iOS 8 人机界面指南（一）：UI 设计基础

ISUX 原创翻译

2014.9

目录

1.1 为 iOS 而设计（Designing for iOS）

iOS 的革新关键词如下：

- **遵从**：UI 能够更好地帮助用户理解内容并与之互动，但却不会分散用户对内容本身的注意力。
- **清晰**：各种大小的文字应该易读，图标应该醒目，去除多余的修饰，突出重点，很好地突显了设计理念。
- **深度**：视觉的层次和生动的交互动作会赋予 UI 新的活力，不但帮助用户更好理解新 UI 的操作并让用户在使用过程中感到惊喜。

无论你是重新设计一个现有的应用或是重新开发一个，请尝试下列方法：

- 首先，去除了 UI 元素让应用的核心功能呈现得更加直接并强调其相关性。
- 其次，直接使用 iOS 的系统主题让其成为应用的 UI，这样能给用户统一的视觉感受。
- 最后，保证你设计的 UI 可以适应各种设备和不同操作模式，这样用户可以在不同场景下舒适地享用你的应用。

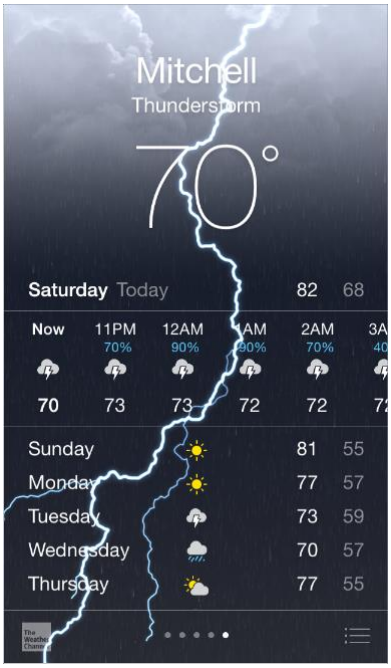
在整个设计过程中，时刻准备着推翻先例，假设问题，并以重点内容和功能（为目标）来驱动每个细节设计。

1.1.1 以内容为核心（Defer to Content）

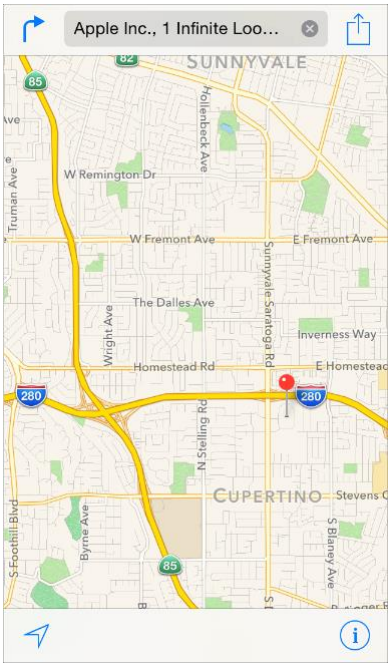
虽然明快美观的 UI 和流畅的动态效果是 iOS 体验的亮点，但内容始终是 iOS 的核心。

这里有一些方法，以确保你的设计能够提升你的应用功能体验并关注内容本身。

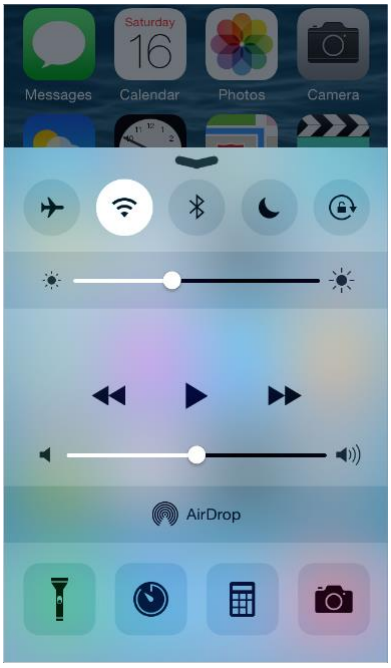
充分利用整个屏幕。天气应用是最好的例子：漂亮的天气图片充满全屏，呈现用户所在地当前天气情况这最重要的信息，同时也留出空间呈现了每个时段的气温数据。



尽量减少视觉修饰和拟物化设计的使用。UI 面板、渐变和阴影有时会让 UI 元素显得很厚重，致使抢了内容的风头。应该以内容为核心，让 UI 成为内容的支撑。



尝试使用半透明底板。半透明的控件——比如控制中心——关联了使用场景，帮助用户看到更多可用的内容，并可以起到短暂的提示作用。在 iOS 中，透明的控件只让它遮挡住的地方变得模糊——看上去像蒙着一层米纸一样——它并没有遮挡屏幕剩余的部分。

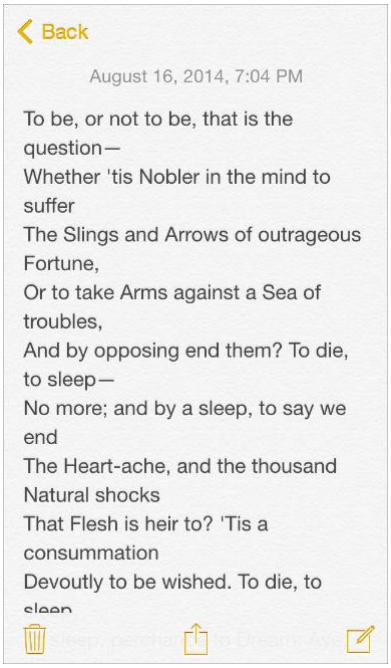


1.1.2 保证清晰度（Provide Clarity）

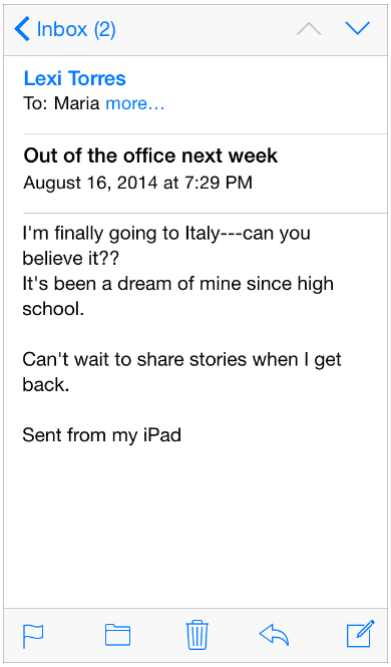
保证清晰度是另一个方法，以确保你的应用中内容始终是核心。以下是几种方法，让最重要的内容和功能清晰，易于交互。

使用大量留白。留白让重要内容和功能显得更加醒目。此外，留白可以传达一种平静和安宁的视觉感受，它可以使一个应用看起来更加聚焦和高效。

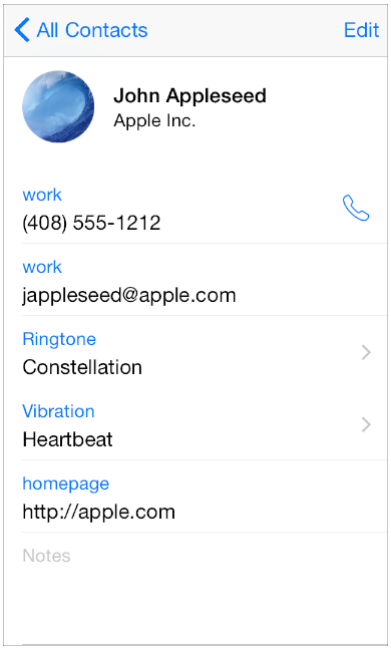
让颜色简化 UI。一个主题色——比如在记事本中使用的黄色——让重要区域更加醒目并巧妙地表示交互性。这同时也给了一个应用一个统一的视觉主题。内置应用使用家族化的系统颜色，无论在深色和浅色背景上看起来都干净，纯粹。



通过使用系统字体确保易读性。iOS 的系统字体自动调整行间距和行的高度，使阅读时文本清晰易读，无论何种大小的字号都表现良好。无论你是使用系统或是自定义字体，务必使用动态型，这样你的应用可以在用户选择不同字号时做出应对。



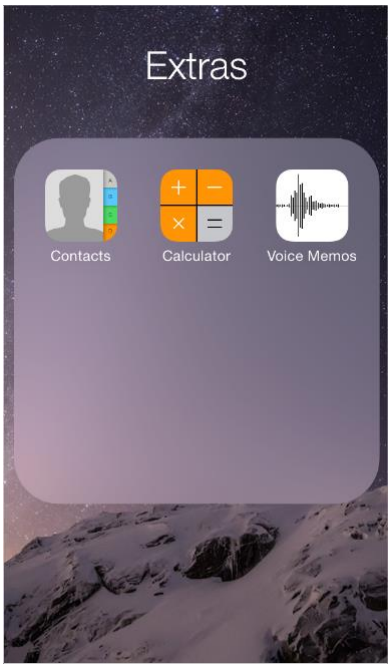
使用无边框的按钮。默认情况下，所有 bar 上的按钮都是无边框的。在内容区域，无边框按钮以文案、颜色以及操作指引标题来表明按钮功能。当按钮被激活时，该按钮呈现高亮的浅色状态来作为操作响应。



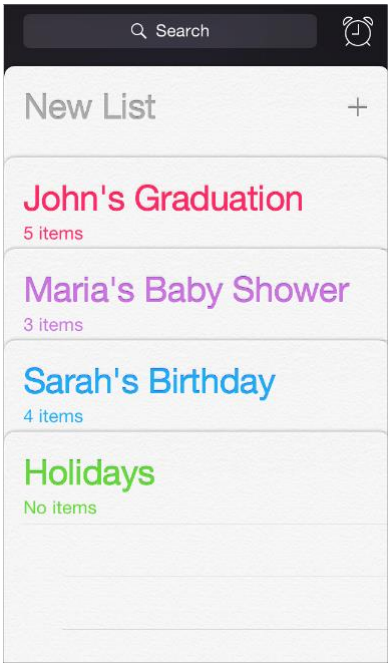
1.1.3 用深度来体现层次（Use Depth to Communicate）

iOS 经常在不同的层级上展现内容，用以表达分组和位置，并帮助用户了解在屏幕上的对象之间的关系。

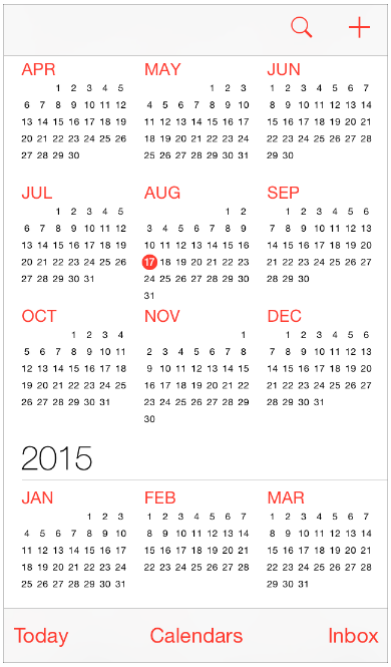
通过使用一个在主屏幕上方的半透明背景浮层来区分文件夹和其余部分的内容。



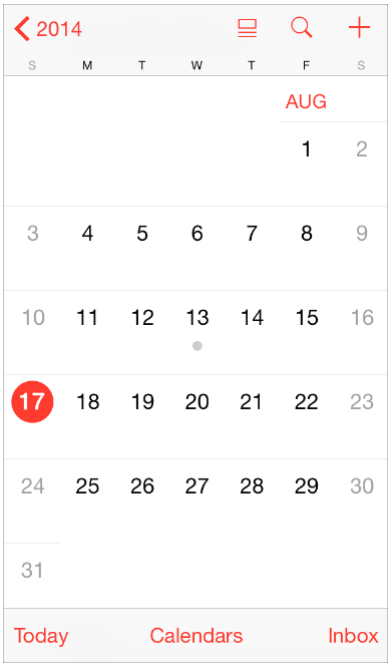
备忘录以不同的层级展示，如插图所示。用户在使用备忘录里的某个条目时，其他的条目被收起在屏幕下方（译者按：其实这个视觉提示使用起来很隐晦）。



