于你去创建用户所喜欢的精选视图：

当表格视图是更好的选择时，不要使用精选视图。有时候，以列表呈现的信息会更容易被用户所阅读和理解。例如，当文本信息在滚动列表中呈现时，人们可以更简单更高效地浏览并与之交互。

要让用户更容易选中一个项。如果用户很难点中精选视图中的项，他们是不太可能会沉浸于你的 app 的。和所有用户想要点击的 UI 对象一样，请确保精选视图中每一个项的最小点击区域有 44×44 点。

如果你要让布局动态变化，请务必谨慎。精选视图允许你在用户浏览以及与项目交互时改变其布局。如果你决定要动态地调整精选视图的布局，请确保其中的变化是有意义且容易被用户理解的。如果缺乏明显动机就去改变视图布局，这会让用户对你的 app 留下不符合预期且难以使用的印象。如果用户在动态布局变化中丢失了当前关注点或所处情境，用户会认为你的 app 彻底失控了。

容器视图控制器

容器视图控制器 (container view controller) 以自定义的方式管理和呈现一系列子视图（或子视图控制器）。由系统定义的容器视图控制器一些例子是标签栏视图控制器、导航视图控制器和分栏视图控制器（你可以在「[标签栏](#)」（第 126 页）、「[导航栏](#)」（第 121 页）和「[分栏视图控制器（仅 iPad）](#)」（第 142 页）中了解这些元素的信息）。

API 备注：如需了解在代码中自定义内容视图控制器的信息，请参阅《UIViewController Class Reference》。

容器视图控制器没有预先定义好的外观和行为。

使用容器视图控制器可以让用户以自定义方式进行导航。

问问自己，是不是必须要用到自定义的容器视图控制器。用户习惯于标准容器视图控制器的外观和行为，比如分栏视图控制器和标签栏视图控制器。你需要确保自定义容器视图所带来的潜在优势，远远大于用户不认识或不知道如何立刻上手的情况。

确保你自定义的容器视图控制器能在所有方向上都能工作。在竖屏和横屏方向上都要给予用户一致的体验，这对于设计容器视图控制器来说至关重要。

一般来说，避免华而不实的视图转场动画。如果你使用故事板来设计一个自定义视图控制器，可以很容易地为内容视图的转场效果自定义动画。但在大多数情况下，太过花哨的视图转场效果会让用户从任务中分心，并会降低你的 app 的审美情趣。

图像视图

图像视图 (image view) 可以显示一张图片或者一系列动态图片。

API 备注：如需了解在代码中定义图像视图的更多信息，请参阅「[图像视图](#)」。

图像视图：

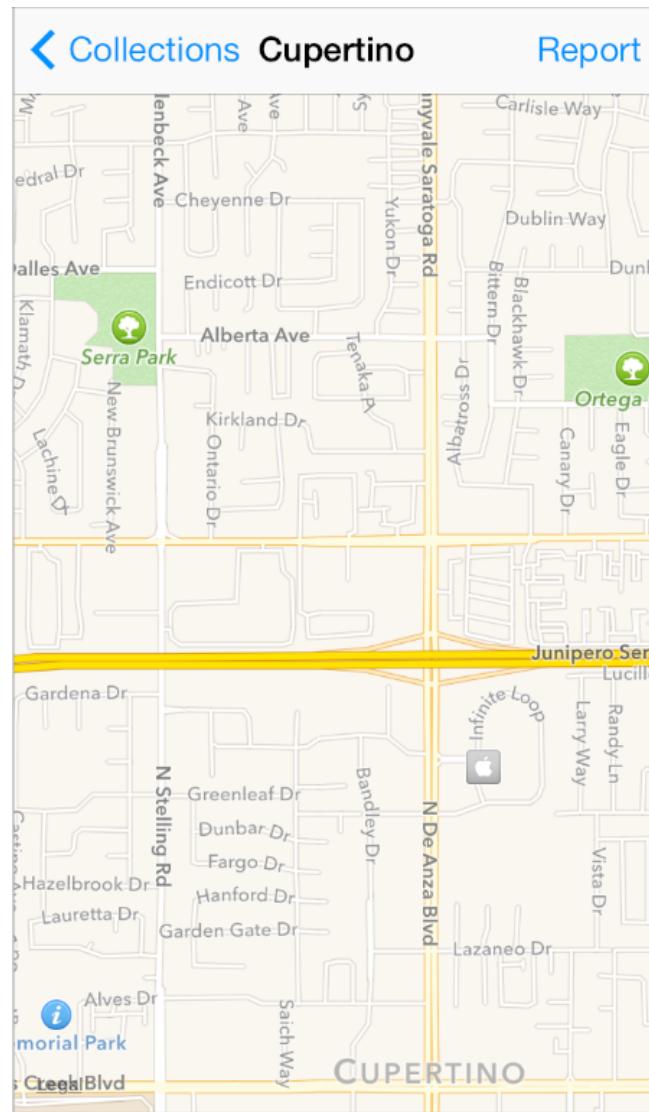
- 没有预先定义好的外观，而且在默认情况下不支持用户的交互行为
- 检查图像及其父级视图的属性，并决定图像是否可以被拉伸、缩放、调整到合适大小，或者固定到一个特定位置

在 iOS 7 中，包含了模板图像的图像视图会将当前着色应用到图片中去。

尽可能确保图像视图中所有的图像都使用相同的尺寸和比例。如果你的图像尺寸各不相同，图像视图将会逐一对其调整；如果你的图像比例不一，它们可能会无法正确呈现。

地图视图

地图视图（map view）可以呈现地理数据，并支持内置「地图」app 的大部分功能（在「iPhotos」中显示如下）



API 备注：如需了解在代码中定义地理视图的更多信息，请参阅「Map Kit Framework Reference」。

地图视图：

- 使用标准地图数据、卫星图像或两者结合来显示一个地理区域
- 可以显示备注（以单点标注）和叠加图层（绘制路径或二维区域）
- 同时支持程序和用户所控制的放大和平移

利用地图视图会给用户提供一个可交互的地理区域视图。如果你在开发一个导航类 app，可以使用地图视图来显示用户的路线（如需了解关于创建导航 app 的更多信息，请参阅「[路线导航](#)」（第 86 页））。

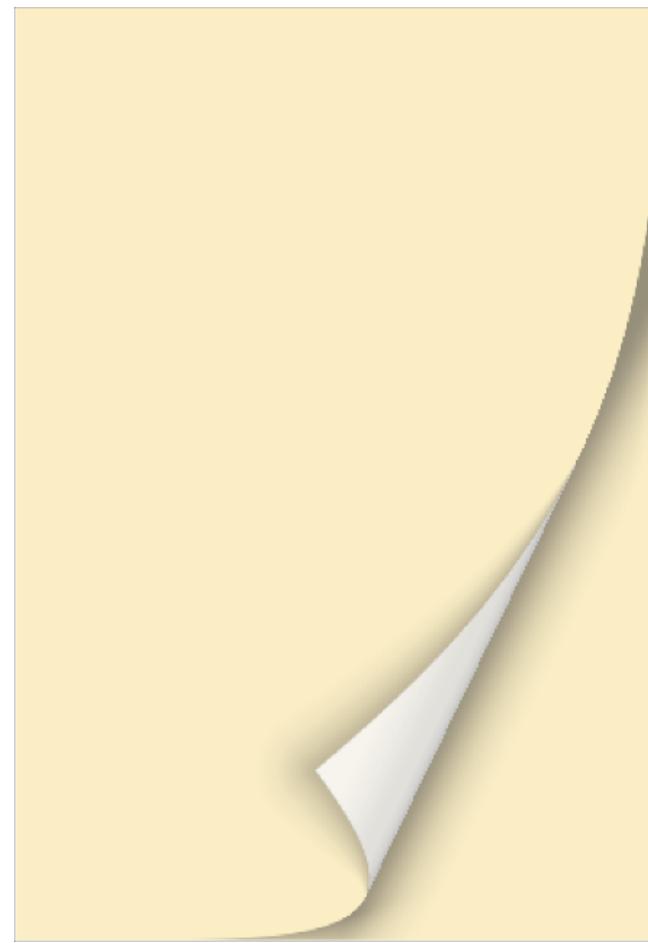
一般来说，要允许用户和地图交互。人们习惯于和内置「地图」app 进行交互，因此在你的地图中他们也希望以类似方式与之交互。

以一致的方式使用标准的图钉颜色。地图图钉用来显示在你地图中的兴趣点（POI）。人们非常熟悉内置「地图」app 中的图钉颜色，所以最好避免在你的 app 中重新定义这些颜色的含义。当你使用标准图钉颜色时，确保其遵循以下准则：

- 用红色表示目的地
- 用绿色表示起始点
- 用紫色标识用户指定地点

页面视图控制器

页面视图控制器（page view controller）通过滚动或翻页两种样式来管理多个页面内容之间的切换。



API 备注：如需了解在代码中定义页面视图控制器的更多信息，请参阅「[页面视图控制器](#)」。

页面视图控制器：

- 对于滚动样式，没有默认外观
对于翻页样式，页面视图控制器可以在两个页面之间增加书脊效果
- 根据指定的样式，用对应的动画从一个页面切换到另一个页面

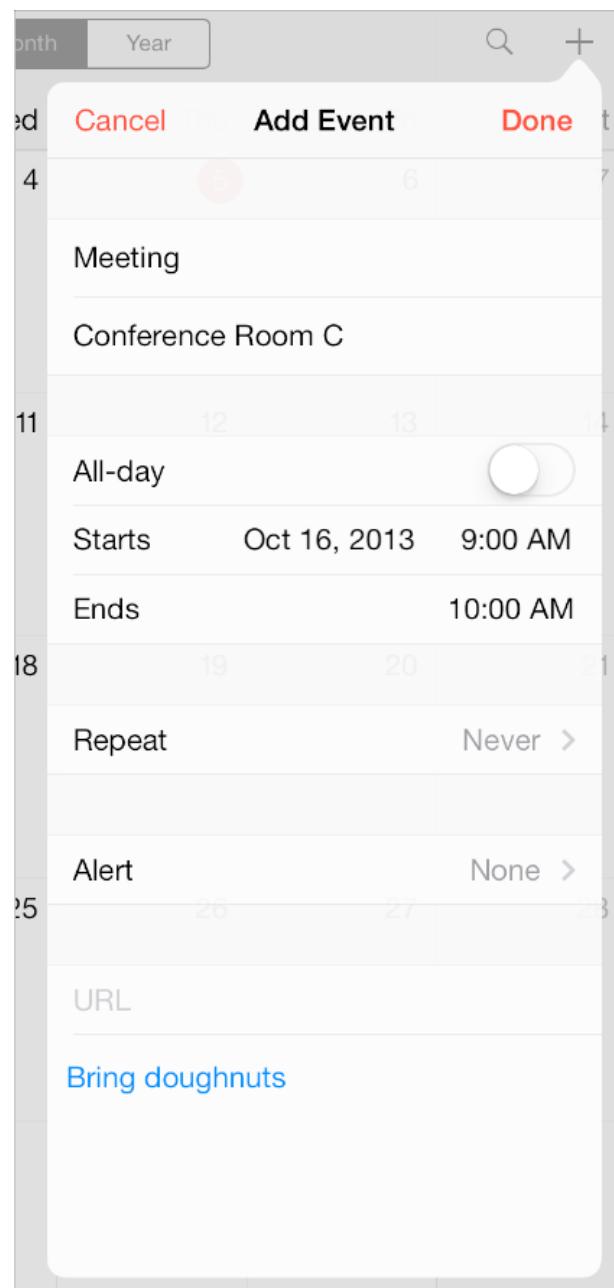
使用滚动样式时，当前页面会滚动到下一页；使用翻页样式时，当前页面会像一本书或笔记本那样翻页。

使用页面视图控制器可以呈现那些线性的内容（如故事文本），或者是可以很自然地划分成块的内容（比如日历）。

如果必要，创建一个自定义方式以允许用户用非线性的方式访问内容。页面视图控制器让用户从一个页面移动到前一页或后一页；但这并不能让用户在不相邻的页面之间快速切换。如果你想用页面视图控制器来显示那些用户可能会用非线性方式访问的内容（如一本词典或一本书的目录），那么你必须自定义一种方式，让用户可以切换到不同的内容区块。

弹出窗口（仅 iPad）

弹出窗口 (popover) 是在用户轻点一个控件或屏幕上的一个区域时出现的临时视图。



API 备注：如需了解关于在代码中定义弹出窗口的更多信息，请参阅《UIPopoverController Class Reference》和「Popovers」。

重要：弹出窗口仅在 iPad app 中可用。

弹出窗口：

- 是一个悬停在屏幕内容上的自包含视图
- 通常显示一个箭头，以标识其浮现的位置
- 有着半透明背景，会模糊其背后的内容
- 可以包含多种对象和视图，如：
 - 表格、图像、地图、文本、网页或自定义视图
 - 导航栏、工具栏或标签栏
 - 可以操作当前 app 视图中的对象的控件或对象

（默认情况，弹出窗口中的表格视图、导航栏和工具栏会使用透明背景，以让弹出窗口的毛玻璃效果显示出 来。）

在 iPad app 中，操作列表通常在弹出窗口内部显示。

弹出窗口可以用来：

- 显示附加信息，或者与焦点、选中对象相关的一系列项目
- 显示一个操作菜单，菜单中包含了几个与屏幕对象紧密相关的选项
- 当一个基于分栏视图的 app 处于竖屏时，弹出窗口用来显示左侧窗格的内容。如果能做到这一点，那你可以提供一个恰当的标题按钮用来显示弹出窗口——按钮最好放在屏幕顶部的导航栏或工具栏中——或者允许用户用轻扫的手势隐藏或显示它。

避免提供「取消弹出窗口」按钮。当不再需要显示弹出窗口时，它就应该自动关闭。要决定弹出窗口什么时候不再显示，请考虑如下场景：

通常，在用户点击弹出窗口范围以外区域时保存他们的工作成果。不是每一个弹出窗口都会有一个明确的取消确认，所以人们可能会误点。只有在人们轻点「取消」按钮时，才丢弃他们在弹出窗口中的工作成果。

尽可能直接地让弹出窗口的箭头指向其浮现的位置。这样做有助于用户记住弹出窗口的出现位置，以及和哪些任务相关。

确保用户在使用弹出窗口时看不到背后的 app 内容。弹出窗口会模糊其背后的内容，而且人们还不能将弹出窗口移到其他位置。

确保同一时间内屏幕上只有一个弹出窗口。你不应该同时展示超过一个的弹出窗口（或外观和行为被设计成弹出窗口的自定义视图）。特别是，你应当避免同时显示一串或一系列弹出窗口，这会从一个窗口中弹出另一个窗口。

如果弹出窗口...	那就这样做...
提供了可以影响主视图的选项，但又不是一个检查器	人们一做完选择或一轻点窗口范围外的任意区域，就关闭弹出窗口，包括显示在窗口中的控件。
作为检查器使用	人们一轻点窗口外任意区域，就关闭弹出窗口，包括显示在窗口中的控件。 在这个场景中，在人们做出选择时不要关闭弹出窗口，因为他们可能是想进行额外选择或改变当前选择的属性。
开启一个任务	当人们轻点弹窗窗口中的一个按钮（如「完成」或「取消」）来完成或取消任务时，关闭弹出窗口。 在这个场景中，当人们点击窗口范围外的区域时，你可能不会想要弹出窗口关闭，因为让人们完成（或彻底放弃）任务可能更为重要。否则，在他们点击弹出窗口范围之外的区域时，保存用户输入的内容，就像你会在他们点击「完成」时所做的那样。