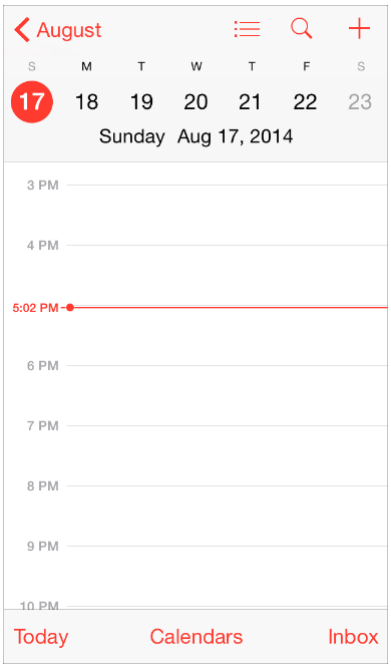
日历有较深的层级，当他们在翻阅年、月、日的时候，以及增强的交互动画给用户一种层级纵深感（循序切换的层次，从年到月到日）。在滚动年份视图时，用户可以即时看到今天的日期以及其他日历任务。当用户处于月份视图时，点击年份视图按钮，月份会缩小至年份视图中的所处位置。



今天的日期依然处于高亮状态，年份出现在返回按钮处，这样用户可以清楚地知道他们在哪儿，他们从哪里进来并且知道如何返回。

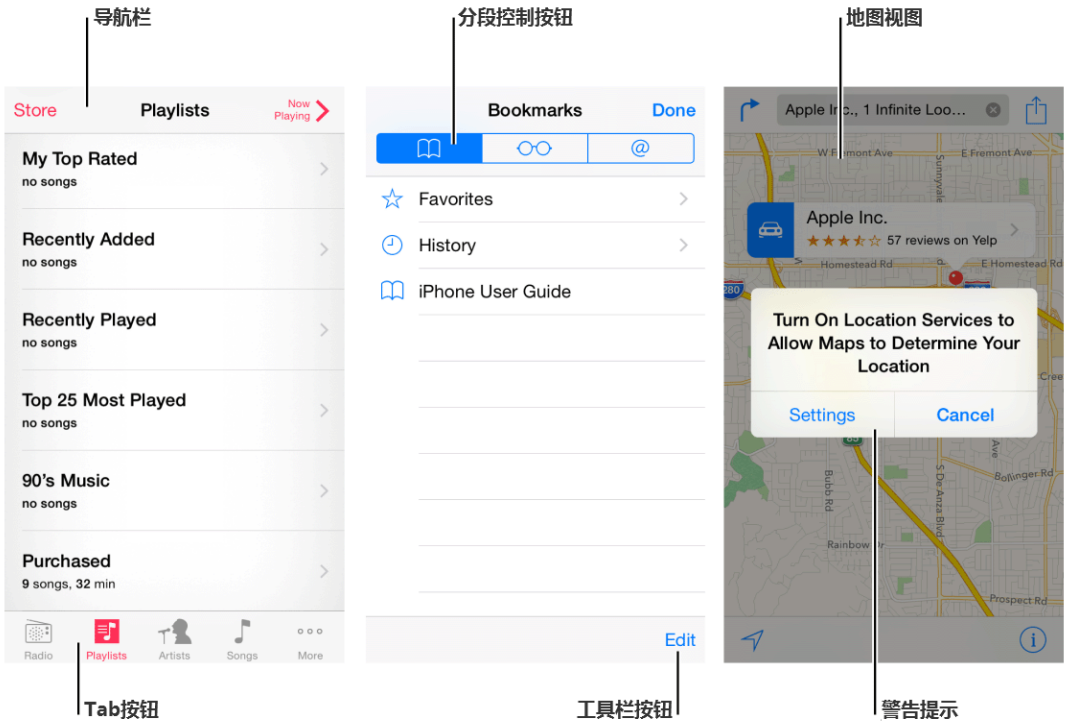


类似的过渡动画出现在用户选择一个日期时：月份视图从所选位置分开，将当前的周日期推向屏幕顶端并翻转出以小时为单位的当天时间视图。这些动画加强了日历上年月日之间的关系的感知度。



1.2 iOS 应用解析（iOS App Anatomy）

几乎所有的 iOS 应用都应用了 UIKit framework 中定义的组件。了解这些组件的名字，作用和构成能够帮助你设计应用过程中做出更好的决定。



UIKit 提供的 UI 组件大致分成以下 4 种大类：

Bars：包含了导航信息，告诉用户他们所在的位置并包含了一些能帮助用户浏览或启动某些操作的控制按钮。

内容视图：包含了应用的主体内容以及某些操作行为，比如滚动、插入、删除、排序等等。

控制按钮：展示信息或者控制动作。

临时视图：短时间出现，给用户重要信息或者额外的选择或者其他功能。

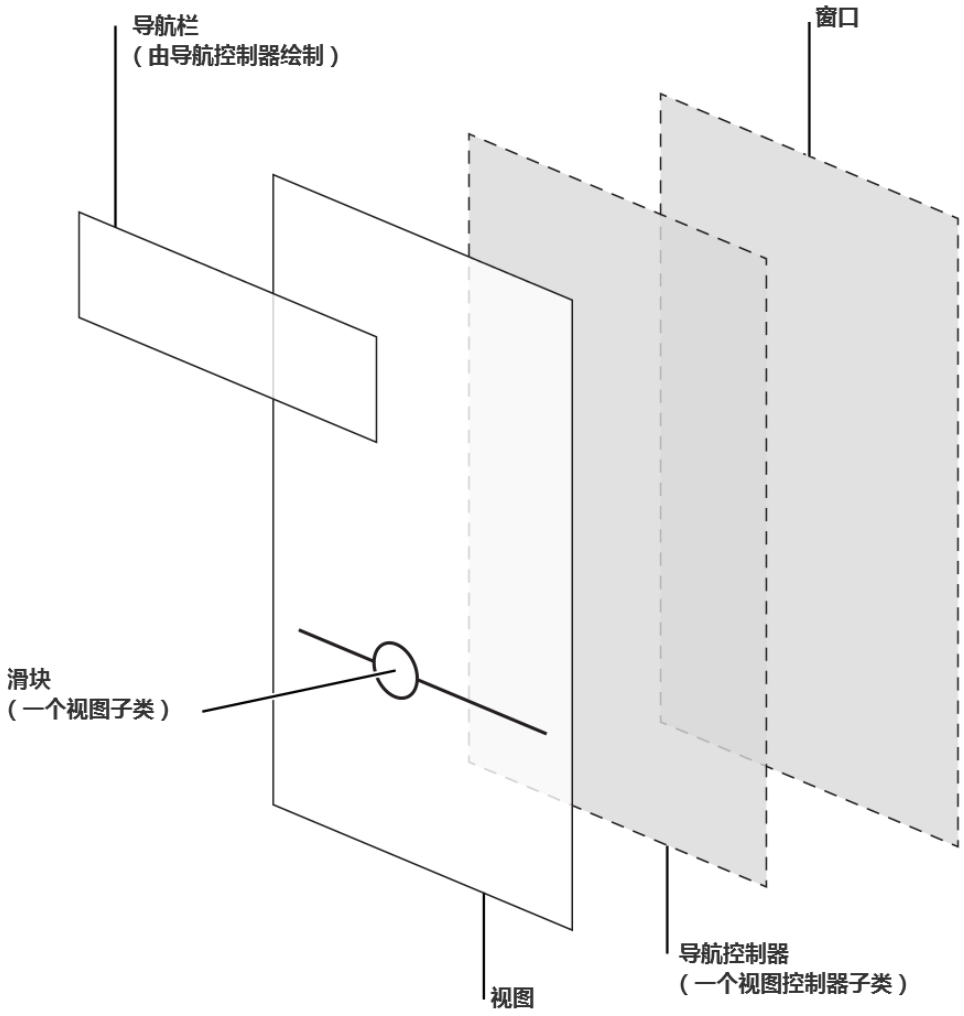
除了定义 UI 组件，UIKit 也定义对象实现的功能，例如手势识别、绘图、辅助功能和打印支持。

从编程的角度来说，UI 组件被认为是不同类别的视图，因为他们从 UIView 得到继承。视图能绘制屏幕内容并且知道用户何时触摸了屏幕。视图的所有类型有：控件（比如按钮和

滑块），内容视图（比如集合视图和表格视图），以及临时视图（如警告提示和动作菜单）。

要在应用中管理一组或者一系列的视图，通常需要使用一个视图控制器，它能协调视图的显示内容，实现与用户交互的功能并能在不同屏幕内容之间切换。比如，“设置”使用了一个导航控制器来展示其视图层级。

下面是一个例子，关于视图与视图控制器如何结合并呈现 iOS 应用的 UI。



虽然开发者认为真正起到作用的是视图和视图控制器，但一般用户感知到的 iOS 应用是不同屏幕内容的集合。从这个角度来看，在应用里，屏幕内容一般对应于一个独特的视觉状态或者模式。

注：一个 iOS 应用程序包含一个窗口。但是，不同于计算机程序中的窗口，iOS 窗口没有可见的部分并且不能在屏幕上被移动到另一个位置。很多 iOS 应用程序只有一个窗口；可以支持外部显示设备器的应用程序可以有不止一个窗口。

在 *iOS Human Interface Guidelines* 中，*screen* 这个词和大部分用户理解的一样。作为一个开发者，你也许需要读一下其他与 UIscreen 相关的章节，这样你可以更好的了解如何关联外部屏幕。

1.3 适应性和布局（Adaptivity and Layout）

1.3.1 为自适应而开发（Build In Adaptivity）

人们通常想随心所欲地使用自己喜欢的应用程序。在 iOS 8 及未来的版本中，你可以使用不同分辨率和自动布局来帮助你定义屏幕布局视图，视图控制器以及需要随显示环境改变的视图（显示环境 display environment 的概念指的是设备的整个屏幕或者其中一部分，比如一个跳出菜单区域或一个分视图控制器的主视图部分）。

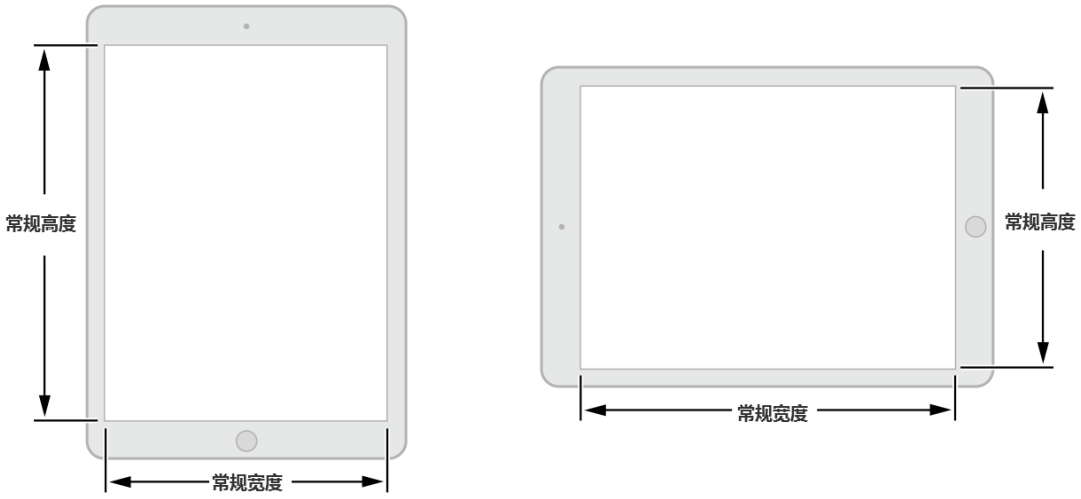
iOS 定义了两个尺寸类别（size class），常规的（regular）和压缩的（compact）。常规尺寸有着较易拓展的空间，而压缩尺寸约束了空间的使用。想要定义一种显示环境，你需要定义横纵尺寸类型。如你所想，iOS 设备可以有横屏竖屏两种不同的使用模式。

iOS 能随着显示环境和尺寸类别变化而自动生成不同布局。举个例子，当垂直尺寸从压缩变为常规时，导航栏和工具栏会自动变高。

当你靠尺寸类别来驱动布局变化时，你的应用在任何显示环境时都能显示得很好。关于如何在 Interface Builder 中更好的使用尺寸类别，你可以查阅 *Size Classes Design Help*。

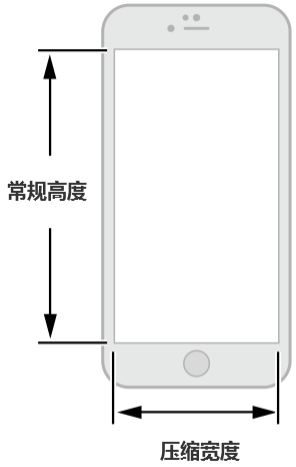
注：在一种尺寸类别中，持续使用 Auto Layout 进行小的布局调整，比如拉伸或压缩内容。

下面的实例可以帮助你理解尺寸类型是如何适配不同设备的显示环境。例如：iPad 在长宽和横屏竖屏时都使用常规尺寸类型。换句话说，iPad 显示环境一直处于垂直和水平的常规状态。



iPhone 的显示环境可根据不同的设备和不同的握持方向而改变。

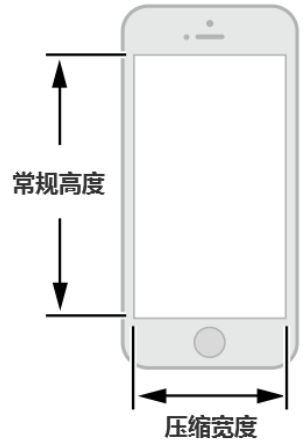
竖屏时，iPhone6 Plus 使用的是常规高度和压缩宽度类型。



横屏时，iPhone6 Plus 使用的是压缩高度和常规宽度类型。

其他 iPhone 型号，包括 iPhone6 使用相同的尺寸类型设置。

竖屏时，iPhone 6，iPhone 5 和 iPhone 4S 使用的是压缩宽度和常规高度。



横屏时，这些设备在宽高上使用的都是压缩类。

1.3.2 在不同环境提供良好体验（Provide a Great Experience in Each Environment）

当你使用自适应来开发 UI 时，你可以保证 UI 跟随显示环境变化而适当地响应。遵照这些指南，你可以给用户良好的设备响应体验。

在所有环境下都保持对主体内容的专注。这是最高优先级。人们使用应用时，与感兴趣的内容发生互动。当使用环境变化的同时，改变专注点会让用户感觉到迷失方向，丧失了对应用的控制。

避免布局上不必要的变化。在所有环境中类似的使用体验让人们在旋转设备或在不同设备上运行你的应用时维持使用模式。例如，如果你使用一个网格在水平的常规模式下显示图像，你无需在一个列表中展示与压缩模式下相同信息，虽然你可能调整了网格的尺寸。

如果你的应用只在一个方向上运行，那么请直接一点。人们希望在各种握持方式下都可以使用你的应用，能满足这个期待是最好的。但是，如果你的应用只在一个方向下运行，那么以下几点请务必注意：

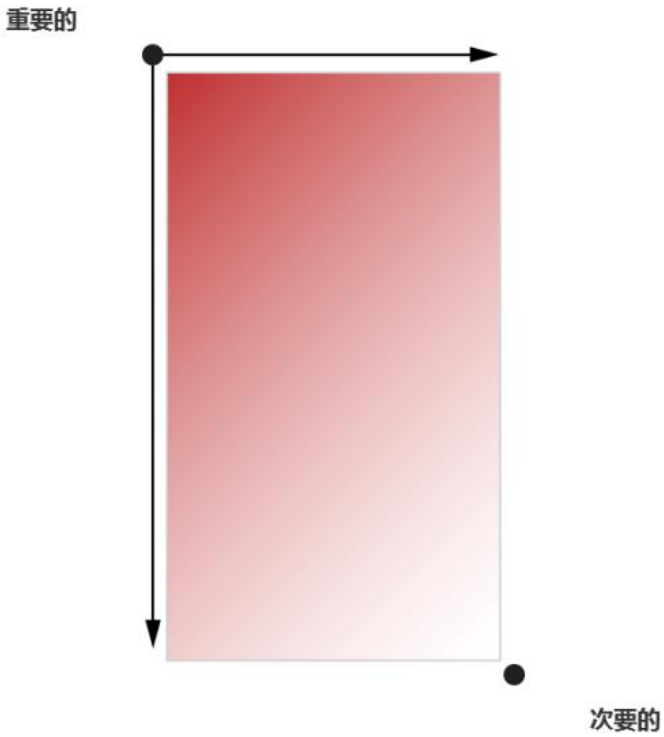
- **避免提示人们旋转设备的提示 UI 出现。**让应用清晰地运行在支持的方向下，让用户明白应该旋转设备，而不是添加不必要的引导性混乱。
- **支持同一个方向上的变化。**例如，如果一个应用只能垂直运行，用户无论用左手或是右手握持时都能触及到 home 键。如果用户在使用应用时 180 度旋转设备，那最好应用内容也能及时响应并旋转 180 度。

如果你的应用将设备方向翻转视为用户输入（的一种指令），那么就按照程序设定的方式来响应设备翻转。举个例子，一个游戏让用户利用设备翻转来移动游戏中的部件，那么这个游戏应用本身（的 UI）不能对翻转屏幕产生响应。在这种情况下，你必须关联两个需要变化的方向，并且允许人们在这两个方向切换直到他们开始（了解并执行）应用的主体任务。一旦人们开始执行主要任务，那就开始按程序设定的那样来响应设备的动作吧。

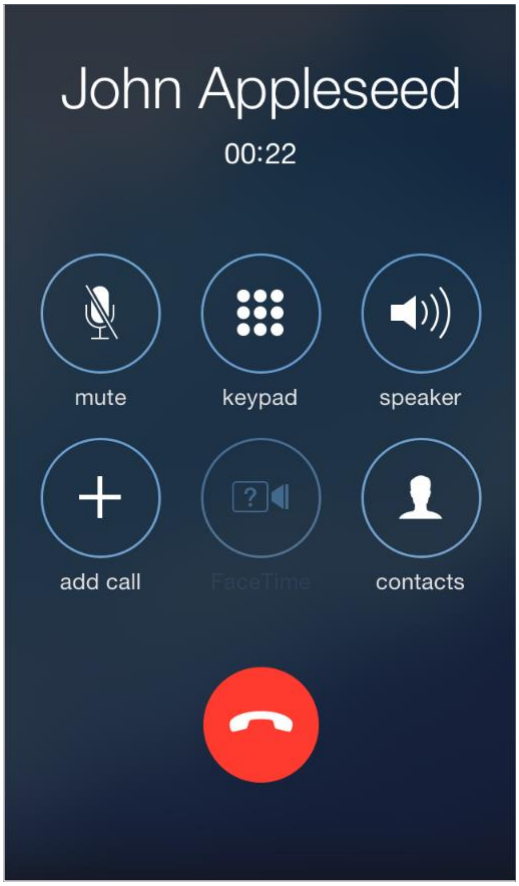
1.3.3 使用布局来沟通（Use Layout to Communicate）

布局包含的不仅仅是一个应用屏幕上的 UI 元素外观。你的布局，应该告诉用户什么是最重要的，他们的选择是什么，以及事物是如何关联起来的。

提升重要内容或功能，让用户容易集中注意在主要任务上。有几个比较好的办法是在屏幕上半部分放置主要内容，以从左到右的习惯，从靠近左侧的屏幕开始。



使用视觉化的重量和平衡向用户展示相关的屏显重要元素。大型控件吸引眼球，而比小的控件更容易在出现时被注意到。而且大型控件也更容易被用户点击，这让它们在应用中更加有用——就像电话和时钟（上面的按钮）——用户经常在容易分心的环境中能（正常）使用它们。



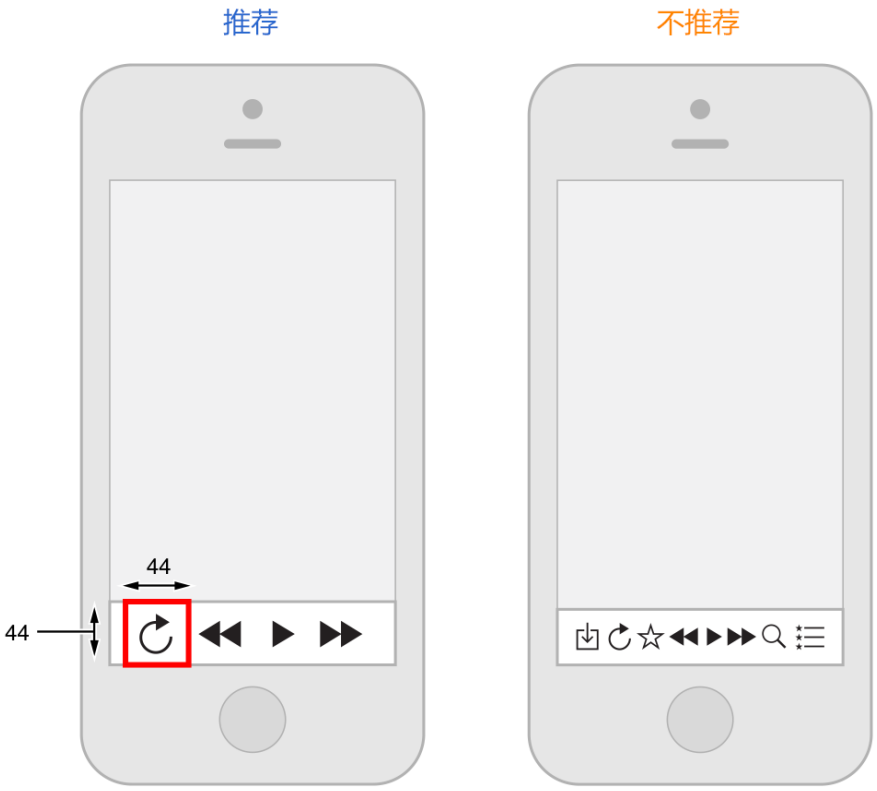
使用对齐来让阅读更舒缓，让分组和层次之间更有秩序。对齐让应用看起来整洁而有序，也让用户在专注于屏幕时更有空间，从而专注于重要信息。不同信息组的缩进与对齐让它们之间的关联更清晰，也让用户更容易找到某个控件。

确保用户能明白处于默认尺寸的首要内容的含义。例如，用户无需水平滚动就能看到重要的文本，或不用放大就可以看到主体图像。

准备好改变字体大小。用户期望大多数应用能有设置字号大小的功能。为了适应一些文本大小的变化，你也许需要调整布局；想要得到更多文本显示相关的信息，你可以查阅章节 Text Should Always Be Legible.