不要在弹出窗口上再展示一个模态视图。除了警告框，弹出窗口之上不应该显示任何东西。

如果可能，允许用户轻点一下以关闭一个窗口并打开一个新的弹出窗口。当不同的几个条栏按钮都能打开弹出窗口时，这个行为显得尤其有用，因为这会减少用户额外的点击。

避免将弹出窗口弄得很大。弹出窗口不应占据整个屏幕。相反，它的大小应恰好足够显示其中的内容，并仍然指向其出现的位置。

理想情况下，弹出窗口的宽度至少要有 320 点，但不能大于 600 点。弹出窗口的高度则没有限制，所以你可以用来显示一段很长的项目列表。但通常，你要尽量避免需要滚动弹出窗口才能开启一个任务或者显示完操作列表。请注意，系统可能会调整弹出窗口的高宽以确保其良好适应屏幕。

在弹出窗口中使用标准的 UI 控件和视图。通常来说，在弹出窗口中使用标准控件和视图看上去会最好，用户更容易理解。

确保自定义的弹出窗口看上去仍然是一个弹出窗口。尽管使用 UIPopoverBackgroundView API 可以很容易地自定义大部分弹出窗口的视觉效果，但要避免创建用户不会将其认为是弹出窗口的新设计。如果你过多改动弹出窗口的外观，用户便不能依赖先前的体验来帮助他们理解在你的 app 中该如何使用这个窗口。

如果合适，在改变弹出窗口大小时保持窗口可见。如果你要用弹出窗口来显示同一信息的精简和拓展视图，你可能需要改变弹出窗口的大小。当你调整可见窗口的大小时，使用动画作为转场效果往往是一个不错的主意，因为这不会让人觉得是一个新的弹出窗口取代了原来的窗口。

滚动视图

滚动视图 (scroll view) 能让人们浏览比滚动视图区域更大的内容（下面显示的图像要比滚动视图区域更高更宽）。



API 备注：如需了解关于在代码中定义滚动视图的更多信息，请参阅「Scroll Views」。

滚动视图：

- 没有预先定义的外观
- 当其初次显现或者用户与之交互时，短暂闪烁滚动条
- 以人们感觉自然的方式响应手势的速度和方向，并随之呈现内容
- 当用户在滚动视图中拖拽内容，内容会随之滚动；当用户快速滑动屏幕，内容要在滚动视图中迅速显示，直到用户触摸屏幕或者在内容到达底部时才停止滚动。
- 可以在分页模式 (paging mode) 中运行，通过拖拽或滑动手势来浏览 app 定义好的一页内容

使用滚动视图可以让人们在有限空间内浏览大尺寸或大量的视图。

适当地支持缩放操作。如果缩放对你的 app 有用，那就让用户可以通过双指开合或双击去放大和缩小滚动视图。若你支持了缩放，你还应该设定一个符合用户任务情境的最大和最小缩放范围。例如，如果允许用户放大文字直到被一个字符填满屏幕，这可能会让用户很难阅读内容。

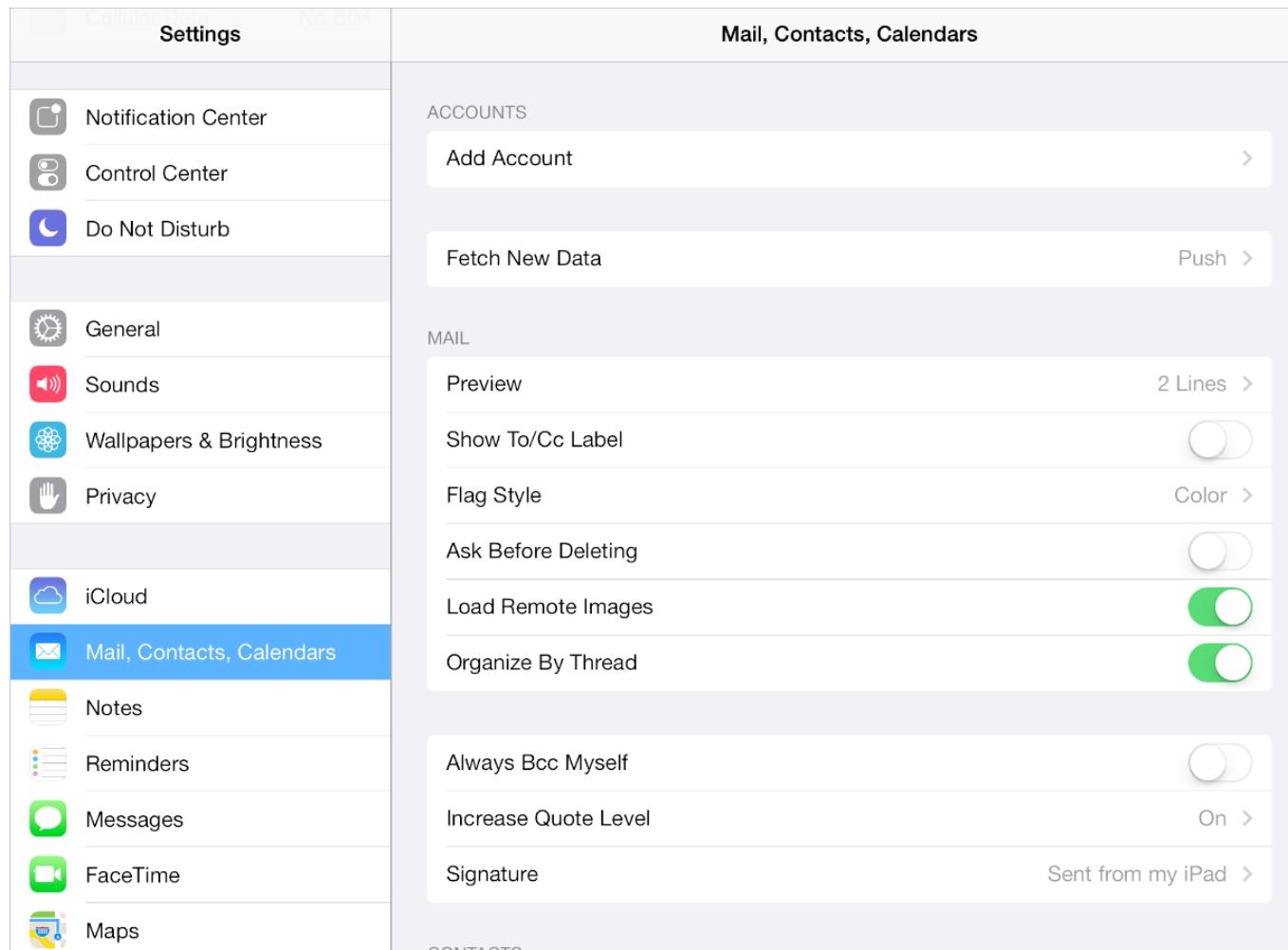
可以考虑在分页模式滚动视图中使用页码控件。如果你想将内容分页、分屏或分块显示，你可以使用页码控件，来向用户显示有多少可用的分块以及他们当前浏览到哪里了。

当你在分页模式的滚动视图中使用页码控件时，最好禁用和页码控件同一方向的滚动条。去除滚动条会让用户聚焦到页码控件上，并让用户有了一种明确的方式去翻阅内容。如需了解关于在 app 中使用页码控件的更多信息，请参阅「[页码控件](#)」（第 173 页）。

通常来说，一次只显示一个滚动视图。人们常常会在滚动时做出大幅度的轻扫手势，所以他们很可能会误点在同一屏中的相邻视图。如果你确定要在同一屏中放置两个滚动视图，可以考虑让它们以不同方向滚动，这样以来同一个手势才不太可能会让两个视图同时滚动。比如竖屏的 iPhone 版「股票」，在以横向滚动视图显示的公司特定信息之上，会以垂直滚动视图显示股票摘要。

分栏视图控制器（仅 iPad）

分栏视图控制器（split view controller）是一个用于管理两个相邻视图控制器显示的全屏视图控制器。



API 备注: 分栏视图控制器中的每一个子视图都负责管理一个窗格的显示。分栏视图控制器本身显示着这些子视图控制器，并对不同方向之间的切换进行管理。如需了解在代码中定义分栏视图控制器的更多信息，请参阅《UISplitViewController Class Reference》和「分栏视图控制器」。

重要: 分栏视图控制器仅在 iPad app 中可用。

分栏视图控制器：

- 会显示两个窗格（左侧窗格的宽度在所有方向上都固定为 320 点；右侧窗格的宽度你可以自己定义）
- 当设备处于横屏方向时，你可以选择让左侧窗口以弹出窗口方式显示
- 可以包含广泛的对象和视图，如：
 - 表格、图像、地图、文本、网页或自定义视图
 - 导航栏、工具栏或标签栏

注意: 即使左侧窗格通常被称作主窗格，右侧窗格是详情窗格，但在代码中并没有将这种关联固定。

使用分栏视图控制器可以在左侧窗格中显示持久信息，在右侧窗格中显示相关详情或从属信息。在这个设计范式中，如果人们选中左侧窗格中的某个项目，右侧窗格中应该显示与之相关的信息。（你需要负责在代码中实现这个效果。）

避免创建比左侧窗格更窄的右侧窗格。如果右侧窗格比左侧窗格窄，分栏视图控制器将不能占满屏幕宽度，这样整个外观会很不平衡。

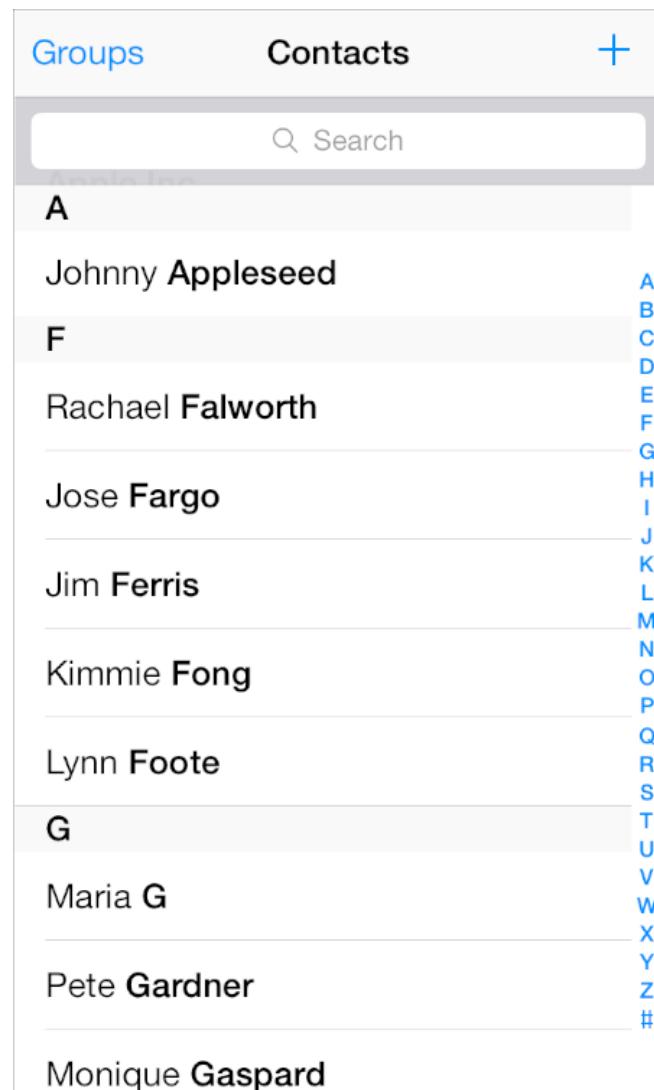
避免在两侧窗格中显示同时显示导航栏。这样做会让用户难以分清两者之间的关系。

通常来说，要始终显示左侧窗格中当前选中的项。即使右侧窗格中的内容会发生改变，但通常要和左侧窗格的选中项目保持相关性。这样的视图体验有助于用户理解左侧窗格项目和右侧窗格内容的关系。

如果合适，要提供替代方式以进入左侧窗格。默认情况下，竖屏方向时只会显示右侧窗格，因此你需要向用户提供一个按钮（通常位于导航栏）用来呼出和隐藏左侧面板。分栏视图控制器也支持轻扫手势来执行呼出/隐藏行为。除非你的 app 会将轻扫手势用来执行其他功能，否则你应当支持用户轻扫以进入左侧窗格。

表格视图

表格视图 (table view) 以一个单列滚动、多行的列表来呈现数据。



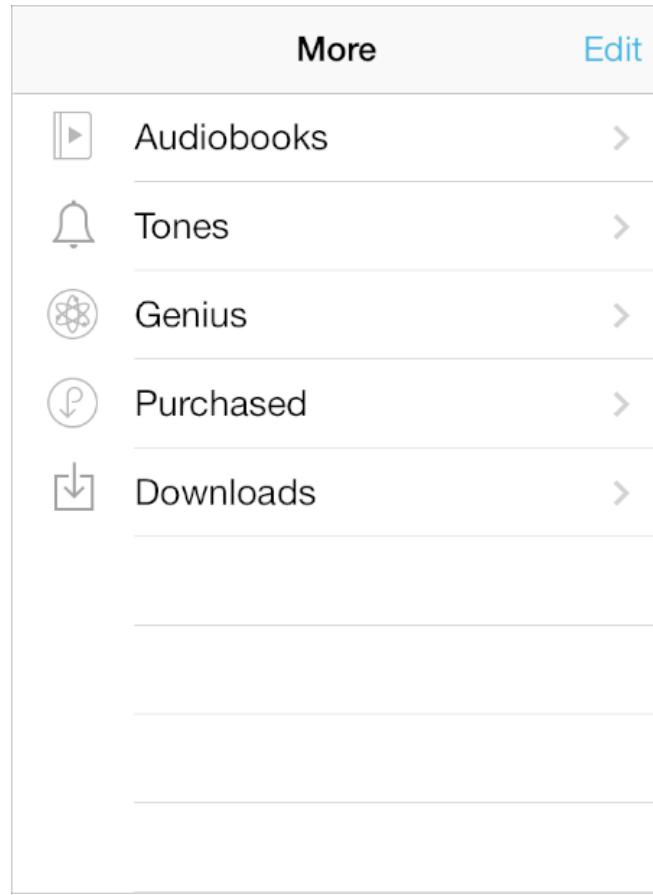
API 备注: 如需了解在代码中定义表格视图的更多信息, 请参阅《Table View Programming Guide for iOS》和「表格视图」。

表格视图:

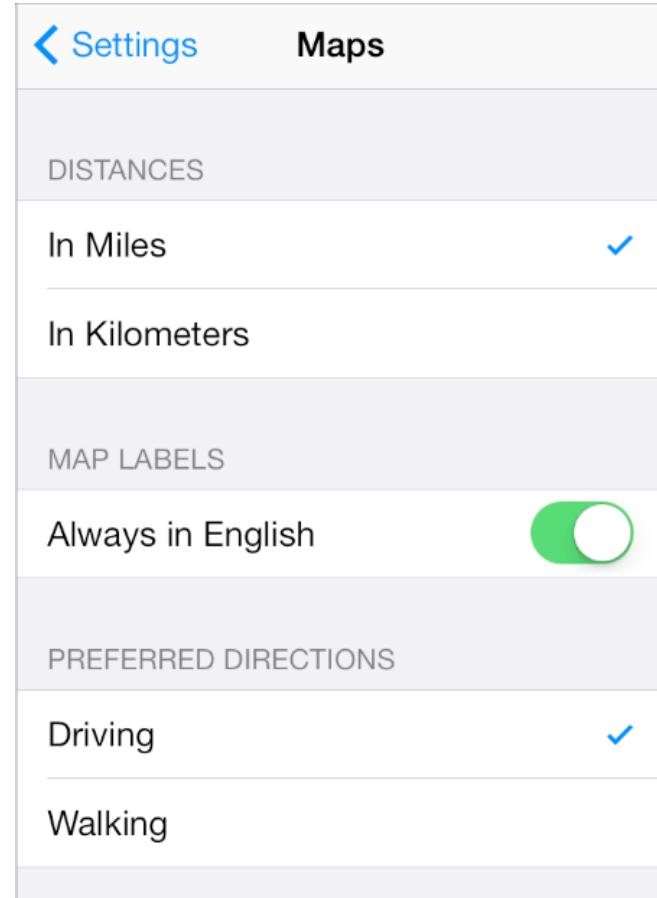
- 以不同的行来显示数据, 这些行可以被分成不同的部分或不同的组类
- 提供控件来让用户添加、删除或多选, 查看某行条目的更多信息, 或者显示另一个表格视图

iOS 定义了两种表格视图样式:

平铺型。在平铺型样式中，行可以被分为若干带标签的段落，而在视图右侧可能会出现垂直排列的索引符号。页眉会显示在这一节的首个条目前，页脚则显示在最后一个条目之后。



分组型。在分组型样式中，行会以多个分组显示，每一组以页眉开始页脚结束。分组型表格视图一般至少包含一组列表项（每一条目是一个列表项），每一组通常至少包含一项内容。分组型表格视图没有索引符号。



在这两种样式中，表格的行在用户点选时都会被短暂高亮。如果选中某一行会进入一个新页面，被选中的行会被短暂高亮，然后一个新页面随之渐渐进入。当用户返回先前的页面时，之前选中的那一行会再次短暂高亮，以提醒用户之前的选项（但并不会一直高亮）。

iOS 包含了一些可以拓展表格视图功能的元素。除非特别注明，这些元素仅适用于表格视图。

表格视图元素	名称	含义
	勾选标记 (Checkmark)	表示此行可以被选中。
	展开指示器 (Disclosure indicator)	展现和此行相关的另一个表格。
	详情展开按钮 (Detail Disclosure button)	在新视图中展现关于此行的更多信息（关于如何在表格区域外使用这个元素，请参阅「弹出窗口 (仅 iPad)」(第 138 页)）。
	行重排按钮 (Row reorder)	表示此行可以被拖拽到表格中的其他位置。
	插入行 (Row insert)	新增一行到列表中。
	删除按钮控件 (Delete button control)	在编辑情境中，显示和隐藏一行的「删除」按钮。
	删除按钮 (Delete button)	删除此行。

除了上面列出的表格专用元素，iOS 还定义了刷新控件，让用户可以用来刷新表格的内容。如需了解更多在你的 app 的表格中使用刷新控件的信息，请参阅「刷新控件」(第 158 页)。

iOS 定义了四种在平铺型和分组型表格布局中最常用的单元格样式。每种单元格样式都有着最适合展示的信息类型。

注意：从编程角度来说，这些样式会应用到表格视图的单元格中，用来告诉表格如何绘制这一行。

Default Cell Style	
	Text Label
	Dahlia
	Daisies
	Dandelion
	Echinacea
	Lavender

默认型 (`UITableViewCellStyleDefault`)。默认型单元格样式包括一个在行左端的图像（可选），和一个在图像右边左对齐的文字标题。

默认样式适合用来显示一系列无需辅助信息便可以区分的列表项。

副标题型 (`UITableViewCellStyleSubtitle`)。副标题样式包括一个在行左端的图像（可选），和一个在图像右边左对齐的文字标题，以及在标题下面左对齐的副标题。

左对齐的文本标签让列表易于浏览。这种单元格样式适用于各列表项目看上去都很相似的情况，这样用户可以用副标题中的详细信息来帮助区分以文本标签命名的项目。

Subtitle Cell Style	
	Text Label Detail text label
	Dahlia This is a dahlia
	Daisies These are daisies
	Dandelion This is a dandelion
	Echinacea This is echinacea
	Lavender

Value 1 Cell Style	
	Text Label Detail text label
	Dahlia This is a dahlia
	Daisies These are daisies
	Dandelion This is a dandelion
	Echinacea This is echinacea
	Lavender This is a field of lav...

Value 1 (`UITableViewCellStyleValue1`)。Value 1 样式会显示一个左对齐的标题，一个右对齐的副标题会在同一行中以更浅的字体显示。

Value 2 (`UITableViewCellCellStyleValue2`)。Value 2 样式以蓝色字体显示右对齐的标题，在同一行中紧接着的是一个黑色字体左对齐的副标题。这种样式通常不包含图像。

在 Value 2 布局中，文本和详情文本之间明确的垂直边距会让用户聚焦于详情文本标签的第一个字。

Value 2 Cell Style	
Text Label	Detail text label
Dahlia	This is a dahlia
Daisies	These are daisies
Dandelion	This is a dandelion
Echinacea	This is echinacea
Lavender	This is a field of lavender

注意：所有的四种标准表格单元格样式均允许添加表格视图元素，比如勾选标记或详情指示器，但添加这些元素会缩小单元格中标题和副标题可用的宽度。

使用表格视图可以清晰而高效地显示少量或大量的信息。例如：

- 提供一系列用户可以选择的选项。**你可以用勾选标记来向用户显示当前列表中可以选中的选项。
无论是平铺型还是分组型表格视图，用户轻点表格某一行中的某个项目都可以显示一个选项列表。当用户轻点一个不属于表格任意行的按钮或其他 UI 元素时，可以使用平铺型表格视图来展示选项列表。
- 展示层级信息。**平铺型表格样式非常适合展示层级信息。每个列表项都可以指向另一个显示不同子信息的列表。用户可以沿着这一系列列表中的选项来浏览层级信息。展开指示器 (disclosure indicator) 告知用户轻点这一行的任何位置，都会展开一个新的列表来显示子类信息。
- 展示可以从概念上分组的信息。**这两种表格视图样式都允许你通过在不同信息段落上放置页眉和页脚来提供情境。

在 iOS 6.0 及以上版本中，你可以使用页眉-页脚视图 (header-footer view) —即 `UITableViewHeaderFooterView` 中的一个常量—来在页眉和页脚中显示文本或自定义视图。如需了解如何在代码中使用页眉-页脚视图，请参阅《`UITableViewHeaderFooterView Class Reference`》。

当你在使用表格视图时，请遵循以下准则：

在用户选择一个列表项时，始终给予反馈。当用户在轻点某个可选中的项目时，他们会希望这个表格行应该有短暂的高亮。在轻点后，用户期望显现一个新的视图（或者在这一行显示一个勾选标记）以表明这个项目已经被选中或激活。

如果表格内容庞大而复杂，不要等到所有数据都加载完了才有所显示。相反，可以一开始就在屏幕上的行中显示文本数据，图像等更多较为复杂的数据则在加载完后再显示。这样做可以让用户立即看到有用的信息，同时也提升了 app 可以被感知到的响应能力。

在等待新数据加载时，可以考虑显示「过期」数据。尽管并不推荐在那些需要频繁处理数据更新的 app 中这样做，但它可以帮助更多的静态 app 立即给到用户一些有用的信息。在你决定这样做之前，请评估数据更新频率，以及用户有多需要立刻看到最新数据。

如果数据加载缓慢或者很复杂，需要告诉用户加载正在处理当中。如果表格包含的信息都很复杂，可能会很难马上显示任何有用的信息。在这种极端情况下，避免显示空白行非常重要，因为这会让用户以为你的 app 已经挂了。相反，应该在屏幕中央显示一个滚动的活动指示器，配上如「加载中」这样的信息标签（informative label）。这会让用户知道加载正在处理当中。

如果合适，为「删除」按钮自定义一个标题。如果能让用户更好地理解 app 的运作方式的话，你可以创建一个标题，以替代系统自带的「删除」字样。

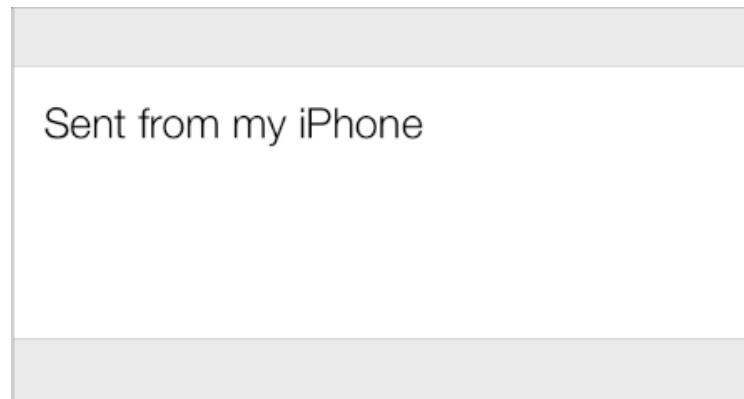
尽量使用简洁的文本标签，以避免被截断。被截断后的文字和词语会让用户难以阅读和理解。所有类型的表格单元格样式都会自动截断文本，这样的截断或多或少会带来一些问题，而这取决于你所使用单元格样式以及截断所发生的地方。

避免同时显示索引和在表格右侧的视图元素。显示在表格右侧的表格视图元素——如详情指示器——会和索引相冲突。

如果你想以非标准的形式来布局你的表格，请自定义一种表格单元格样式。和大幅改动标准样式相比，更好的办法是创建一个自定义的单元格样式。如需了解如何创建你自己的单元格，请参阅《Table View Programming Guide for iOS》中的「Customizing Cells」。

文本视图

文本视图（text view）可以容纳并显示多行文本。



API 备注：如需了解在代码中定义文本视图的内容，请参阅「Text Views」。

文本视图

- 是一个任意高度的矩形
- 当内容超出视图边框太多时，支持滚动
- 支持自定义字体、颜色和对齐（默认情况下，文本视图会以左对齐的黑色系统字体显示）

- 可以支持编辑，当用户在文本视图内轻点时，将会唤起键盘（键盘的输入方式和布局取决于用户的语言设置）

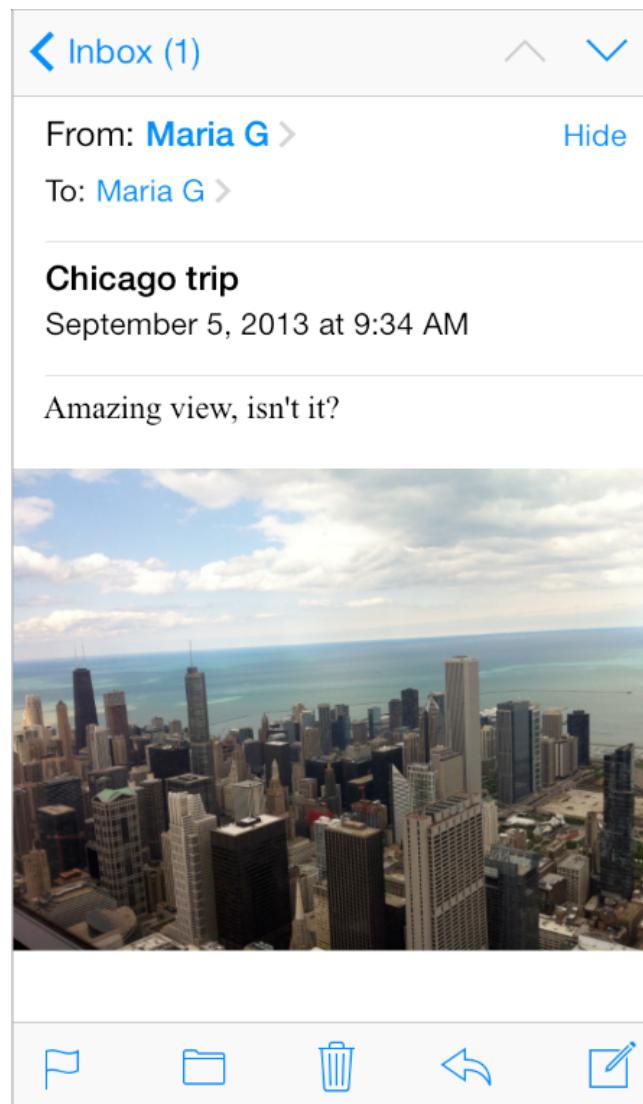
始终确保文字易于阅读。虽然你可以使用属性字符串以富有创意的方式融合不同字体、颜色和对齐方式，但保持文字的可读性仍然很有必要。最好是在文本视图中支持「动态字体」（Dynamic Type）并使用 `UIFont` 的 `preferredFontForTextStyle` 方法以获取显示文本。关于支持「动态字体」的一些准则，请参阅「[文字清晰易读](#)」（第 48 页）；关于程序实现的信息，请参阅《Text Programming Guide for iOS》中的「Text Styles」。

指定不同的键盘类型，以适配你所希望用户输入的不同内容类型。例如，你可能想让用户能方便地访问 URL、PIN 码或者手机号码。但请注意，由于键盘的输入方式和布局由用户的语言设置决定，因此这是你不能控制的。

iOS 提供了不同的键盘类型，每一种设计都是为了方便一种不同类型的输入。如需了解关于可用的键盘类型的信息，请参阅 `UIKeyboardType` 的文档。如需了解关于在 app 中管理键盘的内容的更多信息，请参阅《iOS App Programming Guide》中的「Managing the Keyboard」一节。

Web 视图

Web 视图（web view）是一个可以显示富 HTML 内容的区域（下图中 iPhone 版「邮件」导航栏和工具栏之间的区域就是 Web 视图）。



API 备注: 如需了解更多关于在代码中定义 web 视图的内容, 参阅「Web Views」。

Web 视图:

- 能显示 web 内容
- 在 web 内容中执行一些自动处理, 例如将手机号码转换为电话超链接

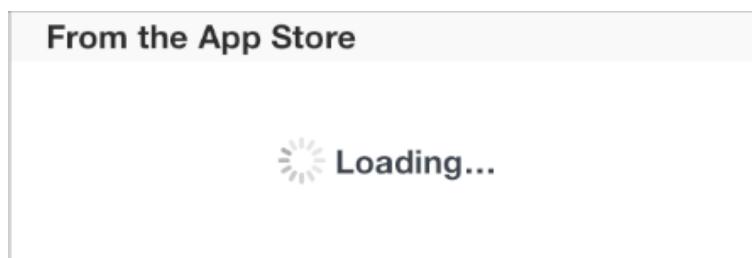
如果你有一个网页或 web app, 你可能会决定用 web 视图来实现一个简单的 iOS app, 以对你的网页或 web app 提供一个封装。如果你计划用 web 视图访问你所控制的 web 内容, 请务必要阅读《Safari Web Content Guide》。

避免使用 web 视图去创建一个看起来和用起来都很像迷你网页浏览器的 app。人们期望使用 iOS 自带的 Safari 去浏览网络内容, 因此我们并不推荐在你的 app 中复制这种广泛存在的功能。