尽量避免 UI 上不一致的表现。在一般情况下，有着相似功能的控件看起来也应该类似。用户常常认为他们看到的不同总有原因，而且他们倾向于花时间去尝试（译者按：因此为了避免用户做无用的尝试所以建议类似功能外观一样）。

给每个互动的元素充足的空间，从而让用户容易操作这些内容和控件。常用的点按类控件的大小是 44 x 44 点（points）。



1.4 起始与停止（Starting and Stopping）

1.4.1 即时启动（Start Instantly）

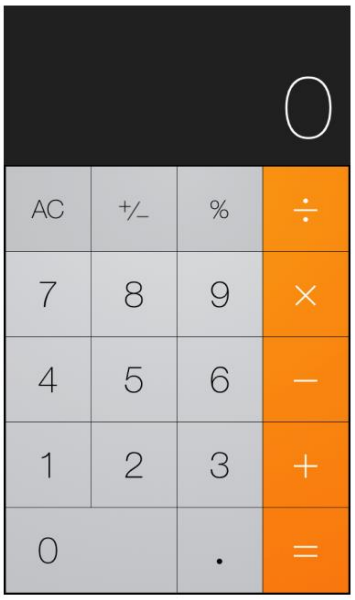
有种说法是用户往往不会花超过一两分钟去审视一个新应用，当你将应用从打开到启动这段时间压缩得很短，并且同时在载入过程中呈现一些对用户有帮助的内容，你就会激发用户的兴趣并给所有用户一个惊喜。

重要：不要在安装过程结束后告诉用户需要重启设备。重启需要时间并且会让人觉得你的应用看上去不可靠而且很难使用。

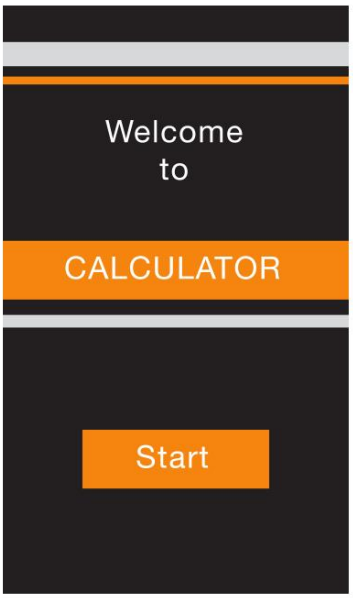
如果你的应用有内存使用问题，或者不重启就无法流畅运行，你必须声明这些问题。关于如何开发一款性能良好的应用，请查阅 iOS 应用编程指南。

尽可能避免使用闪屏或者其他启动体验。用户能够在启动后立即开始使用应用是最好不过的。

推荐（直接启动）



不推荐（使用欢迎屏幕）



避免让用户做过多设置。而应该如此：

- **聚焦在满足 80%的用户需求上。**这样主体用户群就无需设置各种选项，因为你的应用已经默认处于他们想要的状态。如果有些功能有少部分用户想要，换句话说，大部分人不需要的话，就别管它了。
- **尽可能用其他方式获取更多（用户）信息。**如果你能得到用户在内置应用或硬件设置中提供的信息，直接从系统中获取它们，而不需要再次让用户输入。
- **如果你必须获取设置信息，在你的应用中直接向用户询问，然后尽快保存这些设定**（译者注：这段讲的是权限许可，如能否访问照片或者日历或地理位置信息等等）。这样用户就无需强制跳出应用进入系统设置页面了。如果用户需要更改设置，他们可以在任何时候进入应用的设置选项进行修改。

尽可能让用户晚一些再登录。最理想的状态是，用户在无需登录的情况下就能尽量多地浏览内容并使用部分功能。例如，App Store 应用会在用户浏览商品并确定进行购买时，才要求用户进行登录。对于必须登录才能进行后续浏览和操作的应用，用户往往会直接放弃。

如果你的应用必须先登录后使用，那么你应该在登录页面有一些简短的文字，来描述为什么必须先登录，以及这样做会给用户带来什么好处。

谨慎使用新手引导（介绍应用的功能和如何进行操作）。在考虑新手引导之前，你应该完善你的应用，尽可能使应用的功能直观和易于寻找。有句话说得好，*好的应用不需要新手引导*。如果你确实觉得需要新手引导，那么请参考如下的建议，设计一个简洁、有针对性并且不妨碍用户的新手引导。

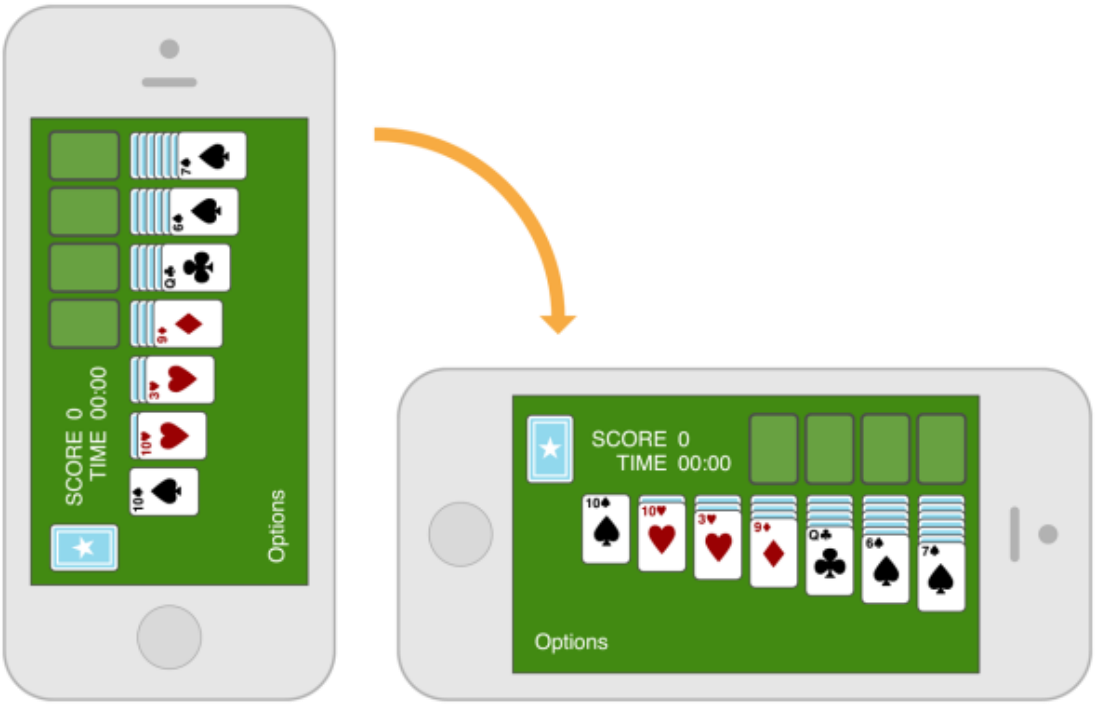
- **只提供开始使用应用所必需的信息。**好的新手引导应该告诉用户接下来第一步应该做什么，或者简短地演示大部分用户感兴趣的一些功能。在能够浏览你的应用之前，如

果用户遇到太多的信息，让用户记住这些不是当前所必须的内容，他们很可能会觉得你的应用很难用。如果在某些特定场景下确实需要一些引导，那么也应该在用户进入这个场景之后再进行。

- **使用交互动画来吸引用户，并让用户通过实际上手来学习如何使用。**对于文字内容的增加应该谨慎，且仅当增加文字对于提升体验有益时才这么做。不要指望用户会阅读大段的文字。例如，可以使用动画而不是文字来描述如何执行一个简单的任务。在引导用户了解较为复杂的任务时，可以通过一些引导浮层来简要说明每一个步骤用户需要做什么。尽可能避免展示应用截图，因为截图是死的，用户可能会混淆截图和应用的实际界面。
- **能够简单地取消或者跳过新手引导。**有些用户看完新手引导之后就不想再看，有些甚至根本就不想看新手引导。请记住用户的选择，不要强迫用户每次打开你的应用都要再做一次选择。

不要太早要求用户去给你的应用评分。过早要求用户进行评分可能会适得其反。如果你想获得用户有价值的反馈和评论，在邀请用户评论之前，请给他们一点时间来使用你的应用，并对你的应用形成印象。例如，你可以等用户访问了一定数量的页面或完成了一定数量的任务之后，再邀请他们进行评价。

一般建议按照屏幕默认的定向方式启动你的应用。尽管如此，如果你的应用只有一种屏幕方向，那么就始终以这个方向启动，让用户在他们自己需要时再改变设备方向。例如，一个游戏或者媒体观看应用只在横屏模式下运行，那么就应该以横屏模式启动，即使设备当前处于竖屏模式。这样的话，如果用户在竖屏模式下打开应用，他们也知道应该把设备转成横屏来进行使用。



注：最好让横屏应用支持两种模式的横屏，即 home 键处于左右两侧的状态。如果设备当前已经处于横向状态，那么就按照当前状态启动应用，除非你有充分的理由不这么做。其

他情况时，可以考虑按 home 键处于右侧的方式启动应用。（想要了解更多关于支持不同设备方向的内容，请参阅 Respond to Changes in Device Orientation.）

准备一张与应用首页看上去一样的闪屏。iOS 会在启动应用时调用这张图，这样可以让用户觉得启动速度很快，降低对等待时间的感知度。具体如何制作闪屏，请查阅 Launch Images。（译者注：Launch Images 章节处在 iOS Human Interface Guidelines 的 Icon and Image Design 部分，翻译将在后续更新中放出，烦请各位耐心等候。）

如果可能，不要让用户在初次启动应用时阅读免责声明或者确认用户协议。你可以直接在 App Store 展示这些内容，使用户在下载前就有所了解；虽然这个办法能最大地减少麻烦，但也不是一直可行。如果在某些情况下你必须展示这些内容，要确保它们与 UI 保持统一并在产品功能与用户体验之间达成平衡。

在应用重启后，需要恢复到用户退出使用时的状态，让他们可以从中断之处继续使用。无需让用户记住是如何达到此种退出状态的。

1.4.2 时刻准备好停止（Always Be Prepared to Stop）

iOS 应用无需关闭或退出选项。当用户切换应用，回到主屏幕或者将设备调至睡眠模式的时候，其实就是停止了当前应用的使用。

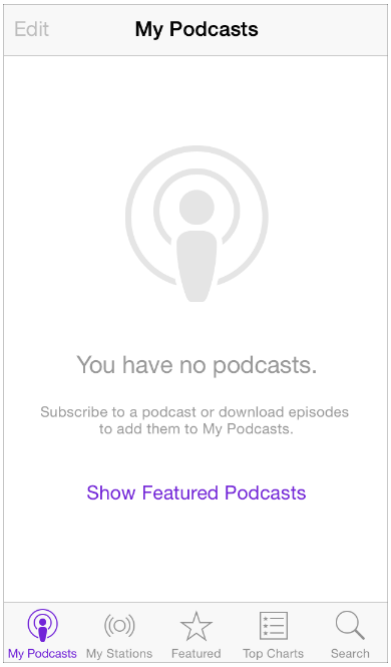
当用户切换应用时，iOS 的多任务系统会将其放置到后台并将新应用的 UI 替换上来。在这种情况下，你必须做到以下几点：

- 随时并尽快保存用户信息，因为在后台的应用随时有可能被终止或退出。
- 当应用停止的时候保存当前状态，使用户可以在回到应用时能从中断之处继续使用。例如，在使用可滚动的数据列表时，退出后保存列表所在的位置。

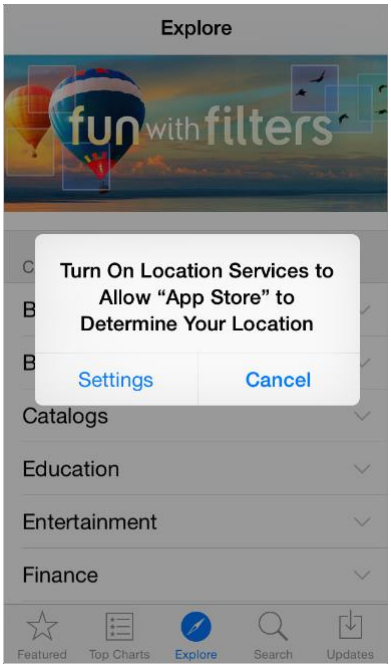
有些应用可能需要一直在后台运行。例如，用户可能希望能在使用一个应用的同时还能一直听歌，接着又想用另外一个应用来检查代办项或者玩游戏。关于如何正确处理多任务，请查阅 Multitasking。（译者注：Multitasking 章节处在 iOS Human Interface Guidelines 的 iOS Technologies 部分，翻译将在后续更新中放出，烦请各位耐心等候。）

不要强制让应用退出。因为这样会让用户误以为是 crash。如果有问题产生，需要告诉用户具体状况以及如何解决。以下有两个建议，取决于出现的问题有多严重而酌情使用：

如果应用中所有的功能当前都不可用，那么应该显示一些内容来解释当前的情形，并建议用户如何进行后续操作。这部分内容给予了用户以反馈，使用户相信你的应用现在没问题。同时这也可以稳定用户情绪，让他们决定是否要采取纠正措施，继续使用应用，还是切换到另一个应用。



如果只有部分功能不可用，那么只要当用户使用这些功能时显示提示即可。不然的话，用户就应该能正常使用应用的其他功能。如果你决定使用警告框来进行提示，请确保只在用户尝试使用不可用的功能时再显示。



1.5 导航（Navigation）

除非导航设计的不合理，不然用户不应明显察觉到应用中的导航体验。放置导航到一个能够支撑你的应用结构和目的却又不过分引起用户注意的状态。

广义来说，有三种主要类型的导航，每种导航都有其适应的应用结构：

- 分层。
- 扁平。
- 内容或经验驱动。

在分层应用中，用户在每个层级中都要选择其中一项，直到目的层级。如果要切换到另一个层级，用户必须回退一些层级，或者直接回到初始层级进行再次选择。系统的设置和邮件应用在这方面是很好的示范，可以参考他们。

查看演示视频

在扁平应用中，用户可以从一个主要分类直接切换到另一个，因为所有的主要分类都可以从主屏直接访问。音乐和 App Store 是两个使用扁平结构的好例子。

查看演示视频

在内容驱动或经验驱动信息结构的应用中，导航的内容也会根据内容或经验来进行设计。例如，在阅读一本电子书时，用户会一页接一页地进行阅读，也会在目录中选择想要阅读的页码跳转后开始阅读。同样的，在游戏应用中，导航的作用也非常重要。

查看演示视频

在某些情况下，在一个应用中结合多种导航类型会有很好的效果。例如，对于扁平信息结构中某一分类下的内容，用分层导航的方式来显示可能会更好。

用户应该时刻清楚自己当前在应用中所处的位置，以及如何前往他们所想到的页面。无论导航类型是否适合你的应用结构，最重要的是用户访问内容的路径应该是合理、可预期和易于寻找的。

UIKit 定义了一些标准的 UI 元素，这些元素即可以构建分层或扁平的导航，也可以实现以内容为中心的导航，例如电子书或者媒体观看类应用。游戏或者其他经验驱动的应用通常需要一些自定义的元素和行为。

使用导航栏（Navigation Bar）帮助用户轻松访问分层内容。导航栏的标题可以显示用户当前所处的层级，而后退按钮可以回到上一层级。查看 Navigation Bar 了解更多。

使用标签栏（Tab Bar）显示同类型的内容或功能。标签栏很适合于扁平信息结构，可以让用户在分类之间随意切换，而不用在意当前所处的位置。查看 Tab Bar 了解更多。

在应用中，如果每屏显示的都是同类项或同类页，可以使用页面控件（Page Control）。页面控件的优点是可以直观地告诉用户共有多少个项目或页面，以及当前所处的位置。查看 Page Control 了解更多。

一般来说，最好能给用户到达每一屏的路径。如果用户需要，就应该考虑使用临时视图，例如模态视图、动作菜单或警告框。查看 Modal View、Action Sheet 和 Alert 了解更多。

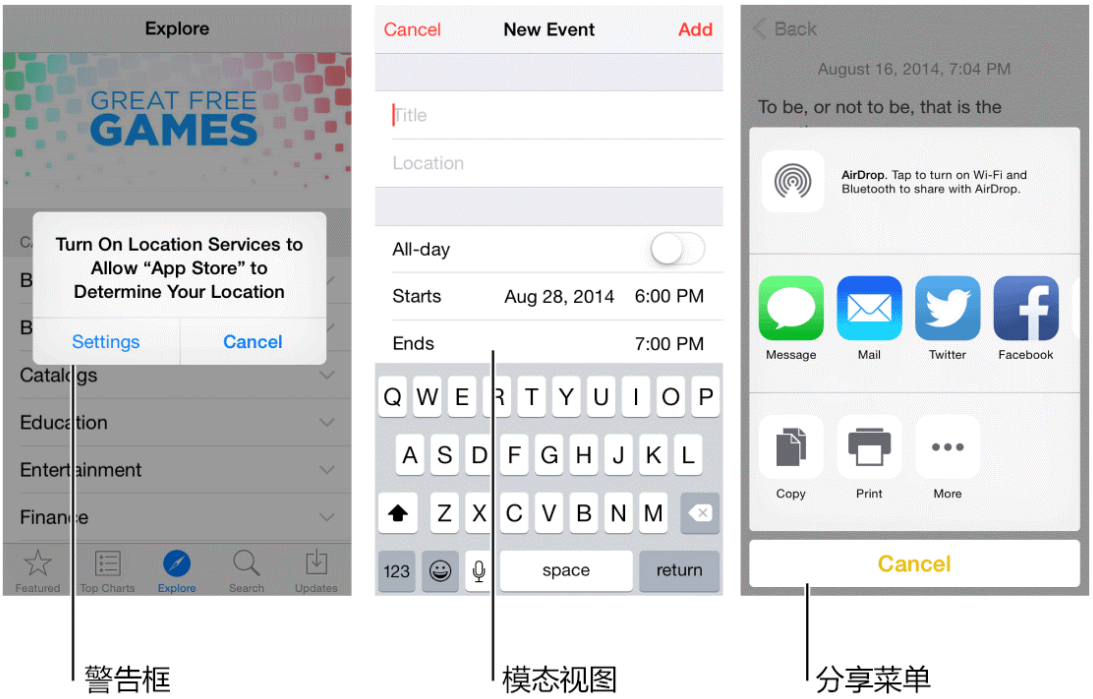
（译者注：上文提到的章节均处在 iOS Human Interface Guidelines 的第 4 章，翻译将在后续更新中放出，烦请各位耐心等候。若有需要，亦可先参考先前已翻译的 iOS7 UI Elements 章节：上，下。）

UIKit 同时还提供了以下相关控件：

- 分段控件（Segmented Control）。分段控件让用户在一屏内就可以查看到不同分类的内容，而不需要切换到其他屏幕。
- 工具栏（Toolbar）。尽管工具栏看起来和导航栏或标签栏相似，但是工具栏不具导航作用。相反，工具栏为用户提供了可以控制当前屏幕内容的控件。

1.6 模态情境（Modal Contexts）

模态是一个承载某些连贯操作或内容的优缺点并存的模式。它可以给用户提供一种不脱离主任务的方式去完成一个任务或者获得信息，但是也会临时性地阻止用户对应用的其他部分进行交互操作。



理想情况下，用户可以与 iOS 应用进行一种非线性的交互，所以，尽可能减少应用中的模态体验是最好的。通常情况，在以下情形下可以考虑使用模态情境：

- 必须引起用户关注的时候。
- 一个独立的任务需要完成或者很明确需要被放弃，为了避免在模棱两可的状态下遗漏用户信息的时候。

保持模态任务的简单，简短和高度聚焦。你肯定不希望用户使用模态视图时像使用应用中的一个 mini 应用一样。如果子任务过于复杂，用户会在进入模态情境时忽略主要任务。在设计一个涉及视觉层次的模态任务时特别要考虑这一点，因为用户有可能迷失并且忘记如何回到之前的操作中去。如果一个模态任务必须包含不同视图的子任务，确保给用户一个独立、清晰的导航路径，并避免迂回。

始终提供明显、安全的途径退出模态任务。确保用户在退出模态视图时可以预期操作的结果。

一个任务需要多层级的模态视图时，确保用户理解点击完成按钮的结果。点击一个低层级视图上的完成按钮是完成这个视图中任务的一部分，还是整个任务？因为存在这种困惑的可能性，所以要尽可能避免在下级视图中添加完成按钮。

保证提醒对话框的内容都是重要且可操作的。提醒对话框会打断用户的体验并且要点击才会消失，所以要让用户感到提醒信息是有用的，打断是有价值的。

尊重用户关于接收通知的选择。用户会设置接收应用通知的形式，必须尊重要用户的喜好设置，否则可能触怒用户，导致其关闭所有的推送通知。

1.7 交互性和反馈（Interactivity and Feedback）

1.7.1 用户知道标准手势（Users Know the Standard Gestures）

用户使用点击、拖拽、捏合等手势与 iOS 设备进行交互。使用手势拉近了用户和设备之间的距离，并且增强了直接操纵感。用户通常期待手势在不同应用之间都是通用的。

除了用户熟悉的标准手势，iOS 还定义了一些系统范围内的操作，例如呼出控制中心或消息中心。在任意应用下都可以使用这些操作。

不要给标准手势赋予不同的行为。除非你的应用是游戏，否则重新定义标准手势会使用户迷惑，且增加使用难度。

不要创建和标准手势功能相似的手势操作。用户已经习惯了标准手势的行为，没有必要让用户学习达到同样效果的不同操作。

可以用复杂手势作为完成某任务的快捷方式，但不能是唯一触达方式。最好给用户提供一些简单、直接的方式完成某操作，即使这种方法需要额外的动作。简单的手势能让用户集中于当前的体验和内容，而不是交互操作本身。

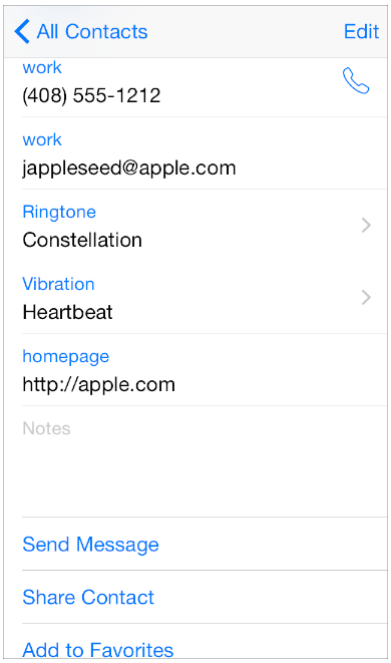
除非是游戏，否则避免定义新的手势。在游戏或其他沉浸式的应用中，操作手势也是有趣体验的一部分。但是在普通应用中，帮助用户达成目标要比操作本身重要的多，所以最好使用标准手势，尽量避免让用户去发掘和记忆新的操作。

在特定的环境中，可以考虑使用多指操作。虽然复杂的操作并不一定适用于所有应用，但对用户会花大量时间使用的应用来说可以丰富体验，例如游戏或创建内容环境。但因为非标准手势可发现性差，要尽量少用，并且不要让这类手势成为完成任务的唯一方式。

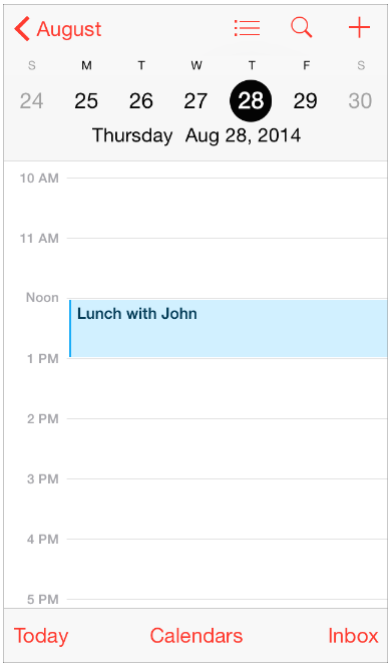
1.7.2 可交互元素吸引用户点击（Interactive Elements Invite Touch）

为了暗示交互性，设计时会使用很多线索，包括颜色、位置、上下文、表意明确的图标和标签等。并不需要过于修饰元素来向用户展示可交互性。

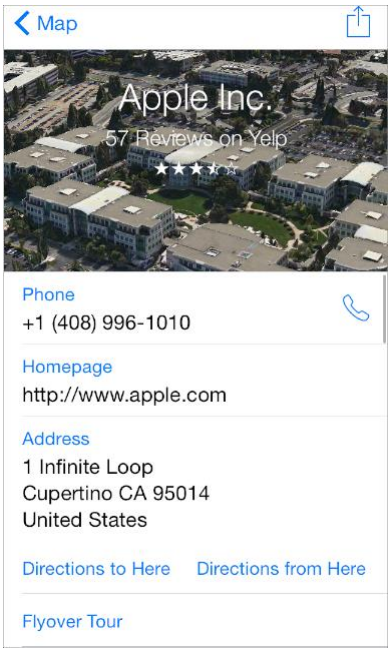
一个关键的颜色可以给用户提供很强的视觉指引，尤其是没有冗余的其它颜色时。为了有对比，使用蓝色标记可交互的元素，并且使用统一的、易识别的视觉风格。