控件

活动指示器

活动指示器（activity indicator）表示某个任务或进程正在进行中（如下图的文本标签所示）。



API 备注：如需了解如何在代码中定义活动指示器，请参阅《UIActivityIndicatorView Class Reference》。

活动指示器：

- 在任务正在进行时旋转，在任务完成时消失
- 不允许用户与之交互

在工具栏或主视图中，使用活动指示器可以告知用户进程正在进行当中，但不会提示该进程会何时结束。

不要显示一个静止不动的活动指示器。用户会将其和一个挂掉的进程联想到一起。

使用活动指示器来让用户知道他们的任务或进程没有停止。有时候，仅仅告知用户进程没有停止也比告诉他们进程何时结束更为重要。

自定义一个与所在视图相协调的活动指示器。如果合适的话，让活动指示器的大小和颜色和所在视图的背景相协调。

「添加联系人」按钮

「添加联系人」按钮 (Contact Add button) 可以让用户将一个现有联系人添加到文本框或其他文本视图中。



API 备注：如需了解如何在代码中定义「添加联系人」，请参阅「Buttons」。

「添加联系人」按钮：

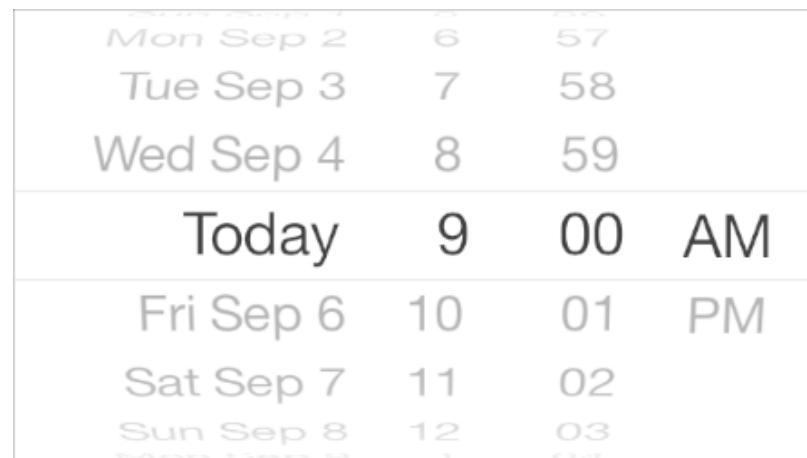
- 会展示用户的联系人列表
- 帮助用户添加一个联系人到「添加联系人」按钮所在的视图中

使用「添加联系人」按钮给了用户一种不需要使用键盘也能访问联系人的简便方式。例如，用户可以在「邮件」的「收件人」字段中轻点「添加联系人」按钮，而不用打字输入收件人的名字。

由于「添加联系人」按钮属于键盘输入联系人信息的替代方式，因此在不支持键盘输入的界面中使用该按钮并不合适。

日期选择器

日期选择器 (date picker) 显示着日期和时间的组件，比如小时、分钟、日和年份。



API 备注：如需了解如何在代码中定义日期选择器，请参阅「Date Pickers」。

日期选择器：

- 最多显示四个独立的滚动，每个滚轮显示一个单独的分类数值，比如月份或小时
- 在视图中央使用深色文本来表示当前选中的数值

- 不能更改其大小（日期选择器的大小和 iPhone 键盘相同）
- 有四种模式，每一种滚轮包含一组不同的数值：
 - **日期和时间。**日期和时间模式（默认模式）以滚动形式显示着日历日期、小时和分钟，以及一个可选的上午/下午滚轮。
 - **时间。**时间模式以滚轮形式显示小时和分钟，以及一个可选上午/下午滚轮。
 - **时期。**日期模式以滚动形式显示了月份、日、年份。
 - **倒数计时器。**倒数计时器模式以滚动形式显示小时和分钟。你可以指定一次倒计时的总时长，最长可达 23 小时 59 分钟。

使用日期选择器可以让用户选择而不是键盘输入一个包含了多个部分（比如日、月份和年份）的日期或时间值。

尽可能在内容中内嵌显示日期选择器。最好避免用户在使用日期选择器时需要进入另一个视图。在 iPad 上，日期选择器可以显示在弹出窗口中或者内嵌到内容里。

如果你的 app 需要，更改分钟滚轮的单位间隔。默认情况下，分钟滚轮会显示 60 个数值（0 到 59）。如果你要展示一个较粗粒度的选项，你可以让分钟滚轮的分钟间隔变大，只要这个间隔能被 60 整除。比如说，你可能会让间隔显示为一刻钟，0, 15, 30 和 45。

详情展开按钮

「详情展开」按钮（Detail Disclosure button）用于展现与该项相关的更多详细信息或功能。



API 备注：如需了解如何在代码中定义「详情展开」按钮，请参阅《UITableViewCell Class Reference》和「Buttons」。

「详情展开」按钮会展开一个独立视图来显示与某个特定项目相关的更多信息或功能。

当「详情展开」按钮在表格的某一行中出现时，轻点此行中按钮外的其他地方均不会触发该按钮；相反，这样只会选中这一行，或者触发由 app 定义的行为。

通常来说，你会在表格视图中使用「详情展开」按钮来让用户获知与这个列表项的更多信息或功能。无论如何，你也可以在其他类型的视图中使用这个元素，以向用户展示与视图中项目相关的更多信息或功能。

信息按钮

「信息」按钮（Info button）用于展现 app 的配置信息，有时候会在当前视图的背面出现。



API 备注：如需了解更多关于在代码中定义「信息」按钮的内容，请参阅「Buttons」。

iOS 包含两种「信息」按钮样式：适用于浅色内容的深色按钮；以及适用于深色内容的浅色按钮。

你可以使用「信息」按钮来展现 app 的配置信息或选项。根据自己 app 的 UI 去选择最为协调的「信息」按钮样式。

标签

标签（label）用于显示静态文本。

Create a stream or join one to share your
best shots and enjoy friends' comments
and contributions right in the iOS photos
app.

API 备注：如需了解在代码中定义标签的更多信息，请参阅《UILabel Class Reference》。

标签：

- 显示任意长度的静态文本
- 不支持除复制文本之外的用户交互行为

你可以使用标签去命名或描述你的部分 UI，又或者是给用户提供一些简短的信息。标签最适合用来显示相对较少的文本。

确保让你的标签清晰易读。最好是在标签中支持「动态字体」并使用 UIFont 的 preferredFontForTextStyle 方法以获取显示文本。如果你选择使用一个自定义字体，请不要为了花哨的字体或艳丽的颜色从而牺牲文本清晰度。

（关于在 app 中使用文本的准则，请参阅「[颜色和文字设计](#)」（第 48 页）；如需了解关于「动态字体」的信息，请参阅《Text Programming Guide for iOS》中的「Text Styles」。）

网络活动指示器

网络活动指示器 (**network activity indicator**) 会在状态栏中出现，表示网络活动正在进行。



API 备注：你可以在代码中使用 `UIApplication` 方法中的 `networkActivityIndicatorVisible` 去控制指示器的可见度。

网络活动指示器：

- 当网络活动正在进行时，会在状态栏上旋转；当网络活动停止时则消失
- 不支持用户交互行为

当你的 app 的网络连接超过好几秒时，显示网络活动指示器以提供反馈。如果操作在这之前就已完成，那么你不需要显示网络活动指示器，因为很可能在用户注意到它之前，指示器就消失了。

页码控件

页码控件 (**page control**) 能表示打开了多少视图以及当前所见的是哪一个（在「天气」中显示如下）



API 备注：如需了解关于在代码定义页码控件的更多信息，请参阅「Page Controls」。

页码控件：

- 在 app 中每个当前打开的视图都会显示为一个指示圆点（从左到右，这些圆点代表着视图被打开的先后顺序）
- 默认情况下，使用不透明的圆点来表示当前所见的视图，用半透明圆点来表示所有其他视图
- 不允许用户不按顺序访问视图
- 不要将圆点收缩或挤压在一起以显示更多被打开的视图；如果你尝试展示超过视图范围的圆点，这些圆点将被截断

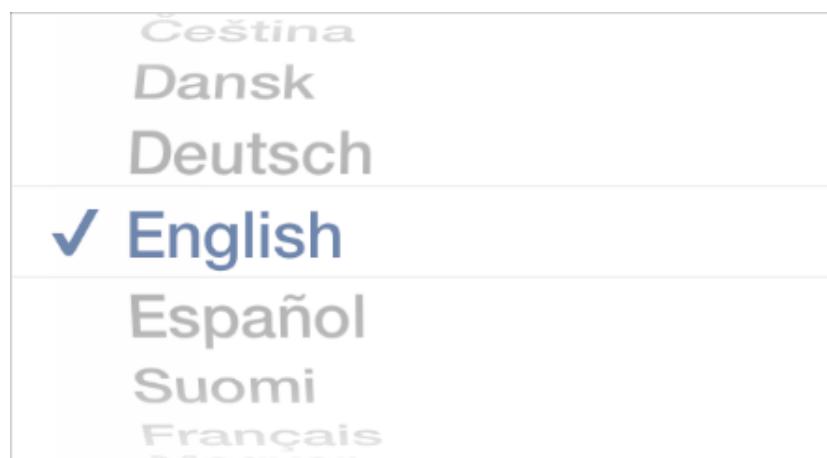
当你的 app 中每一个视图都是和其他视图同级时，你可以使用页码控件。

如果你 app 是在一系列层级视图中显示信息，则不要使用页码控件。页码控件并不能让用户通过指定路径来追溯他们的操作步骤，所以它不能成为一个层级导航方式。

将页码控件在打开视图底部或屏幕边缘垂直居中。在这个位置，页码控件始终可见却又没有太引入注目。避免显示对于当前屏幕方向来说太多的圆点。例如，竖屏方向的 iPhone 屏幕最多可以容纳大约 20 个圆点。

选择器

选择器 (picker) 用来显示一组数值，用户可以从中选择一个。



API 备注：如需了解如何在代码中定义选择器，请参阅《UIPickerView Class Reference》。

选择器：

- 是日期选择器的通用版本（如需了解关于日期选择器的更多信息，请参阅「日期选择器」（第 153 页））
- 显示一个或多个滚轮，每个滚轮都包含一组数值
- 在视图中央使用深色文本来表示当前选中的数值
- 不能调整大小（选择器的尺寸大小和 iPhone 键盘相同）

使用选择器可以让用户更容易地在一组数值中进行选择。

一般来说，当用户对整组数值都很熟悉时，可以使用选择器。由于滚轮静止时，大部分数值都会被隐藏，所以最好让对这些数值有所预期用户可以。如果你需要展示一大组用户不太了解的选项，选择器可能不是最合适的选择。

尽可能将选择器内嵌到内容中显示。最好避免让用户在使用选择器时需要进入另一个视图。

如果你需要展示大量数值，考虑使用表格视图而不是选择器。因为表格视图有着更高的高度，滚动起来会更快。

进度视图

进度视图 (progress view) 用于展示有已知持续时间的任务或进程进度 (如下图中「邮件」工具栏所示)。



API 备注: 如需了解如何在代码中定义进度视图, 请参阅《UIProgressView Class Reference》。

进度视图:

- 是一条轨迹, 随任务或进程进度从左到右持续填充
- 不允许用户交互行为

iOS 定义了两种进度视图样式:

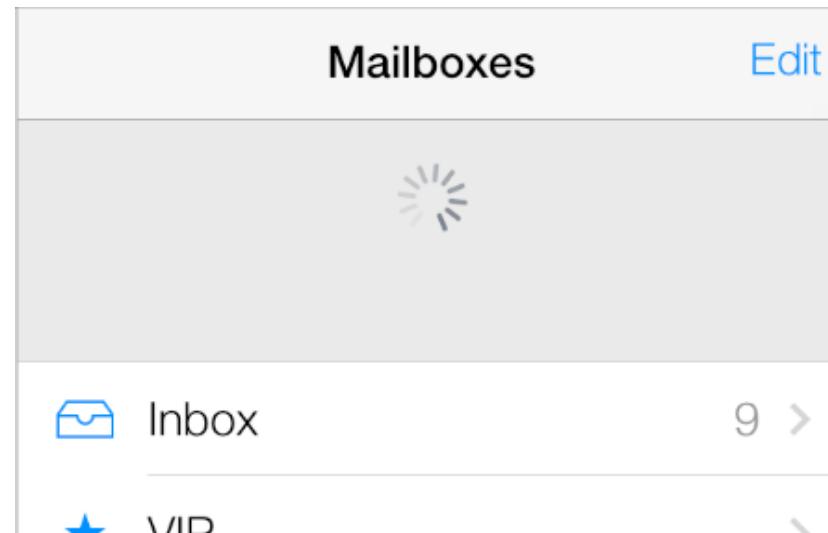
- **默认型。**默认样式看上去更突出, 适合用在 app 的主要内容区域中。
- **条栏型。**条栏样式比默认样式更纤细, 适合用在工具栏中。

使用进度视图能给那些有着明确持续时间的任务提供反馈, 尤其是在显示任务大约还要多久才能完成时至关重要。

如果合适, 请根据你的 app 风格来设计进度视图的外观。你可以通过自定义进度视图来指定外观, 比如为整条轨迹和已填充的进度视图自定义颜色或图像。

刷新控件

刷新控件 (refresh control) 用于执行用户发起的内容刷新—通常在表格中刷新 (如下图中邮箱列表上方所示)。



API 备注：如需了解如何在代码中定义刷新控件的内容，请参阅《UIRefreshControl Class Reference》。

刷新控件：

- 看上去和活动指示器很像
- 可以展示一个标题
- 默认情况下隐藏，当用户从表格顶部边缘往下拖拽刷新时才出现

使用刷新控件能给用户一种一致的方式，去告知表格或其他视图要立即更新内容，从而不需要等待后续的自动更新。

不要因为你提供了刷新控件就停止内容的自动更新。尽管用户喜欢主动请求立即刷新，但他们同样会喜欢内容自己能够自动刷新。如果你过于依赖用户去执行所有刷新，那些没有注意到刷新控件的用户可能就会疑惑，为什么你的 app 总是显示过期内容。一般来说，你需要为用户提供一种选项去立即刷新内容，但你不能奢望由用户负责每一次更新。

只在必要时才提供一个简短的标题。但尤其需要注意的是，不要使用标题去描述该如何使用刷新控件。

圆角矩形按钮

圆角矩形按钮 (rounded rectangle button) 在 iOS 7 中已经不再使用。取而代之使用的是系统按钮——一种类型为 `UIButtonTypeSystem` 的 `UIButton`。如需了解相关指南，请参阅「[系统按钮](#)」（第 162 页）。

分段控件

分段控件 (segmented control) 是一组分段的直线集合，每一个分段都相当于一个可以显示不同视图的按钮。



API 备注：如需了解如何在代码中定义分段控件，请参阅「[Segmented Controls](#)」。

分段控件：

- 由两个或两个以上宽度相同的分段组成，具体宽度取决于分段数量
- 可以显示文本或图像

使用分段控件来提供紧密相关而又互相排斥的选项。

确保每个分段都容易点击。为了保证每个分段都有 44×44 点的舒适点击区域，请控制分段数量。在 iPhone 上，分段控件应等于或少于 5 个分段。

尽可能保证每个分段的内容长度一致。由于分段控件中所有分段的宽度都相同，当有些分段被内容填满而有些又没有时会看起来不太美观。

避免在单个分段混合放置文本和图像。分段控件可以包含文本或图像。一个独立的分段可以包含纯文本或纯图像，但不能同时存在。一般来说，最好避免在同一个分段控件中，一些分段放置文字，而另一些分段又放置图像。

如果自定义了分段控件，必要时请调整内容布局。如果你为分段控件自定义了背景样式，请确保自动居中的控件内容仍然显示良好。你可以使用条栏度量 (bar metrics) API 去调整分段控件内部的内容布局（如需了解关于指定条栏度量的更多信息，请参阅 UISegmentedControl 中 appearance-customization APIs 一节）。

滑块

滑块 (slider) 允许用户在一个限定范围内调整数值或进度（如下图中自定义左右两侧所示）。



API 备注：如需了解如何在代码中定义滑块，请参阅「Sliders」。

滑块：

- 由一条水平的滑轨和一个滑块（用户可以滑动的圆形控件）组成
- 可以放置图像（可选），用来传达左右两端数值的含义
- 填充滑轨从最小值（通常在左端）到滑块之间的部分

使用滑块可以让用户精确地控制他们可以选择的值，或者操作当前进程。

如果必要，可以为滑块创建自定义外观。例如，你可以：

- 定义滑块的外观，以便用户一眼就可以看出滑块是否激活
- 在滑轨两端放置图像以帮助用户理解滑块的作用
- 通常，这些自定义图像对应着滑块控制范围内的最小值和最大值。滑块可以控制图像尺寸，例如，在最小值这端显示一个很小的图像，在最大值那段显示一个很大的图像。
- 为滑轨定义两种不同的外观，这取决于滑块的哪一边显示哪一种控制状态。

不要使用滑块去显示一个音量控件。如果你需要显示音量滑块，用 MPVolumeView 类去使用系统自带的音量滑块。请注意，在当前激活的音频输出设备不支持音量控制时，音量滑块会被替换为适当的设备名称。

步进器

步进器 (stepper) 能以常数量来增加或减少某个数值。



API 备注：如需了解如何在代码中定义步进器，请参阅「Steppers」

步进器：

- 是一个分为两段的控件，其中一段默认显示加号，另一段默认显示减号
- 支持自定义图像
- 不显示用户所调整的数值.