返回按钮使用多个线索指明其可交互性并传达其功能：它出现在导航中，显示了一个指向后方的图标，使用了关键色，显示了上一级页面的标题。



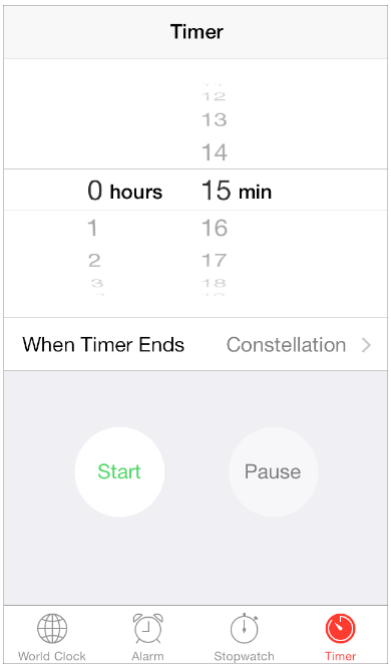
一个图标或者标题提供了清晰的名称指引用户点击它。例如，地图中的标题"立交桥路线"，"定位到这里"，清晰地说明了用户可做的操作。结合关键色，就可以省去按钮边界或其他多余的修饰。



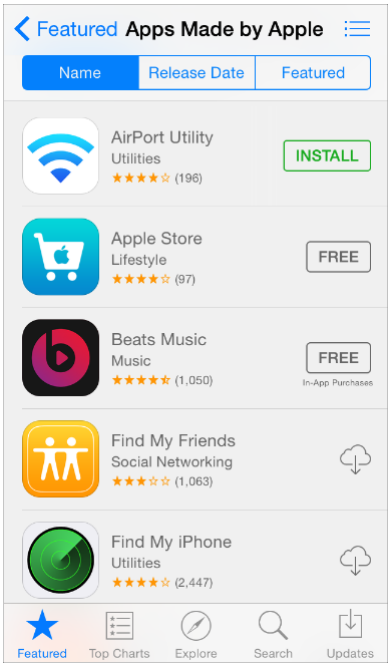
在内容区域，有必要给按钮添加边界或背景。操作条中的按钮、动作表单和提醒对话框可以不需要边界，因为用户知道在这种区域中大多数选项是可交互的。但是在内容区域，按钮有必要使用边界或背景将按钮从其他内容中区分出来。例如，系统自带的音乐、时钟、照片和 App Store 应用会在一些特别的场景中使用这种按钮。

照片应用中给分享按钮增加了边框，与其他解释性文本进行了区分。

时钟应用在秒表和计时页面中给按钮增加背景来强调开始和暂停按钮，并且使按钮在周围的内容中更容易点击。



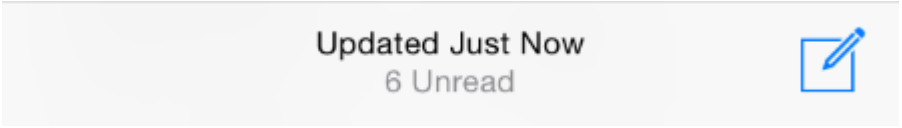
App Store 应用中使用有边界的按钮，将按钮和整个内容条区分开来，点击整条内容查看详细信息，点击按钮进行下载安装。



1.7.3 反馈有助于理解（Feedback Aids Understanding）

反馈会帮助用户了解应用当前在做什么，发现接下来可以做什么以及理解动作产生的结果。UIKit 提供了很多反馈。

尽可能将状态或其他的反馈信息整合到 UI 中。用户不进行操作或不跳出当前内容就能获得需要的信息是最好的。例如，邮箱应用将当前邮箱的状态显示工具条上，这样就不会影响当前内容。



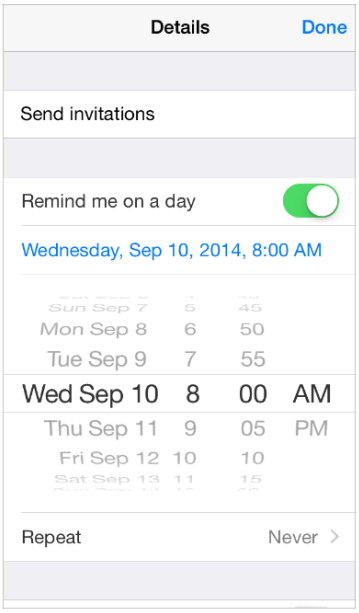
避免显示不必要的警告框。警告框是一种很强的反馈机制，只有在传递非常重要也是理论上可行的信息时才需要使用它。如果用户常看到很多不是重要信息的警告框，他们很快就会忽略所有对话框提醒。

1.7.4 输入信息的方式要简单（Inputting Information Should Be Easy）

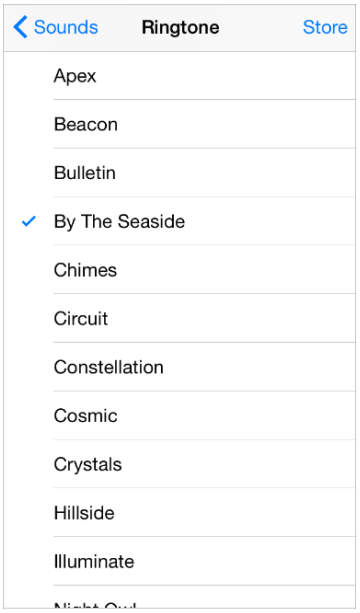
不管用户是点击控件还是使用键盘，输入信息都会花费时间和精力。如果发挥有用的效用前就让用户输入大量信息会减弱用户继续使用的欲望。

让用户更容易地进行选择。例如，使用选择器或者表格代替纯文本，避免要求用户打字来提高选择效率，降低选择成本。

提醒中的时间日期选择器



设置中的一列选项



适宜地从 iOS 中获取信息。设备上存储了大量的用户信息。可以的话，不要让用户提供你可以轻易找到的信息，例如联系人或日历信息。

提供有用的反馈来平衡用户的输入。付出和回报的概念可以帮助用户感到进程在被推进。

1.8 动画（Animation）

iOS 的用户界面中遍布着细微、精美的动画，它们使得应用的体验更具吸引力、更具动态性。适当的动画可以：

- 传达状态和提供反馈
- 增强直接操作的感觉
- 帮助人们可视化他们的操作结果

查看演示视频

谨慎地增加动画，特别是在那些无法提供沉浸性体验的应用中。看起来过多的无理由的动画会阻碍应用的流畅性，降低性能，还会分散用户在任务中的注意力。尤其要说的是，要有目的和限制性地使用运动效果和 UI 组件中的动态行为，并确保对结果进行测试。一旦被

合理的使用，这些效果能提高用户的理解度和愉悦度；过度使用他们则会使应用看起来很迷惑，很难控制。

在合适的时候，使自定义的动画与内置动画保持统一。人们习惯于谨慎添加动画，尤其是在那些不能提供沉浸式用户体验的应用中。如果应用主要关注一些严肃的任务或者生产性任务，那么动画就显得多余了，还会无端打乱应用的使用流程，降低应用的性能，让用户从当前的任务中分心。

开发者的自定义动画应该切合内置 iOS 应用的动画。用户习惯于内置 iOS 应用使用的精细动画。事实上，用户趋向于把视图之间的平滑转换，对设备方向改变的流畅响应和基于物理力学的滚动效果看作是 iOS 体验的一部分。除非你的应用能够给用户沉浸式的体验–比如游戏–自定义动画应该可以与内置应用的动画相媲美。

使用风格类型一致的动画。在应用中使用风格类型一致的动画非常重要，可以让用户构建基于使用应用获得的用户体验。

大多数情况下，恰当一点的做法是让自定义动画更具现实性。用户乐意于接受自由的艺术创作，但当你的动画违背物理定律和自然法则的时候，他们会感觉到非常迷惑。

1.9 品牌推广（Branding）

品牌推广并不仅仅是在应用中展示品牌的颜色和 logo。理想状态下，你开发的某个特定品牌的应用应该通过创建独特的外观和感觉来为用户提供难忘的体验。

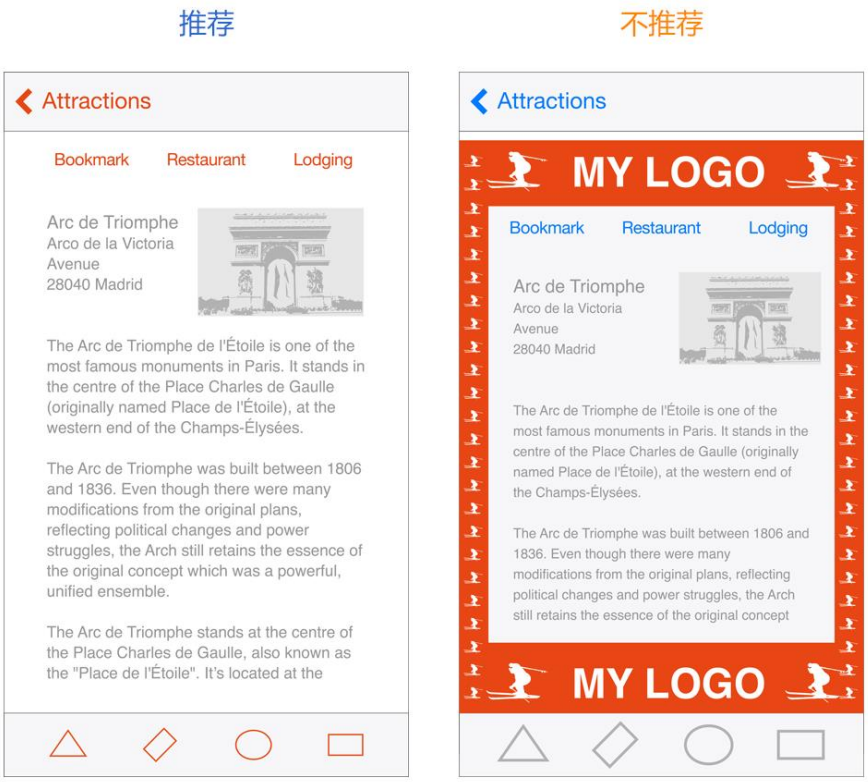
在 iOS 系统之下可以很容易地使用自定义的图标、颜色和字体来创建区别于其他应用的 UI。当你进行这些元素的设计时，牢记以下两点：

- 每个自定义的元素本身都需要具备良好的观感和功能性，但它也应该与应用中其他元素保持一致，无论应用中其他元素是自定义的还是标准的。
- 为了在 iOS 中感觉舒适，你的应用虽然不必看起来跟内置的一样，但是需要对它的遵从、清晰度和深度（如欲了解更多，参见 1.1 为 iOS 而设计（Design for iOS））进行整合。花些时间弄清楚在你的应用中遵从、清晰和深度所代表的意味，并把它们在你的自定义元素中表达出来。

当你需要让用户意识到你的品牌时，你应该遵循以下几点：

以精致优雅不唐突的方式植入品牌的颜色和图片。用户使用你的应用来完成事务或者进行娱乐，他们不希望被强迫着去观看广告。为了获得最好的用户体验，你可以通过字体、颜色和图像的设计来潜移默化地地提醒用户你的品牌身份。

避免远离用户关心的内容。比如，在屏幕顶部展示一个二级栏目，仅用来展示品牌资产，这意味着内容没有足够的空间，可以考虑以其他低侵入性的方法无处不在地展示品牌，比如巧妙地定制屏幕的背景。



抵抗住诱惑，不要把你的 logo 贯穿整个应用。移动设备的屏幕多数相当小，logo 的每一次出现都会占据空间而将用户与他们想看的内容隔离开。而且，在应用中显示 logo 并不能像在网页中显示 logo 那样达到相同的目的：对于用户来说通常会很容易在不知道网页所属的情况下访问一个网页，但却极少有用户会在完全不看一个 iOS 系统中的应用图标的情况下就打开它。

1.10 颜色与字体（Color and Typography）

1.10.1 色彩有助于增进沟通（Color Enhances Communication）

在 iOS 系统中，颜色会用于表征交互，传递活性以及提供视觉连续性。内置的应用程序选择使用那些看起来更具个性的、纯粹、干净的颜色，并辅以或亮或暗的背景组合。

如果你要创建多样的自定义颜色，要确保它们能够和谐共存。例如，如果你的应用的基本风格是柔和的色调，你就应该创建一个协调的柔和色调的色板用于整个应用。

注意在不同情境下的颜色对比。例如，如果在导航栏的背景与栏按钮标题之间没有足够的对比，按钮就会很难被用户看到。 依据经验的法则来说，需要区分的颜色必须至少存在50%的亮度差异。（我们）需要将设备置于不同的光照环境之中（包括晴朗的室外）来测试设备上的观感效果。

提示：一种发现需要更高对比度的区域的方法是降低 UI 的饱和度并在灰度模式下查看它。如果在灰度版本中你很难区分可交互与非可交互元素或背景等，你有可能需要增加这些元素之间的对比度。

当你使用自定义的栏颜色时，着重考虑半透明的栏和应用内容。当你需要创建能匹配特别颜色的栏颜色时（比如一个已有品牌中的颜色），可能在你获得你想要的结果之前，你需要用各种颜色进行实验。栏的显示将会同时受到 iOS 系统所提供的半透明栏与藏在栏后面的应用内容的呈现所影响。

API 注释：使用浅色（TintColor）的属性值给予栏按钮颜色，使用栏浅色（BarTintColor）的属性值为栏本身赋色。欲了解更多关于栏属性的内容，可参见 *UINavigationBar Class Reference*,，*UITabBar Class Reference*，*UIToolbar Class Reference* 和 *UISearchBar Class Reference*。（译者注：相关章节翻译将在后续更新中放出，烦请各位耐心等候。）

注意颜色的盲区。多数色盲的人很难区分红色与绿色。需要对你的应用进行测试以确保在其中你没有将红色与绿色作为区分两个不同状态或值的唯一方式，一些图像编辑软件或工具能够有效的帮你验证颜色的盲区。通常意义来说，使用多种方式来表征原色的交互性是非常好的（需要了解更多关于在 iOS 系统中表征交互性的信息，详见 Interactive Elements Invite Touch）。

考虑选择一种基准色颜色来表征交互性与状态。在内置的应用中基准色有比如备忘录中的黄色与日历中的红色等。如果你定义一种用于表征交互和状态的基准色，要确保你的应用中的其他颜色不会与它发生冲突。

色彩可以向用户传达信息，但不一定会以你希望的方式。每个人眼中的色彩是不一样的，不同的文化为色彩赋予的意义也是不相同的。花时间来研究如何使用色彩才可能会被其他国家或者文化接受。你要尽可能确定应用中运用的色彩向用户传达了恰当的信息。

大多数情况下，不能让颜色喧宾夺主，让用户分心。除非色彩是应用的目的和本质所在，通常情况下色彩应该用来从细微细节之处提升用户体验。

1.10.2 文字应该清晰易读（Text Should Always Be Legible）

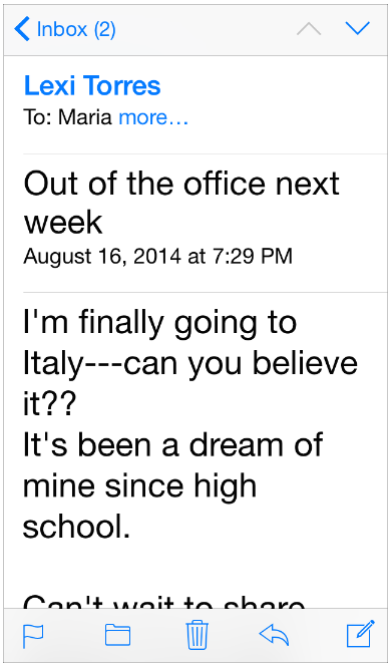
文字首先必须是清晰可辨的。如果用户不能看清楚应用中的字词，那么文字再好看也是没是无意义的。当你在你的应用中采用 Dynamic Type 时，你可以实现：

- 能自动调整文字的粗细，字母间距以及行高。
- 为语义上有区别的文本模块指定不同的文本样式，比如正文、脚注或者标题。
- 文本可以根据用户在动态文字和可访问性设置中指定字体大小的变化作出适当的响应。

注：如果你是用自定义字体，你仍然可以依据系统的字号设置来规划字体范围。当用户改变设置时，你的应用也必须响应式的配合。

就你而言，要采用 Dynamic Type 需要一些工作。为了学习如何使用文字样式并确保当用户改变文字型号设置时你的应用能够获取通知，可以参考 *Text Styles* in *Text Programming Guide for iOS*。（译者注：相关章节翻译将在后续更新中放出，烦请各位耐心等候。）

文本尺寸的响应式变化需要优先考虑内容。并不是所有的内容对于用户都是同等重要的。当用户选择更大的文本尺寸时，他们是想要使他们关注的内容更容易阅读；他们并不总是想要屏幕上的每个单词都更大。



例如，当用户选择具备更大易用性的文本尺寸时，邮件将会以更大的尺寸显示邮件的主题和内容，而对于那些没那么重要的信息——如时间和收件人——则采用较小的尺寸。

在适当的情况下，当用户选择一个不同的文本尺寸时要调整页面布局。例如，当用户选择小的文本尺寸时，你可能想将内容由一列的布局方式改为两列。如果你决定根据不同的文本尺寸调整布局，你可以选择针对尺寸的子集来实现——如包含小，中和大尺寸——而不是对于每个可能的尺寸都进行布局的调整。

确保一个自定义字体在不同尺寸下的所有类型都具备可读性。实现这一效果的方法之一是效仿在不同的文本尺寸下 iOS 系统呈现字体样式的一些方法。例如：

- 文本永远都不应该小于 11 点（points），即使是用户选择极小的文本尺寸。相较而言，内容样式使用 17 点的字号作为大尺寸的默认文本尺寸设置。
- 通常来说，字号与行距值在每一档的文本尺寸设置中差别为 1 点。唯一例外的是两种标题的样式，它们被应用在极小、小和中尺寸的设置中，使用了相同的字号、行距和字距。
- 在最小的三种文本尺寸中，字间距相对较大；而在最大的三中文本尺寸中，字间距相对紧凑。
- 标题和内容的样式使用相同的字体尺寸，同时，为了区分标题与内容样式，标题样式使用更重的值。
- 导航控制栏的文本使用相同的字号，而内容文本的样式则使用大尺寸的设置（值为 17 点）。
- 文本总是使用常规或者中重，一般不适用轻或者加粗。

通常情况下，应用中整体应该使用单一字体。多种字体的混杂会使你的应用看上去支离破碎和草率。相反，使用一种字体和少数样式。根据语义用途，使用 UIFont 类的 API 来定义不同文本区域的样式，比如正文或者标题。

1.11 图标和图形（Icons and Graphics）

1.11.1 应用图标（The App Icon）

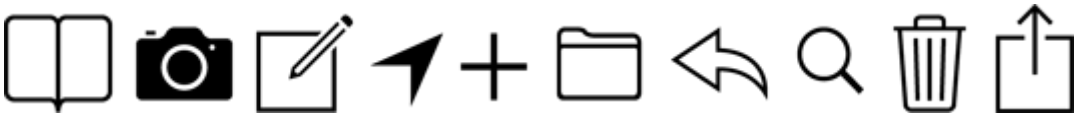
每个应用都需要一个漂亮的图标。用户常常会在看到应用图标的时候便建立起对应用的第一印象，并以此评判应用的品质、作用以及可靠性。

以下几点是你在设计应用图标时应当记住的。当你确定要开始设计时，请参考 App Icon 来获取更详细的设计规格与指导。（译者注：App Icon 章节处在 iOS Human Interface Guidelines 的 Icon and Image Design 部分，翻译将在后续更新中放出，烦请各位耐心等候。）

- 应用图标是整个应用品牌的重要组成部分。将图标设计当成一个讲述应用背后的故事，以及与用户建立情感连接的机会。
- 最好的应用图标是独特的，整洁的，打动人心的，让人印象深刻的。
- 一个好的应用图标应该在不同的背景以及不同的规格下都同样美观。为了丰富大尺寸图标的质感而添加的细节有可能让图标在小尺寸时变得不清晰。

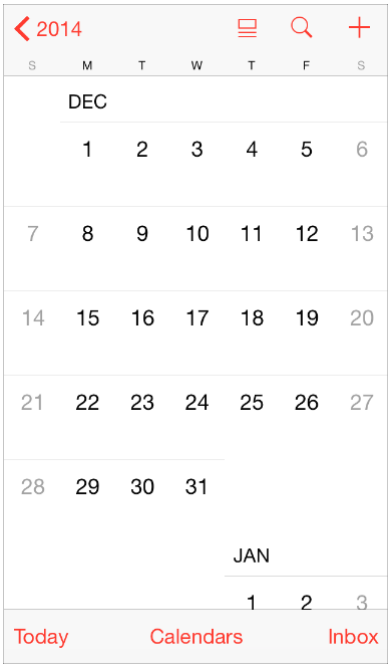
1.11.2 栏图标（Bar Icons）

iOS 提供了一系列小的 icon，用以代表各种常见任务与操作，它们常用在标签栏(Tab Bar)、工具栏(Toolbars)与导航栏(Navigation Bar)中。用户通常都已经了解这些内置图标的含义了，因此可以尽可能的多使用它们。



如果需要自定义动作或者内容，你也可以设计自定义图标。设计这些小的线性图标与设计应用图标有很大的区别，请参考 Bar Button Icons 来了解更多内容。（译者注：Bar Button Icons 章节处在 iOS Human Interface Guidelines 的 Icon and Image Design 部分，翻译将在后续更新中放出，烦请各位耐心等候）

请注意，你有时候也可以用文字来代替工具栏和导航栏的图标。 就像 iOS 的日历里面，工具栏上便是使用"今天"、"日历"和"收件箱"来代替图标进行表意的。



想要决定在工具栏和导航栏中到底是用图标还是文字，可以优先考虑一屏中最多会同时出现多少个图标。如果数量过多，可能会让整个应用看起来难以理解。使用图标还是文字还取决于屏幕方向是横向还是纵向，因为水平视图下通常会拥有更多的空间，可以承载更多的文字。

1.11.3 图形（Graphics）

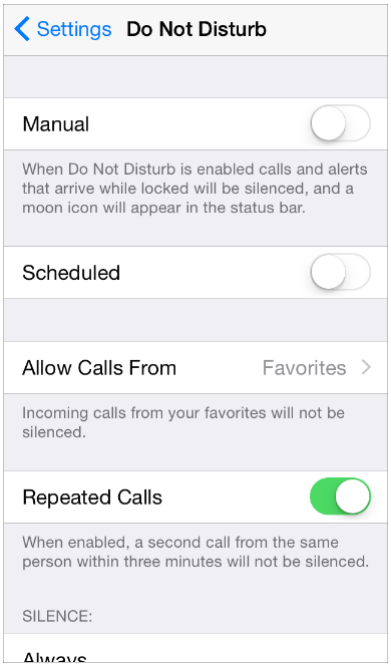
iOS 应用大多数图形丰富。无论是你需要展示用户的照片，还是需要创建自定义图片，以下这些需求都应该遵守：

- **支持 Retina 显示屏。**确保你应用中的所有图片资源都提供了高分辨率规格。尤其需要注意的是，iPhone 6 Plus 需要提供@3x 规格的图片，而所有其他的高分辨率 iOS 设备都需要提供@2x 规格的图片。
- **显示照片或图片时请使用原始尺寸，并不要将它拉伸到大于 100%。**你不会希望在你的应用中看到拉伸和变形的图片。可以让用户自己来选择他们是否想要缩放图片。