在用户可能需要对数值进行微调时，可以使用步进器。

避免在用户想要大幅度调整数值时使用步进器。在「打印机选项」的操作菜单中使用步进器去设置打印份数会很合适，因为用户很少会去大幅度改变这个数值。但另一方面，让用户使用步进器去选择打印页码范围就会很糟糕，因为即便设置一个合适的页码范围都需要点击很多下。

确保步进器所影响的数值变化显而易见。步进器不会显示任何数值，所以你需要确保让用户感知到他们使用步进器所做的数值调整。

开关

开关 (switch) 表现了两种互斥的选项或状态。



API 备注：如需了解如何在代码中定义开关的，请参阅「Switches」。

开关：

- 显示一个项目的二元状态
- 仅用于表格视图

在表格中使用开关来向用户提供一种在两个选项中指定一个的方式，比如是/否或开/关，以管理项目状态。

你可以使用开关控件去改变视图中其他 UI 元素的状态。根据用户所做的选择，去显示或隐藏新的列表项，或者将某个列表项激活或关闭。

系统按钮

系统按钮 (system button) 用来执行 app 定义的操作。

Button

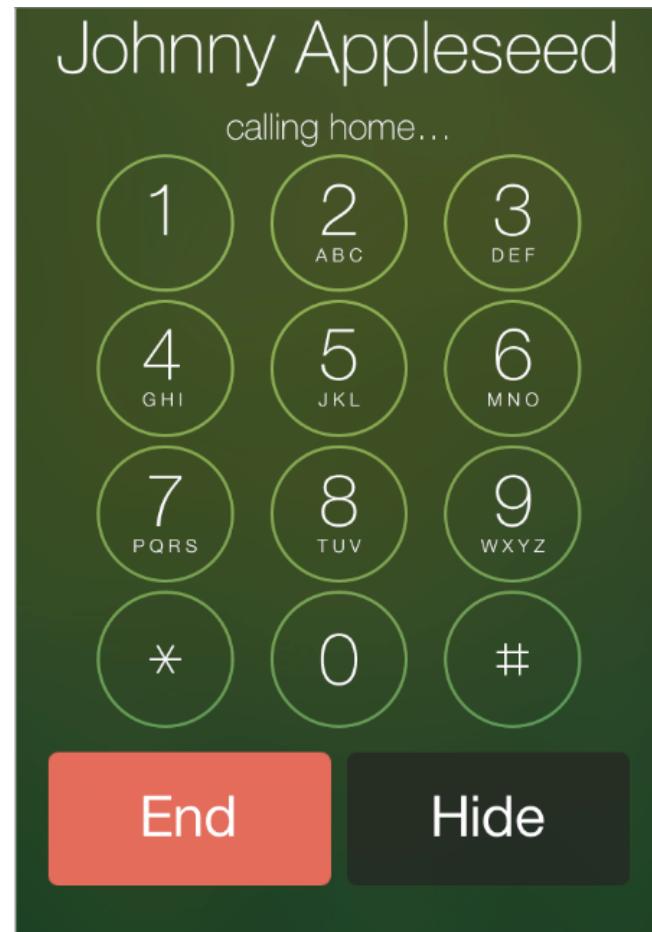
API 备注: 在 iOS 7 中，`UIButtonTypeRoundedRect` 已经被 `UIButtonTypeSystem` 所取代。对于 iOS 6 中使用圆角矩形图标的 app，当其被安装到 iOS 7 上时，会自动获取系统的按钮外观。如需了解如何在代码中定义系统按钮，请参阅「Buttons」。

系统按钮：

- 默认情况下没有边框和背景
- 可以包含图标或文本标题
- 支持自定义样式，比如边框和背景图像（想要添加自定义外观，请使用 `UIButtonTypeCustom` 类型的按钮并提供自定义背景图像）

使用系统按钮可以去执行某个操作。当你要为系统按钮提供标题时，请遵循以下方法：

- **使用动词或动词短语来描述按钮所执行的操作。**一个基于操作行为的标题能向用户显示按钮的交互性，并告知他们点按之后会有什么结果。
- **使用标题大写样式 (title-style capitalization)。**除了冠词、并列连词以及少于四个字的介词，每个单词的首字母都要大写。
- **避免使用太长的标题。**过长的文本会被截断，这会让用户难以理解其含义。

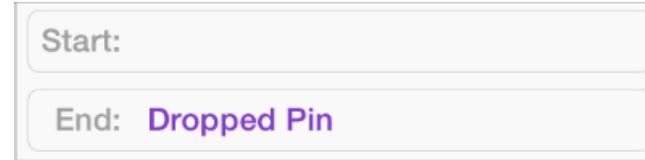


如果合适，为内容区域中的系统按钮添加边框或背景显示。大多数时候，你可以通过撰写一个清晰的行动号召（call-to-action）标题、指定色调和提供上下文情境，来避免让按钮增加装饰。但在某些内容区域，无论怎样，通过增加边框或背景显示来吸引用户关注也是合适的。

以 iPhone 为例，被加上边框的数字按键强化了用户拨打电话的心理模型，而「结束」和「隐藏」按钮的背景让用户有了易于点按的更大区域。

文本框

文本框 (text field) 支持用户输入单行的文本（如下图中目的描述和占位符文本处所示）。



API 备注：如需了解关于定义文本框并自定义以显示图像和按钮的更多信息，请参阅「Text Fields」。

文本框：

- 是一个固定高度的圆角框
- 当用户在其中轻点时，会自动呼出键盘
- 可以包含系统提供的按钮，如「书签」按钮
- 可以显示使用多种样式的文本（如需了解更多关于此的内容，请参阅 UITextView）

使用文本框来让用户输入少量信息。

如果有助于用户理解如何使用文本框，那可以自定义文本框。例如，你可以在文本框的左侧或右侧显示自定义图像，或者添加一个系统提供的按钮，如「书签」按钮等。一般来说，你可以在文本框左侧传达文本框含义，而在右侧显示附加功能，如书签等。

合适的话，在文本框右侧显示「清除」按钮。在这个元素出现时，轻点它便会清除文本框中的内容，无论你在文本框中还可能显示了什么其他图像。

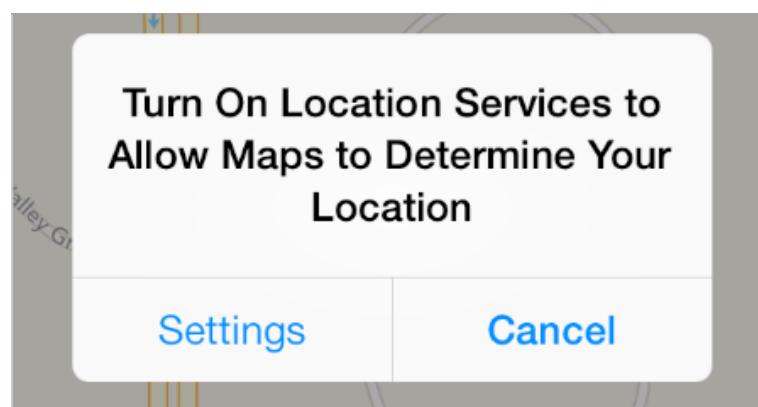
如果可以帮助用户理解文本框的目的，可以在其中显示提示语。当文本框中没有其他文字时，可以在其中显示占位文本一如「姓名」或「地址」。

指定一种键盘类型，以适配你所希望用户输入的内容类型。例如，你可能想让用户更容易地输入 URL、PIN 码或者手机号码。iOS 提供了各种不同的键盘类型，每种设计都是为了方便一种不同类型的输入。如需了解可用的键盘类型，请参阅 `UIKeyboardType` 的文档。如需了解关于在 app 中管理键盘的内容，请参阅《iOS App Programming Guide》中的「Managing the Keyboard」。但请注意，你不能控制键盘的输入方式和布局，因为这些由用户的语言设置决定。

临时视图

警告框

警告框 (alert) 向用户提示会影响他们使用 app 或设备的重要信息。



API 备注：如需了解关于在代码中使用警告框的信息，请参阅《UIAlertView Class Reference》。

警告框：

- 显示一个必需的标题，和一条可选的消息
- 包含一个或多个按钮

警告出现得少会有助于用户对其认真对待。最好严格控制你的 app 显示的警告框数量，并确保每一个警告框都提供了重要信息和有用的选择。

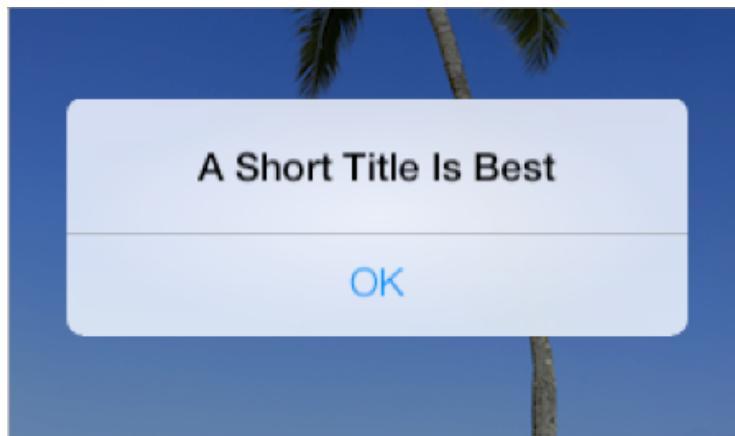
如果警告框用来做这些事情...	可以这样做来代替警告框...
提供与 app 的标准功能相关的信息	设计一种引人注目但又和你的 app 的样式相协调的方式去展示信息
告知用户任务正在正常进行	使用进度视图或活动指示器（在「进度视图」（第 158 页）和「活动指示器」（第 152 页）中有所描述），又或者将状态信息融入到 app 的界面中去。
要求用户对发起的任务进行确认	使用操作列表（在「操作列表」（第 168 页）中有描述）。
告知用户他们所不能解决的问题	如果问题不是很严重，可以将信息放到 app 的界面中显示；否则，请使用警告框。

避免创建不必要的警告框。通常来说，警告框在以下这些场景中是不需要的：

你将读到的是警告文本设计指南，了解这些定义非常有用：

- **标题大写样式 (Title-style capitalization)** 指的是每一个单词的首字母都要大写，除了冠词、并列连词以及少于四个字母且不是第一个单词的介词。
- **句子大写样式 (Sentence-style capitalization)** 指的是第一个单词的首字母需要大写，单词的其余部分小写，除非他们是专有名词或专有形容词。

简明扼要地描述当前情境，并告诉人们他们可以做什么。理想情况下，你撰写的文本要给用户足够的情境，让他们理解为什么警告框会出现，以决定点哪个按钮。



保持标题足够简短，尽可能以一行显示。冗长的警告标题让人难以快速阅读，而且它还可能会被截断或者让警告消息需要滚动。

避免一个词的标题。单个词的标题，比如「错误」或「警告」，很少能提供任何有用的信息。

可能的话，尽量使用短句。一段简洁清晰的陈述往往比一个完整的句子更容易理解。

尽可能撰写一个不需要补充说明的标题。例如，如果你用一个问句—或者偶尔是两个短句—作为警告标题，很可能你可以不需要添加消息说明。

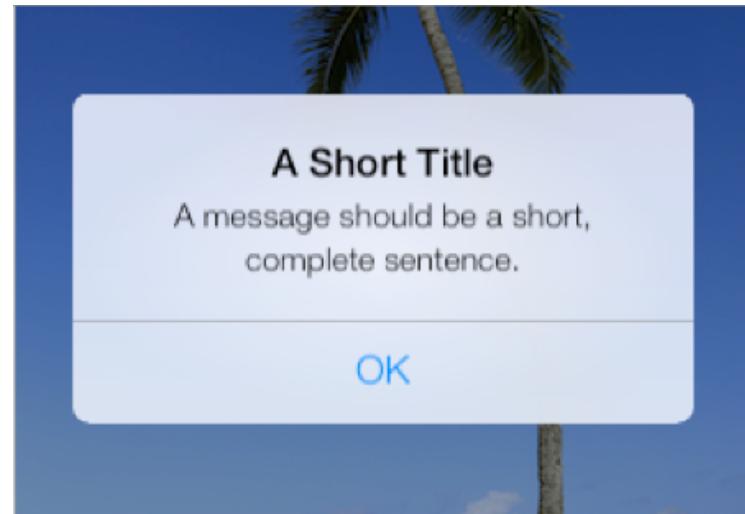
不要刻意避免使用负面措辞。用户理解大多数警告框都是为了告诉他们发生的问题，或者对危险情况作出警告。因此，负面但直接的措辞效果会好于正面但委婉的措辞。

尽可能避免使用「你」、「你的」、「我」和「我的」。有时候，这些直接指向人的文本可能会引起歧义，甚至可能会被理解成侮辱或傲慢。

恰如其分地使用大小写和标点符号。具体来说：

当警告框标题...	则使用...
是一个短句或一个简单但又不是问句的句子	标题大写样式，且句末没有标点
是一个简单的问句	句子大写样式，以问号结尾
由两个或更多的句子组成	句子大写样式，为每个句子选用合适的结束标点

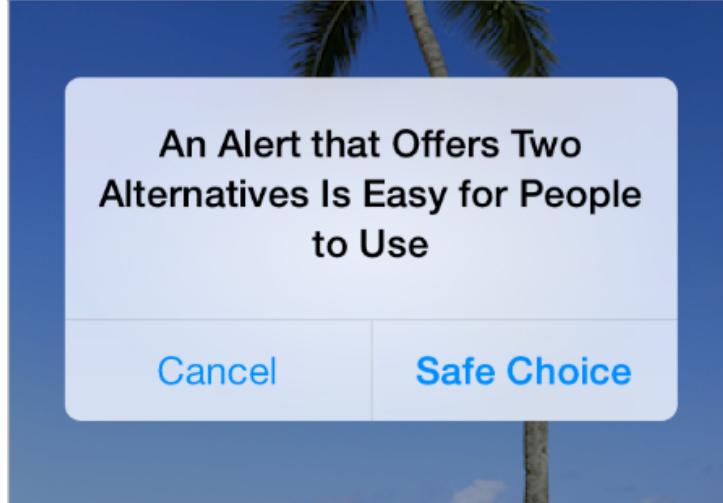
如果必须为警告框提供一条可选的消息正文，请撰写一个简短、完整的句子。如果可能，尽量让消息足够简短以便能在一两行之间显示。如果消息过长导致警告框出现滚动条，这会给用户以糟糕的体验。在消息中使用句子大写样式，并以合适的标点作为结束。



避免在警告文本中描述点击哪个按钮从而导致文本过长。理想情况下，清楚明白的警告文案和合乎逻辑的按钮标题能给人足够的信息去理解当前情况并做出选择。如果你需要提供详细的指引，请遵循以下准则：

- 确保使用「轻点」（tap）（不是「触摸」（touch）或「点击」（click）或「选择」（choose））来描述动作选项。
- 不要用引号将按钮的标题括起来，但要保证其首字母大写。

确保警告框在竖屏和横屏方向上显示正常。横屏方向中的警告框高度会有所限制，可能会和竖屏方向中的样子不太一样。建议你优化警告文本的长度，以便在两种方向上不用滚动都可以被阅读。



一般来说，使用两个按钮的警告框。两个按钮的警告框通常是最有用的，因为对人们来说在两个按钮之间做选择最容易。单个按钮的警告框就不那么有用，因为它通常只是提示用户，并没有赋予用户任何对当前状况的控制能力。包含三个或三个以上按钮的警告框明显比双按钮警告框复杂，应该尽可能避免使用。如果你在一个警告框中添加了太多按钮，它会导致警告框需要滚动，这将是一种糟糕的用户体验。

注意：如果你发现需要向用户提供超过两个的选择，可以考虑使用操作菜单以代替（如需了解如何使用操作菜单，请参阅「操作菜单」（第 168 页））。

合适地放置按钮。理想情况下，最自然的点击按钮应符合两个标准：它执行了用户最想要做的操作；即便用户不小心误点，也不会造成严重问题。具体来说：

- 如果按钮不会造成破坏性后果，而这又是用户最有可能的操作，那么它应当在双按钮警告框的右边。取消按钮则应该放在左边。

- 如果按钮会造成破坏性后果，而这又是用户最有可能的操作，那么它应该放在双按钮警告框的左边。取消按钮则应该放在右边。

注意：在警告框显示时点按「主屏幕」按钮，应如预期的那样退出此 app。这样做的效果类似于轻点「取消」按钮——即，警告框被取消且操作没有被执行。

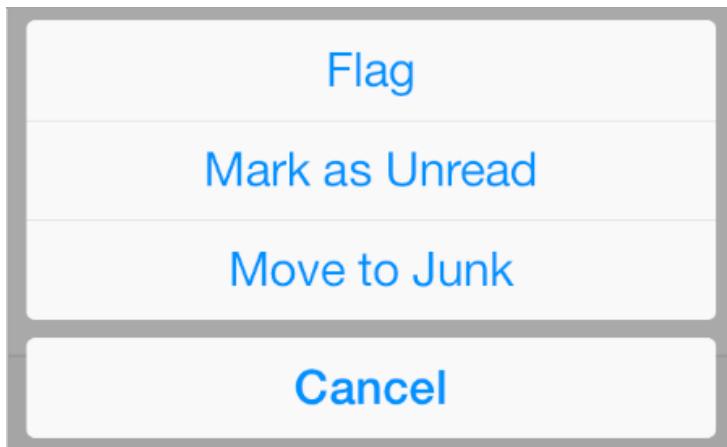
为警告框按钮撰写简短而合乎逻辑的标题。好的按钮标题一般只有一到两个词，描述了轻点按钮后的结果。当你在创建警告框按钮的标题时，请遵循下面这些准则：

- 和所有按钮标题一样，使用标题大写样式，且句末没有标点符号。
- 和所有按钮标题一样，使用和警告框文本直接相关的动词和动词短语——例如，「取消」、「查看所有」、「回复」或「忽略」。
- 如果没有更好的选择，使用「好」(OK)作为简单的接受选项。避免使用「是」和「否」。
- 尽可能避免使用「你」、「你的」、「我」和「我的」。使用这些词语的标题往往会引起歧义，还可能造成冒犯。

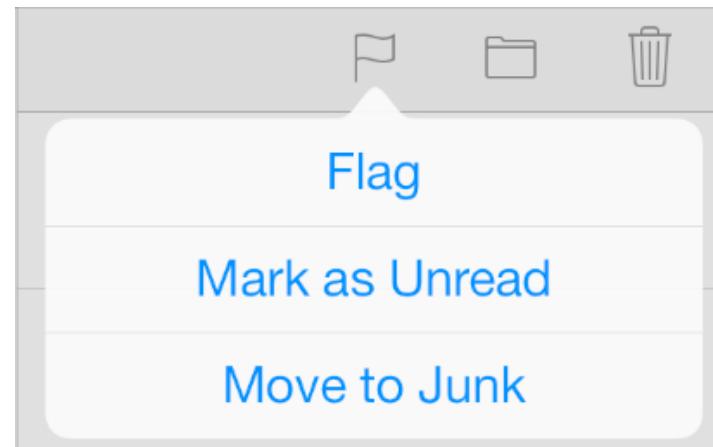
操作菜单

操作菜单 (action sheet) 显示着与用户所发起的任务直接相关的一系列选项。

在 iPhone 中，操作菜单从屏幕底部出现。



在 iPad 上，操作菜单通常会在弹出窗口中显示。



API 备注：如需了解如何在代码中定义操作菜单，请参阅「Action Sheets」。

操作菜单：

- 作为用户操作的结果出现
- 显示两个或两个以上的按钮

使用操作菜单可以：

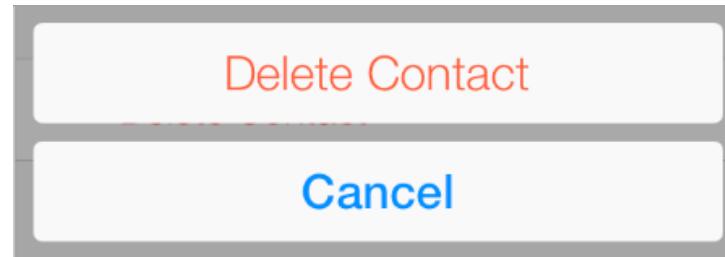
- **提供完成一项任务的不同方法。**操作列表让你可以提供在当前任务情境中奏效的一系列选项，而且这些选项不会永远在 UI 中占据位置。
- **在完成一个有潜在风险的任务前获得用户确认。**操作菜单让用户可以去考虑他们的下一步操作可能带来的潜在风险，并为他们提供一些替代方案。

在 iPhone 上，要放置「取消」按钮以便用户可以轻松但安全地放弃任务。将「取消」按钮放置在操作列表的底部，有利于用户在选择前通读所有选项，。

在 iPad 上，基于用户发起任务的方式来决定操作列表的显示方式。具体如下：

如果任务从如下情境中发起...	则显示操作列表时...	是否包含取消按钮？
从弹出窗口外	不需要动画——即，操作列表和弹出窗口同时出现	否，因为用户可以在弹出窗口以外轻点就能让操作列表消失
在弹出窗口中	需要动画——即，操作列表从弹出窗口内容的顶端滑出	是，因为用户希望不用关闭弹出窗口也能取消操作列表

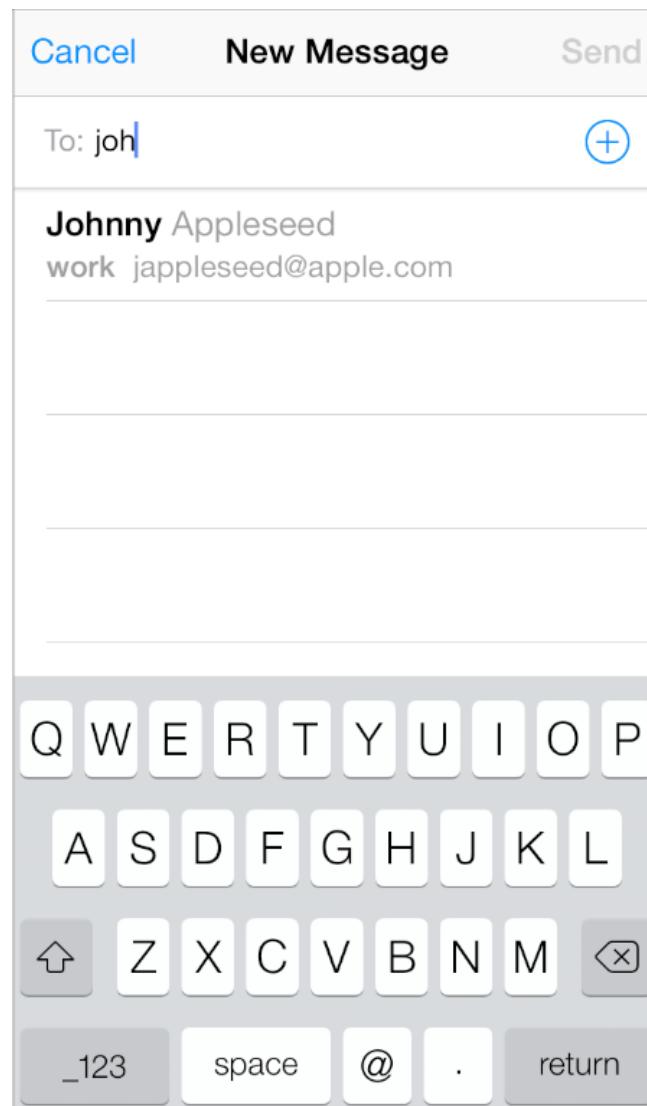
无论在何种设备上，均使用红色来标识那些执行潜在破坏性操作的按钮。在操作列表顶部显示红色按钮，因为越靠近操作列表顶部的按钮越容易引人注意。另外，在 iPhone 上，破坏性的按钮离操作列表底部越远，用户就越不可能在寻找「主屏幕」按钮时误点。



避免让用户滚动操作列表。如果你的操作列表中存在太多按钮，用户必须要滚动才能看完所有选项。这对用户来说是一种烦躁的体验，因为他们必须花更多的时间来分辨这些选项。同样，用户在滚动时很可能会误点某个按钮。

模态视图

模态视图（modal view）是一个以模态形式展现的视图，它为当前任务或情境提供自包含（self-contained）的功能。



API 备注：如需了解在代码中定义模态视图的更多信息，请参阅《UIViewController Class Reference》。

模态视图：

- 占据整个屏幕；在 iPad 上，也可能会占据整个父视图的区域（例如一个弹出窗口）
- 包含完成任务所需的文本和控件
- 通常显示一个完成任务并退出视图的按钮，和一个放弃任务并取消视图的「取消」按钮

当需要完成和你的 app 的基本功能相关的自包含任务时，你可以使用模态视图。模态视图尤其适用于那些所需元素在 app 的主要界面中并非一直出现且有多个步骤的子任务。

在 iPad 上，选择一种适合当前任务以及 app 视觉风格的模态视图样式。你可以使用如下定义的任何一种样式中：

模态视图样式	外观	推荐用于
全屏	占据整个屏幕。	呈现一个较为复杂的任务，用户可以在模态视图情境内部完成这个任务。
页面列表	宽度固定为 768 点；高度与当前屏幕高度相同。 在横屏时，模态视图两侧的屏幕区域都会变暗。	呈现一个较为复杂的任务，用户可以在模态视图情境内部完成这个任务。
表格列表	尺寸固定为 540 x 620 点，在屏幕上居中对齐。 在横屏时唤起键盘，表格列表会上移到状态栏正下方。	从用户那里获取结构化信息。
当前情境	使用与父视图一样的尺寸	在分栏视图窗格、弹出窗口或其他非全屏视图内部显示模态内容。

在 iPad 上，不要在弹出窗口底部显示模态视图。除警告框外，没有任何元素应显示在弹出窗口之上。在极少数情况下，用户在弹出窗口中进行的操作结果必须要以模态视图展现，那也请在模态视图出现前先将弹出窗口关闭。

在 iPhone 上，确保模态视图的整体外观要和你的 app 的外观相协调。例如，模态视图常常包含一个导航栏，而导航栏又包含标题和取消或完成视图任务的按钮。在这种情况下，模态视图的导航栏应当使用和 app 导航栏相同的外观。

如果合适，在所有设备上都显示一个说明任务内容的标题。你也许还需要在视图的其他区域中显示文本，以完整描述任务内容或提供一些指引。

在所有设备上，选择一个合适的转场样式来展现模态视图。使用和你的 app 相协调的转场样式，可以提升用户对模态视图所代表的临时情境转换的感知。要做到这一点，你可以从以下转场样式中选择一个：

- 垂直型。**在垂直样式中，模态视图会从屏幕底部向上滑入，在被取消时向下滑出（这是默认的转场样式）。
- 翻转型。**在翻转样式中，当前视图从右至左水平翻转，随之显示模态视图。从视觉上来说，模态视图看上去像是当前视图的背面。当模态视图被取消时，它会从左至右翻转，随之显示之前的视图。

如果你要在 app 中改变当前模态视图的转场样式，请确保这种改变对用户有价值。用户很容易就能注意到 app 中的行为变化，并且会认为这些变化有特别的意义。因此，最好是建立一种合乎逻辑并保持一致的范式，让用户可以轻松感知并记忆。在没有充分理由时，避免改变转场样式。

图标和图像设计

- 「图标和图像尺寸」 (第 172 页)
- 「应用图标」 (第 174 页)
- 「启动画面」 (第 179 页)
- 「条栏按钮图标」 (第 181 页)
- 「报刊杂志图标」 (第 184 页)
- 「Web Clip 图标」 (第 186 页)
- 「创建可伸缩图像」 (第 187 页)

图标和图像尺寸

每个 app 都需要一个应用图标和一个启动画面。此外，一些 app 还需要自定义导航栏、工具栏和标签栏上的图标以呈现 app 的特有的内容、功能或模式。

和你的 app 中其他的图像图标不同，表 38-1 中的图标和图像需要符合特定标准，以便 iOS 可以正确地显示它们。此外，一些图标和图像问句还有命名要求。（如果你需要支持标准分辨率的 iPhone 或 iPod touch 设备，以下列出的高分辨率尺寸需要除以 2。）