不要使用带有苹果符号与版权的图片。这些符号都拥有版权，并且产品的设计可能会经常改变。

1.12 术语和措辞（Terminology and Wording）

你在应用中呈现的每一个字都是与用户进行对话的一部分。把握这样的对话机会，为你的用户提供清晰的表意与愉悦的体验。



设置是面向全体用户的一个基础应用，它使用了简明扼要的语言来描述了用户可以进行的操作。举个例子，设置→勿扰模式（Do Not Disturb）就没有使用难以理解的复杂术语，而是用了简单的语言，给用户描述了里头的一系列操作。

保证你使用的术语是用户能理解的。根据你对用户群的理解来决定在应用中使用什么样的词汇。举个例子，在一款针对小白用户的应用中使用技术术语是不合适的，但对于针对高端用户的应用来说，使用技术术语是很自然的事情。

使用非正式的友好语气，但不需要太过低三下四。避免太正式太僵化，或者太过嘻嘻哈哈，傲慢无礼。请记住，用户可能会反复阅读这些文本，因此有些起初看上去很俏皮的语句，多看几次之后可能会显得幼稚和烦人。

像新闻编辑一般遣词造句，避免不必要的冗余语句。当你的文案足够简明扼要，用户就可以很轻松地阅读和理解它。确定最重要的信息，精炼它并且突出它，让用户不需要读一大段文字才能了解他们在找什么，以及下一步要做什么。

给控件加上短标签或者容易理解的图标。让用户只扫一眼就能知道这个控件是干什么的。

描述时间时要注意准确性。今天和明天这些词汇确实显得比较友好，但有时候会让用户费解，因为你可能没有办法确定用户当前的时区和时间。举个例子，假如有一项活动会在半夜 12 点前开始，对于在同一个时区的用户而言，这个活动是在今天开始的，但对于那些在早一点的时区里的用户而言，这个活动在昨天就已经开始了。

为你的应用写一则漂亮的 App Store 描述，最大程度地把握住这个与潜在用户沟通的绝佳机会。除了准确描述你的应用、强调应用的品质与亮点以外，你还需要：

- **修正所有的拼写、语法与标点符号错误。**这些小错误也许不会影响用户正常使用，但是可能会让他们对应用的整体品质产生负面印象。
- **尽量少用全大写的词汇。**虽然大写单词有时候可以吸引注意力，但是全大写的段落不适合阅读，而且会产生一种朝用户扯着嗓子吼的感觉。
- **可以描述 bug 修复情况。**如果你的应用新版包含用户一直期待的 bug 修复，那在你的软件描述中提到这一点就是个很好的做法。

1.13 与 iOS 的整合（Integrating with iOS）

与 iOS 整合，指的是在当前平台上给用户提供一种舒适的、宾至如归般的体验，当然这并不意味着我们要把每一个应用做的和内置应用一模一样。

最好的与 iOS 整合的方式便是深刻地了解 iOS 的主题与核心——这一部分在上文为 iOS 而设计（Designing for iOS）部分中已有详细描述，并寻求出如何在你的应用中融合与表达这种主题。当你这么做的时候，遵循本章中的指引可以帮助你为你的用户提供他们想要的体验。

1.13.1 正确使用标准 UI 元素（Use Standard UI Elements Correctly）

尽可能使用 UIKit 提供的标准 UI 元素。多使用标准元素而非自定义元素，你与你的用户都将受益：

- 标准 UI 元素会根据 iOS 官方的更新而自动更新——而自定义元素不会。
- 标准 UI 元素对于你自定义外观和行为来说拥有优秀的扩展性。举个例子，iOS 中所有的视图（Views）都是可自定义颜色的，它让应用配色变得很简单。想要了解更多如何给 UI 元素定义颜色，可以参考 iOS 7 UI Transition Guide 中 Using Tint Color 的相关章节。
- 用户更熟悉和习惯标准的元素，因为这对于他们来说没有学习成本，他们可以立刻明白这些元素的用途。

想要充分地利用标准 UI 元素的优点，有一些关键点需要特别注意：

严格遵循每个 UI 元素的设计规范。当你应用中的 UI 元素的外观与功能都是用户所熟悉的，他们可以很容易地根据先前的经验来使用他们，进而更好地使用你的应用。你可以从这些章节中找到各种 UI 元素的设计规范：Bars, Content Views, Controls, Temporary Views.

（译者注：上文提到的章节均处在 iOS Human Interface Guidelines 的第 4 章，翻译将在后续更新中放出，烦请各位耐心等候。若有需要，亦可先参考先前已翻译的 iOS7 UI Elements 章节：上，下。）

不要混用不同版本的 iOS 里的 UI 元素。你一定不希望让用户觉得你的 UI 元素来自于与当前设备版本不同的 iOS 系统。

大体来说，请避免创造自定义 UI 元素来表现标准交互行为。先问问你自己为什么一定要创建一个与标准 UI 元素行为完全相同的自定义元素。如果你只是想改变标准 UI 元素的外观，可以考虑使用 UIKit 外观定制 API（UIKit appearance customization APIs），或者给元素填充别的颜色。如果你需要定义一个与标准控件稍有不同的行为，请确保你在改变了这个 UI 元素的属性和行为之后，这个元素仍然能完成你所希望的操作。

不要用系统自带的按钮和图标表达其他含义。iOS 提供了多种可用的按钮和图标。请确认你了解它们的准确表意；不要单纯凭借你看到这些图标样式的猜测和理解来解读和使用它们。（你可以在 Toolbar and Navigation Bar Buttons 和 Tab Bar Icons 中了解到这些按钮和图标的准确含义。）

如果你所需要的功能无法用系统提供的按钮和图标来表现，你也可以设计自定义按钮。自定义按钮的设计可以参考 Bar Button Icons。

（译者注：上文提到的章节均处在 iOS Human Interface Guidelines 的第 4 章，翻译将在后续更新中放出，烦请各位耐心等候。若有需要，亦可先参考先前已翻译的 iOS7 UI Elements 章节：上，下。）

如果你的应用是沉浸式体验，那么创造全新的自定义 UI 是合理的。因为你在创造一个统一的体验环境，让用户在其中能够有所期待并探索如何控制应用。

1.13.2 弱化文件和文档处理（Downplay File and Document Handing）

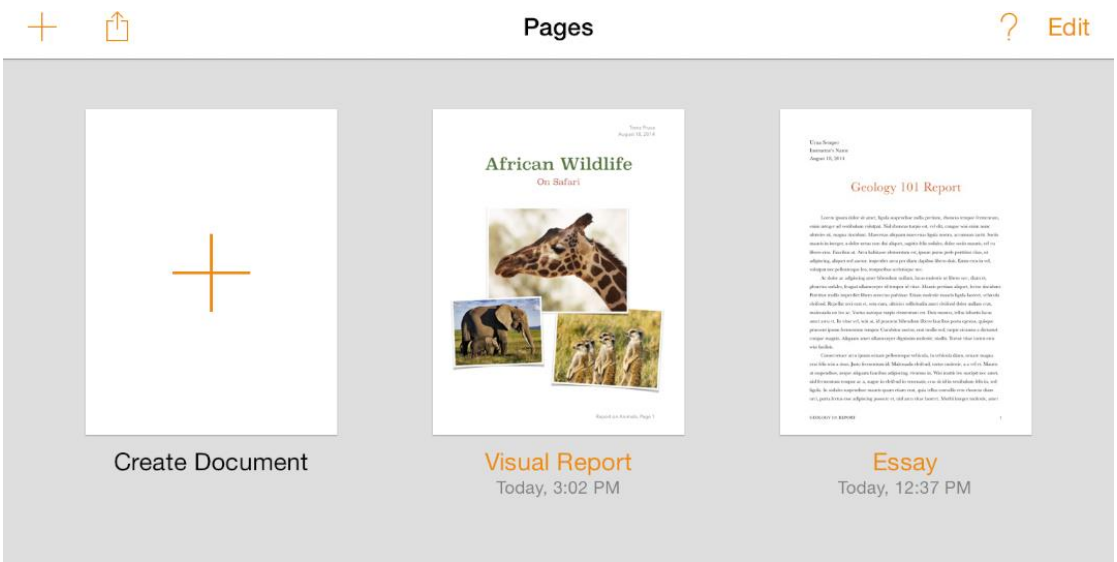
iOS 应用可以帮助用户创建和处理文件，但这并不意味着用户需要过分考虑 iOS 设备的文件系统如何运作。

如果你的应用中支持用户创建和编辑文档，那么提供一个清晰的图形库视图（document library view）让用户能够方便地打开或者新建文档是一个好的做法。理想状况下，这样的图形库视图拥有以下特征：

- **高度图形化。**用户通过屏幕上的缩略图就可以一目了然，快速找出自己想要的文件。

- **让用户用最少的动作完成自己的任务。**比如说，用户可以快速地水平滚动文件列表，然后轻点一下自己想要的文件来打开它。
- **包含新建文档功能。**一个图形库视图应该支持让用户点击一个新建文档的占位图便完成新建文档操作，而不是让用户通过访问别的地方来新建文档。

举个例子，Pages 应用的图形库视图里面，既展示了用户已存在的文档，也加入了便捷的新建文档操作。



如果你的应用允许用户使用在其他应用中创建的文档，你可以通过模态文档选择视图控制器(modal document picker view controller)来帮助用户触达它们。这个控制器可以提取用户在 iCloud 中的文档，还可以通过文档提供者扩展（Document Provider extensions）来提取在其它应用中创建和储存的文件。想要了解更多文档提供者扩展的内容，可以参考 Document Provider Extensions；想要了解更多文档提取视图控制器，请参考 *Document Picker Programming Guide*.

给用户足够的信心，让他们相信除非主动取消或者删除，他们的成果会被随时妥善保存。

如果你的应用帮助用户创建于管理文档，不能要求用户每次都能及时保存。无论是打开另一个文档或切换应用的时候，iOS 应用都应当承担起帮助用户保存输入内容的责任。

如果你的应用的主要功能不是创造内容，但又允许用户查看或编辑信息，这种情况下你需要询问用户是否要保存修改。在这种场景下，比较好的做法是提供“编辑”按钮，点击后进入编辑状态，同时编辑按钮变成“保存”和“取消”按钮，这种变化可以提示用户当前处于编辑模式。“保存”可以保留修改内容，“取消”则退出编辑模式。

1.13.3 必要时提供可配置选项（Be Configurable If Necessary）

某些应用需要用户手动安装或设置选项，但是大部分应用不需要如此。一个好的应用可以让大部分用户快速上手，并通过主界面给用户提供便捷的调整体验的方式。

当你的应用在默认状态下就能满足大部分用户的期望，用户对设置的需求就减少了。如果你需要储存用户的基本资料，可以优先向系统请求和拉取相关信息，而不是上来就让用户自己填写它。如果你一定要提供用户鲜少用到的设置项，请参考 *App Programming Guide for iOS* 中的 The Setting Bundle 部分来了解如何在代码中定义它们。

尽可能在主界面提供设置选项。如果用户在进行主线任务时有可能频繁改变设置，将设置项放在主 UI 中会很方便。如果用户只是偶尔才会用到设置项，那么可以将其放在独立的视图中。

如果应用内相关设置需要在系统设置中改变，帮助用户直接访问系统设置。尤其是，如果你要用一段文字来描述如何改变这个设置，比如"设置>隐私>定位服务"，倒不如直接放置一个按钮，点击后即可到达设置中的定位服务。想要了解如何实现，请参考 Settings Launch URL.

1.13.4 充分利用 iOS 技术（Take Advantage of iOS Technologies）

iOS 提供了丰富的技术方式来支持用户完成他们所期望的各种任务和场景。这意味着在绝大多数情况下，将系统提供的技术整合到你的应用中，往往比自定义一种新的技术更为可靠。

某些 iOS 技术，比如多任务并行（Multitasking）与语音向导（VoiceOver）等等，是所有应用都应该包含的系统级特性。而另外一些技术是否整合到应用中，则取决于应用本身的功能性。比如处理门票和礼品卡的应用（Passbook）支持用户通过 In-App Purchase 完成购买，展示应用内置广告（iAd Rich Media Ads）则可以整合 Game Center，同时支持 iCloud。