表 39-1 自定义图标和图像的尺寸（像素单位）

描述	iPhone 5 和 iPhone touch 的尺寸（高分辨率）	iPhone 5 和 iPhone touch 的尺寸（高分辨率）	iPad 的尺寸（高分辨率）	iPad 2 和 iPad mini 的尺寸（标准分辨率）
应用图标（必需）	120 x 120	120 x 120	152 x 152	76 x 76
App Store 中的应用图标（必需）	1024 x 1024	1024 x 1024	1024 x 1024	1024 x 1024
启动画面（必需）	640 x 1136	640 x 960	1536 x 2048 (竖屏) 2048 x 1536 (横屏)	768 x 1024 (竖屏) 1024 x 768 (横屏)
Spotlight 搜索结果图标（推荐）	80 x 80	80 x 80	80 x 80	40 x 40
设置图标（推荐）	58 x 58	58 x 58	58 x 58	29 x 29
工具栏和导航栏图标（可选）	大约 44 x 44	大约 44 x 44	大约 44 x 44	大约 22 x 22
工具栏图标（可选）	大约 50 x 50 (最大: 96 x 64)	大约 50 x 50 (最大: 96 x 64)	大约 50 x 50 (最大: 96 x 64)	大约 50 x 50 (最大: 96 x 64)
App Store 中的「报刊杂志」封面图标（「报刊杂志」app 需要）	最长边至少 512 像素	最长边至少 512 像素	最长边至少 512 像素	最长边至少 512 像素
Web Clip 图标（推荐 web app 和网站使用）	120 x 120	120 x 120	152 x 152	76 x 76

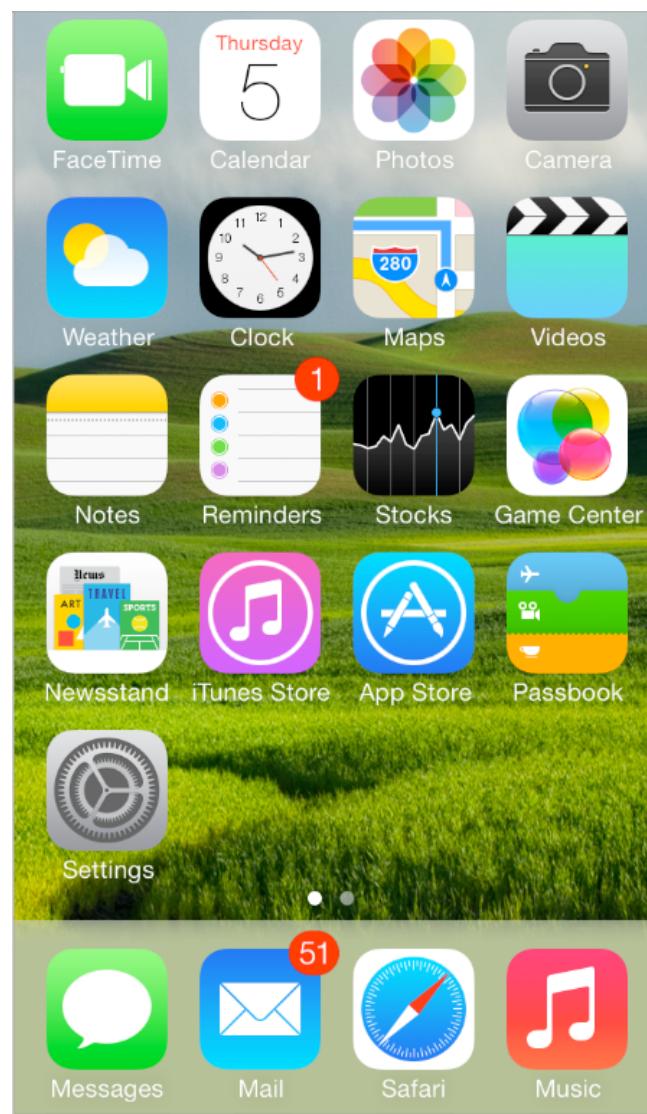
所有图像和图标都推荐使用 PNG 格式，但要避免使用交错的 PNG 图片。

图标和图像标准色彩深度是 24 位——即，给 8 位的红、绿、蓝加上一个 8 位的 alpha 通道。

你不需要强制使用 web 安全色。

应用图标

每个 app 都需要一个漂亮、印象深刻的应用图标，在 App Store 中吸引用户，并在主屏幕页面中脱颖而出。iOS 可以在 Game Center、搜索结果、设置页和 app 创建的文档中使用不同版本的图标。



注意：App 会在其支持的 iOS 版本中使用相同的 app 图标。如果你决定为你 app 的 iOS 7 版本重新设计一个图标，那新图标应当替换旧的那个，即便在不同版本的 iOS 中显示的 UI 不同。

为达到最佳效果，请寻求专业视觉设计师的帮助。一位经验丰富的视觉设计师可以帮助你的 app 建立起整体的视觉风格，并将其应用到所有的图标和图像中。

使用用户会很容易识别的通用图像。通常来说，避免关注一个元素次要或晦涩的方面。例如，「邮件」的应用图标用了一个信封，而非一个乡村风格的邮筒、邮包或邮局符号。

拥抱简洁。尤其是要避免在你的图标中塞入大量不同的图形。寻找一个可以代表你 app 实质的单一元素，以一种简单、独特的形状传达出来，并谨慎地增加细节。如果图标的内容或形状过于复杂，细节便会成为干扰，还可能在更小尺寸时模糊不清。

建议：要测试你的应用图标在小尺寸时的显示效果，将其放到「主屏幕」页面的一个文件夹中。最好再移动一些应用图标到文件夹中，看是否你的图标显示良好并仍然能清晰辨识。

抽象地演绎你 app 的主要目的。由于图像的细节在很小的尺寸上会变得难以辨别，因此在 app 中使用照片或截屏很少有很好的效果。通常，以艺术的方式演绎现实是更好的做法，因为这样做会让你强调主体中你想要用户记住的方面。

如果你想要描绘现实物体，那就准确呈现。描绘现实物体的图标，应准确复制其特点，如布纹、玻璃、纸张和金属，并能传达出物体的重量和触感。

确保图标在各种背景中显示良好。不要只在浅色或深色背景上测试你的图标，因为你不能预测用户会选择怎样的壁纸。

避免使用透明。应用图标应该是不透明的。如果图标的边界小于推荐尺寸，或者使用透明度以营造「看透」的区域，都会导致图标看起来像是浮在黑色背景上，这在用户设置的漂亮背景上看起来毫无吸引力。

不要在你的设计中使用 iOS 的界面元素。你肯定不希望用户将你的图标和图像和 iOS 的 UI 混淆起来。

不要在你的设计中使用 Apple 硬件产品的复制品。这些代表 Apple 产品的符号是有版权的，它们不能在你的图标和图像中被复制。通常，在你的设计里避免出现任何特定设备是明智的做法，因为这些设计会频繁变更，基于它们的图标很快就会显得过时。

不要在你的界面中再次使用 iOS 内置 app 的图标。在整个系统的不同地方出现相同的图标，却又表示有细微差别的事物，这会让用户混淆。

除了 App Store 中的图标必须被命名为 iTunesArtwork，你可以将应用图标命名为任何你想要的名字。只要你用 CFBundleIcons 键值去声明命名，并在所有高分辨率图标名字中添加 @2x 的后缀，iOS 会根据其大小是否适用去自动选择一个图标。如需了解更多关于图标命名的信息，参阅《iOS App Programming Guide》中的「App Icons」。

为不同的设备创建不同尺寸的应用图标。如果你在创建一个通用的 app，你需要提供四种尺寸的应用图标。

对于 iPhone 和 iPod touch，需要这两种尺寸：

- 120 x 120 像素
- 60 x 60 像素（标准分辨率）

对于 iPad，需要这些尺寸：

- 152 x 152 像素
- 76 x 76 像素（标准分辨率）

一个叠加遮罩前的 120 x 120 像素图标



一个叠加遮罩后的 120 x 120 像素图标



当 iOS 在设备的「主屏幕」页面中显示一个应用图标时，它会自动叠加一个圆角遮罩。确保你的图标有 90° 直角，以便在叠加遮罩后看起来会很棒。

创建一个大尺寸的应用图标，以展示在 App Store 中。虽然很重要的是这个版本要能立即被识别成你的应用图标，但它可以拥有更丰富的细节。这个版本的图标不会被叠加任何视觉效果。

App Store 需要创建两种尺寸的大图标，以便在所有设备上显示良好：

- 1024 x 1024 像素
- 512 x 512 像素（标准分辨率）

这个版本的应用图标一定要分别命名为 `iTunesArtwork@2x` 和 `iTunesArtwor`。

注意：iOS 可能会将大尺寸图像用于其他用途。例如，在 iPad 中，iOS 会将大尺寸图像用来生成大尺寸的文档图标。

如果你正在开发一个 ad-hoc 分发的 app（即，仅在企业内部分发，而不通过 App Store），你也必须提供大尺寸版本的应用图标。这个图标会在 iTunes 中作为你的 app 的标识。

文档图标

如果你的 iOS app 创建了一个自定义类型的文档，你想让用户一眼就能认出这些文档。但你不必为此去设计一个自定义图标，因为 iOS 会使用你的应用图标去生成文档图标。

Spotlight 和「设置」图标

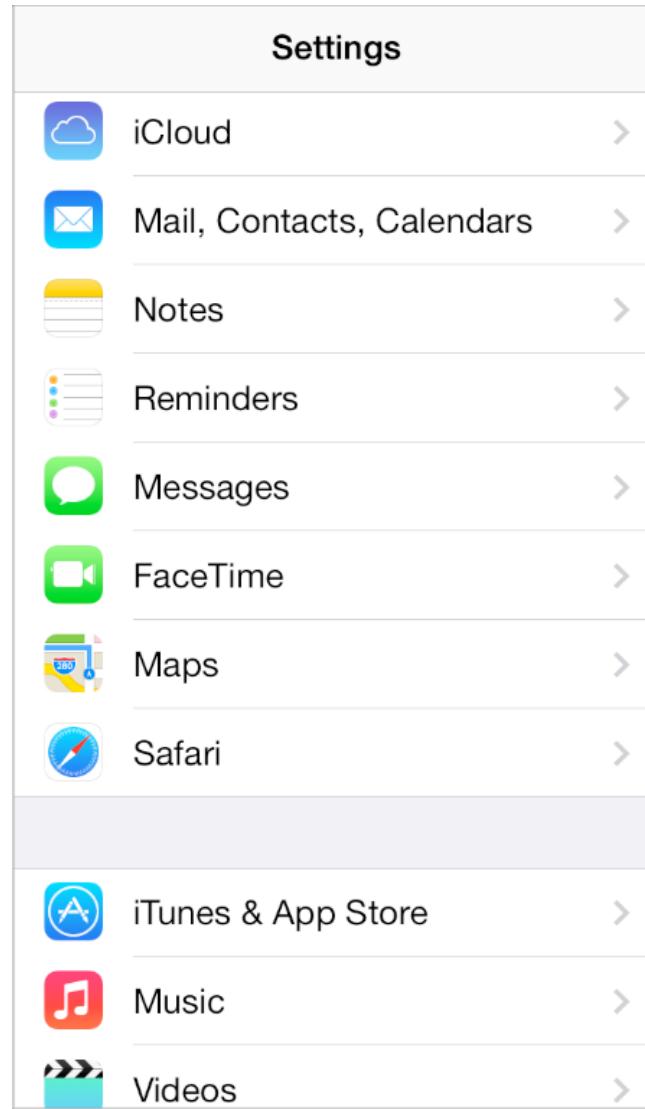
每一个 app 都应提供一个小图标，以便 iOS 可以在 Spotlight 搜索中匹配到这个 app 名称时显示。提供设置的 app 也应提供一个小图标，以在内置的「设置」app 中区分它们。

这些图标应该清晰易识别，以便用户可以从搜索结果或「设置」页的列表中认出你的 app。例如，内置 app 在「设置」中的图标很容易辨别，即使它们很小。

你可以将这些小图标命名为任何名字，只要你用 `CFBundleIcons` 键值去声明命名，并在所有高分辨率图标名字中添加 @2x 的后缀。你可以使用自定义名称，因为 iOS 会根据其大小是否适用去自动选择一个图标。如需了解更多关于图标命名的信息，请参阅《iOS App Programming Guide》的「App Icons」一节。

应用图标

Spotlight 和「设置」图标



对于所有设备来说，使用单独的 Spotlight 搜索结果和「设置」图标。如果你没有提供这些图标，iOS 可能会压缩你的应用图标，显示在这些地方。

至于 iPhone、iPod touch 和 iPad 中的 Spotlight 搜索结果，则按下面两种尺寸创建图标：

- 80 x 80 像素
- 40 x 40 像素 (标准分辨率)

iPhone、iPod touch 和 iPad 中的「设置」，按下面两种尺寸创建图标：

- 58 x 58 像素
- 29 x 29 像素 (标准分辨率)

注意：如果你的图标背景是白色的，不要为了增加其在「设置」中的可识度而添加灰色遮罩。iOS 会自动添加一个 1 像素的描边，以便所有图标在「设置」的白色背景中看起来都很棒。

启动画面

启动画面是在你的 app 启动时显示的一个简洁的占位图像。启动画面会让用户觉得你的 app 响应迅速，因为它立即展现并马上就被你的 app 的第一屏所替换下来。

注意：你必须提供至少一个启动图像。通常来说，一个 iPhone app 至少包含一个竖屏方向的启动图像；一个 iPad app 至少包含一个竖屏的启动图像和一个横屏的启动图像。

由于 iOS 要求你提供不同的启动图像用于不同用途，你需要给每张图片一个名字以指定其应被如何应用。启动图像文件名的格式要包含你所指定的设备、分辨率和图片方向的修饰语。如需了解如何合适地命名启动图像，请参阅《iOS App Programming Guide》中「App Launch (Default) Images」一节。

简洁的启动图像会提升用户体验。尤其是，启动图像不宜提供：

- 如闪屏一般的「app 进入体验」
- 「关于」窗口
- 品牌元素，除非它们真的是你 app 的首屏的静态部分

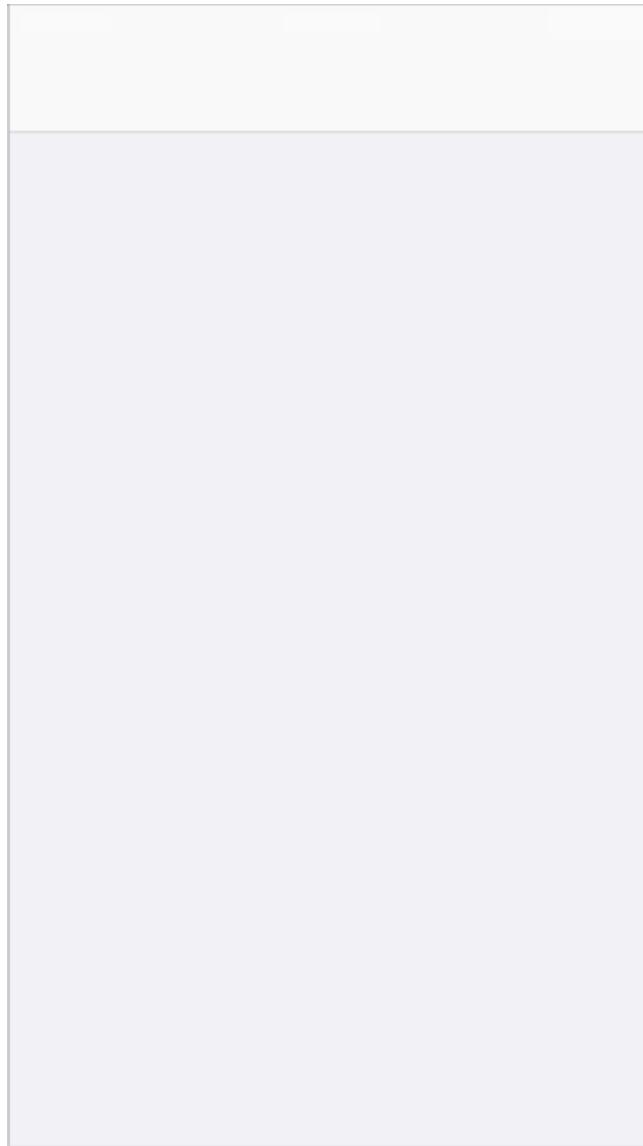
由于用户可能会频繁在 app 间切换，所以你应竭尽所能地让启动时间缩短，而且你设计的启动画面应淡化启动体验而不是吸引用户注意。

设计一个app 的第一屏一样的启动画面，但不包括：

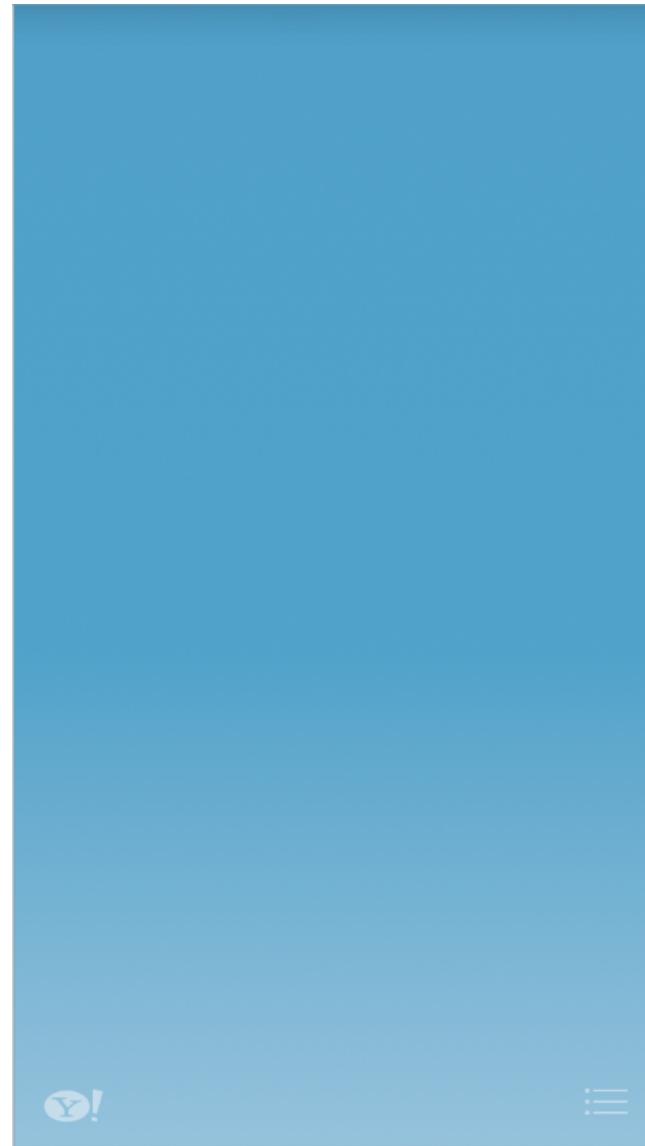
- 文本。启动图像是静态的，所以你在其中显示的任何文本都不会被本地化
- 可能会变化的 UI 元素。如果 app 完成启动后有一个看起来不太一样的元素，用户可能会在启动图像和 app 的首屏之间感受到一个不愉悦的闪烁。

如果你认为下面这些准则会导致一个朴素、无趣的启动图像，你说对了。请记住，启动画面并不是一个供你艺术表达的机会。它的唯一目的是让用户认为你的 app 启动迅速并已准备就绪。例如，「设置」和「天气」都提供了一个和静态背景图差不多的启动画面。

「设置」的启动图像



「天气」的启动图像



为不同设备创建不同尺寸的启动画面。所有设备上的启动画面都必须包含状态栏区域。根据以下尺寸创建启动画面：

对于 iPhone 和 iPod touch (第 5 代)

- 640 x 1136 像素

对于其他 iPhone 和 iPod touch 设备：

- 640 x 960 像素
- 320 x 480 像素 (标准分辨率)

对于 iPad 竖屏模式：

- 1536 x 2048 像素
- 768 x 1024 像素 (标准分辨率)

对于 iPad 横屏模式：

- 2048 x 1536 像素
- 1024 x 768 像素 (标准分辨率)

条栏按钮图标

iOS 定义了很多标准条栏按钮图标，比如「刷新」、「分享」、「新增」和「收藏」。你应尽可能地在你的 app 中使用这些按钮和图标，去表现标准的任务。（如需了解更多关于你可以使用的标准按钮和图标，请参阅「[工具栏和导航栏按钮](#)」（第 124 页）和「[标签栏图标](#)」（第 127 页）。）

如果你的 app 包含不能被标准图标表现的任务或模式，或者标准图标和你的 app 风格不太协调，，你可以设计你自己的条栏按钮图标。在一个较高的水准上，你应当致力于设计这样一个图标：

- **简洁流畅。**过多的细节会使图标显得凌乱或难以辨认。
- **不会和系统图标混淆。**用户应可以一眼就能从标准图标中认出你的自定义图标。
- **通俗易懂并被广泛接受。**努力创建一个大多数用户都可以正确理解的符号，且不会感觉被冒犯。

重要：确保在你的设计中没有使用复制 Apple 产品的图像。这些符号是有版权的，且这些设计可能会频繁更新。

无论你在 app 中是只使用自定义图标还是混合使用自定义和标准图标，所有的这些图标在感知尺寸、细节程度和视觉粗细上都应属于同一体系。

例如，看看 iOS 的条栏图标系列，你会看到它们在大小、细节和粗细上如此相似，进而产生了一种和谐统一的感觉：



要创建一系列连贯的图标，一致性是关键：尽可能让每个图标都使用相同的角度和相同的笔画粗细。为确保所有的图标在感知上都有一致的大小，你需要以不同的实际尺寸制作一些图标。例如，系统内置的图标系列看上去都是一样大，但实际上「收藏」和「语音邮箱」图标比其他三个图标大一些。



如果你正在设计一个自定义的标签栏图标，你应当提供两个版本：一个是未选中的样式，另一个是选中的样式。选中的样式通常是未选中样式的填充版本，但在一些设计中可能会有不同的用法。



要创建一个有着内部细节的图标（如「Radio」图标）的填充版本，请反转这些细节，以便它们在选中状态下仍然醒目。「键盘」图标也有一些内部细节，但如果直接填充背景、将圆圈变为白色轮廓，这样的选中版本会让人迷惑且难以识别。



有时候，当图标被选中时，需要稍作改动才会看上去不错。例如，由于「定时器」和「播客」图标包含打开区域，其选中版本缩小了一点线条以适应一个环形的围绕。



如果当一个图标在被填充时变得难以识别，一个好的替代方案是，使用更粗的线条去刻画被选中版本。例如，相比被用于未选中版本的 2 像素粗的线条，「语音邮箱」和「阅读列表」图标的选中版本都用了 4 像素粗细的线条去刻画。



有时，某个图标形状有一些细节用线条作为轮廓看上去不太好。在「音乐」和「艺术家」图标这样的案例中，你都可以为图标的两个版本使用填充样式。由于选中样式的颜色更深，用户会很容易区分这样的选中和未选中图标样式。



一个为工具栏、导航栏或标签栏自定义的图标，也会被认为是一个**模板 (template)** 图像，因为 iOS 会用它作为一个遮罩去生成为你再 app 运行时所看到的图标。如果你创建了一个全色的模板图像，iOS 会自动忽略颜色。

为了设计一个自定义条栏图标，请遵循以下准则：

- 在合适的 alpha 透明图层中使用纯白色。
- 不包括阴影。
- 使用反锯齿。

如果你想要创建一个看上去和 iOS 7 图标家族近似的条栏图标，可以使用非常细的线条去描绘。具体来说，2 像素的线条（高分辨率）适合用于一些细节丰富的图标，3 像素的线条适合用于细节不那么丰富的图标。

无论图标是何种视觉风格，以如下尺寸创建工具栏或导航栏图标：

- 大约 44 x 44 像素
- 大约 44 x 44 像素（标准分辨率）

无论图标为何种视觉风格，以如下尺寸创建标签栏图标：

- 大约 50 x 50 像素（最大 96 x 64 像素）
- 对于标准分辨率，大约 25 x 25 像素（最大 48 x 32 像素）

不要在自定义的标签栏图标中包含文字，使用标签栏项目 API 去设置每个标签的标题（如，`initWithTitle:image:tag:`）。如果你需要调整标题的动态布局，你可以使用标题调整 API，例如 `setTitlePositionAdjustment:`。

报刊杂志图标

如果你的 app 使用 Newsstand Kit 来发布基于订阅的期刊内容，你需要为其提供图标，显示在用户设备里的 App Store 中。

所有的「报刊杂志」app 都需要提供一个相当于 App Store 中默认封面的「报刊杂志」封面图标。次图标的长边至少为 1024 像素（对于普通分辨率设备，是 512 像素）。注意，这和所有 iOS 都需要提供的应用图标是两套图标。

重要：所有「报刊杂志」图标的高宽比应在 1: 2 和 2: 1 之间。所有图标必须扁平，还要有 90° 的垂直转角。另外，不要在「报刊杂志」图标中添加任何透视效果。

默认的「报刊杂志」图标应该是对一期典型封面的概括描述，主要呈现那些在多期封面中相对一致的部分。例如：

- 避免添加用户不会在实际封面中看到的默认封面图标元素，例如「点击这里查看最新一期」。
- 避免使用有时效的设计或标题，例如和节假日相关的图像或者时事标题。

尤其是，不要在默认「报刊杂志」图标中再次使用往期封面，因为用户可能会将你的 app 和某个特定主题混淆。例如，杂志和报纸的默认图标看起来应当是这样的：

此外，你也需要为每一期新刊提供单独的图标，以便其可以出现在「报刊杂志」的书架和 iOS 设备的多任务界面中。和默认的封面图标不同，每一期的图标应当显示当期内容的细节。

建议你为每一期创建一个单独的大图标，以允许 iOS 在两个地方缩放显示。

具体来说，你所创建的每一期的图标，其长边长度至少为 1024 像素（对于标准分辨率设备，是 512 像素）。为在「报刊杂志」书架和多任务界面中显示当期图标，iOS 会以如下尺寸缩放你的大图标：

表 43-1 每期报刊杂志图标的最大缩放尺寸

设备	缩放尺寸（在书架中）	长边缩放尺寸（在多任务处理界面中）
iPhone and iPod touch	180 x 160 像素（标准分辨率中为 90 x 80 像素）	120 像素（标准分辨率中为 60 像素）
iPad	长边为 252 像素（标准分辨率中为 126 像素）	152 像素（标准分辨率中为 76 像素）

如需了解如何设置「报刊杂志」应用，请参阅《iTunes Connect Developer Guide》。

重要：iOS 7 不会为「报刊杂志」图标添加任何视觉装饰（如装订边缘或者多页样式）。如果你的 app 需要支持 iOS 的早期版本，你可以在 `info.plist` 文件中添加 `binding type` 和 `binding edge`，以定义「报刊杂志」图标在 iOS 6.1 和更早版本的设备上如何显示。关于这些键值的更多信息，请参阅《Information Property List Key Reference》中「Contents of the UI Newsstand Icon Dictionary」一节。

Web Clip 图标

如果你有一个 web app 或者网站，你可以提供一个自定义图标，在用户使用 web clip 功能时可以在他们的主屏幕上显示。用户轻点这个图标即可一步访问你的 web 内容。你可以创建一个代表你整站的图标，或者只代表单个页面的图标。

iOS 也在 Safari 的「收藏」中以矩阵显示 web clip 图标，当用户在 Safari 中轻点 URL 网址或打开一个新标签页时出现。

如果你的 web 内容有一个常见的图像或的配色作为特征，将其纳入你的图标是正确的做法。无论如何，要确保你的图标在设备上看起来很棒，你应当遵循本节中的准则。（如需了解如何在 web 内容中添加代码以提供自定义图标，请参阅《Safari Web Content Guide》。）

对于 iPhone 和 iPad touch，创建的图标大小应是：

- 120 x 120 像素
- 60 x 60 像素（标准分辨率）

对于 iPad，创建的图标大小应是：

- 152 x 152 像素
- 76 x 76 像素（标准分辨率）

注意：通过将你的图标命名为 `apple-touch-icon-precomposed.png`，可以防止为其添加任何效果。

创建可伸缩图像

你可以创建可伸缩的图像来自定义一些标准界面元素的背景，例如弹出框、按钮、导航栏、标签栏和工具栏（包括这些条栏上的项目）。为这些元素提供可伸缩的图像可以让 app 有更好的表现。

对于很多界面元素，你还可以通过定义端点（end cap）来改变背景外观。端点定义了这个图像里不可伸缩的区域。例如，你可以创建一个包含了四个端点的可伸缩图像，来定义一个按钮的四个角。当图片被拉伸以填满按钮的背景区域时，被端点定义的部分不会被拉伸。

根据你所提供的可伸缩图像尺寸，iOS 即可以拉伸图像也可以平铺图像以填充 UI 元素的背景区域。**拉伸 (stretch)** 图像意味着等比放大图片，不必考虑其原始宽高比。拉伸的方法虽好，但对可能会失真的多像素图片并不适合。**平铺 (tile)** 图像是将原始图像多次重复平铺以填充目标区域。平铺的性能比拉伸差一些，但这是实现纹理和图纹效果的唯一方式。

一般来说，你应当提供最小尺寸的图像（不包括端点），来产生你想要的效果。例如：

- 如果你想要一个无渐变的纯色，创建一个 1×1 像素的图像。
- 如果你想要一个垂直渐变，创建一个宽 1 像素、高度等于 UI 元素背景的图像。
- 如果你想要有一个重复的纹理样式，你需要创建一个与纹理的重复部分尺寸匹配的图像。
- 如果你想提供一个不重复的纹理样式，你需要创建一个和 UI 元素背景区域尺寸匹配的静态图像。

文档修订历史

下面这张表格描述了《iOS Human Interface Guidelines》的变化。

日期	备注
2013-10-22	新增高分辨率 iPad mini 的应用图标尺寸、更正 Newsstand 图标尺寸以及恢复 iPhone 5 设计准则。
2013-09-18	为 iOS 7 重新整理和更新。

下面这张表格描述了《iOS 人机界面准则》中文版的变化。

日期	备注
2014-01-02	基于 2013-10-22 版本完成翻译



Apple Inc.
Copyright © 2013 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Apple Inc.
1 Infinite Loop Cupertino, CA 95014 408-996-1010

Apple, the Apple logo, AirPlay, Apple TV, Finder, iPad, iPhone, iPod, iPod touch, iTunes, Keynote, OS X, Passbook, Safari, Shake, Siri, Spotlight, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries.

AirPrint, Multi-Touch, and Retina are trademarks of Apple Inc.

Genius, iAd, and iCloud are service marks of Apple Inc., registered in the U.S. and other countries.

App Store is a service mark of Apple Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.