

iOS7人机界面指南

界面设计基础（目录）

- [为iOS而设计](#)
- [iOS应用解析](#)
- [起始与停止](#)
- [布局](#)
- [导航](#)
- [模态对话](#)
- [交互性和反馈](#)
- [术语和措辞](#)
- [动画](#)
- [排版和颜色](#)
- [图标和图形](#)
- [品牌](#)
- [与iOS的整合](#)

为iOS7而设计

iOS7 的革新关键词如下：

- 遵从：新UI更好的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：各种大小的文字易读，图标醒目，去除了多余的修饰，重点突出，很好地突显了设计理念
- 深度：视觉层次和生动的交互动作赋予UI新的活力，不但帮助用户更好的理解新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app或是重新开发一个，尝试一下苹果重新设计系统内置app的方式：

- 首先，去除了UI元素让app的核心功能呈现的更加直接并强调其相关性。
- 其次，直接使用iOS7的系统主题让其成为app的UI，这样能给用户统一的视觉感受。
- 纵观全局，以内容和功能为核心来指导设计，从前的设计模式可以先放到一边。

以内容为核心

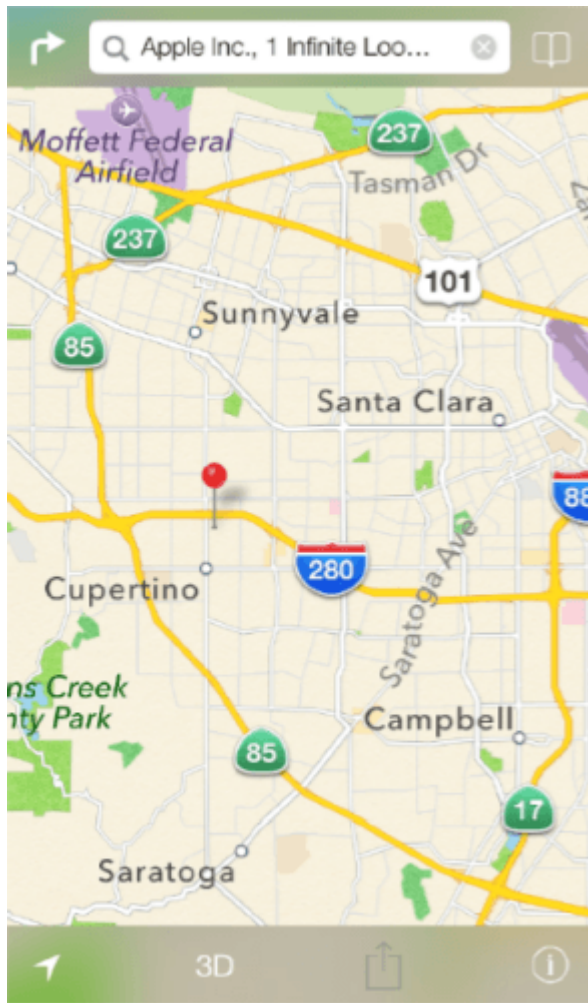
虽然明快美观的UI和流畅的动态效果是iOS7体验的亮点，但内容始终是iOS7的核心。

这里有一些方法，以确保您的设计能够提升您的app功能体验并关注内容本身。

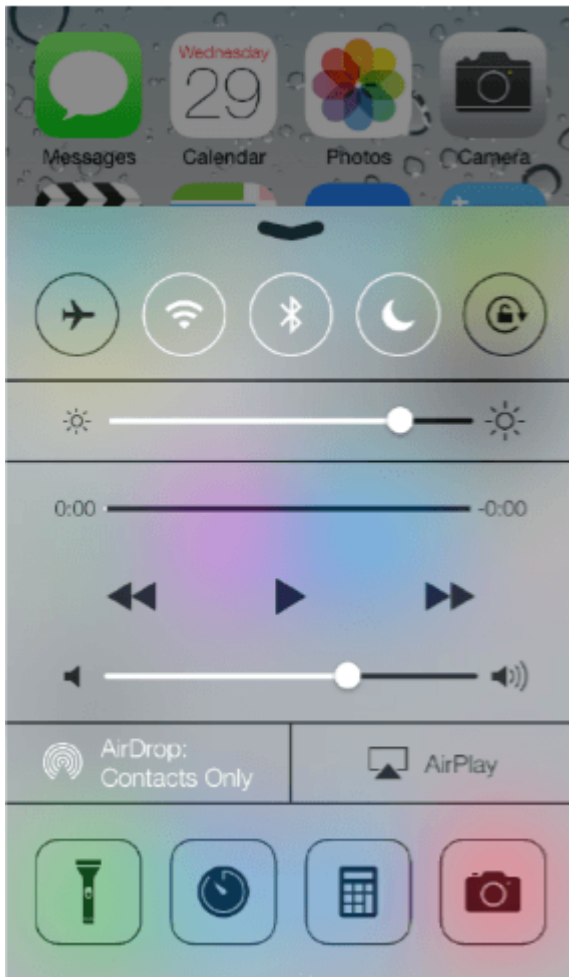


充分利用整个屏幕。无需使用边框、底图等等多余的UI元素，让内容扩展到屏幕边缘。

天气app是最好的例子：漂亮的天气图片充满全屏，告知用户天气情况，同时也很好的呈现了如每个时段气温等等的其他重要信息。



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。

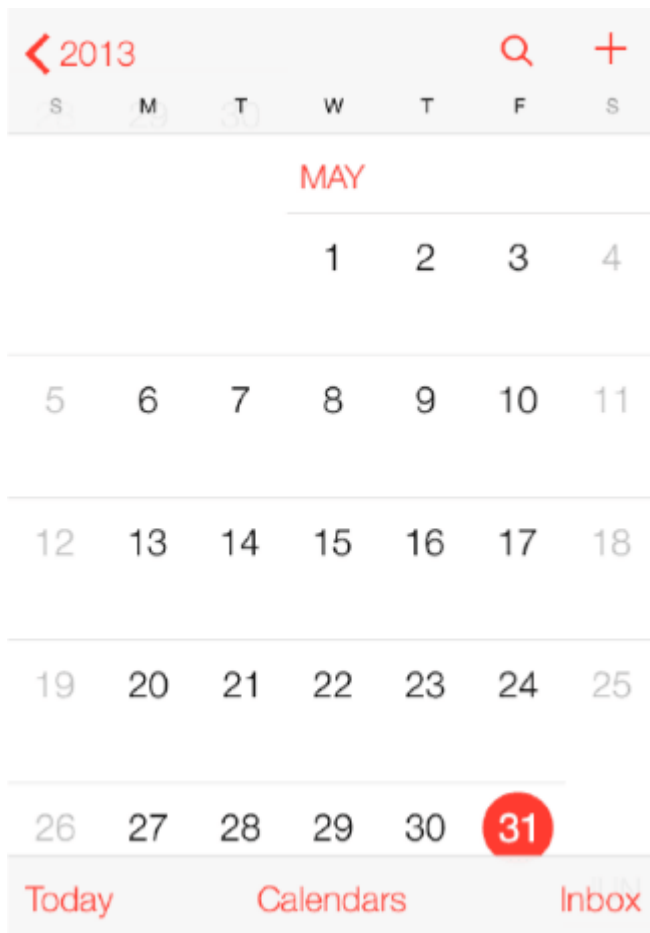


尝试使用半透明底板。半透明的底板可以让用户看到后面的内容，在某些场景下起到了上下文提示的作用，另一个角度上来说，也让用户（比以前）看到了更多内容。

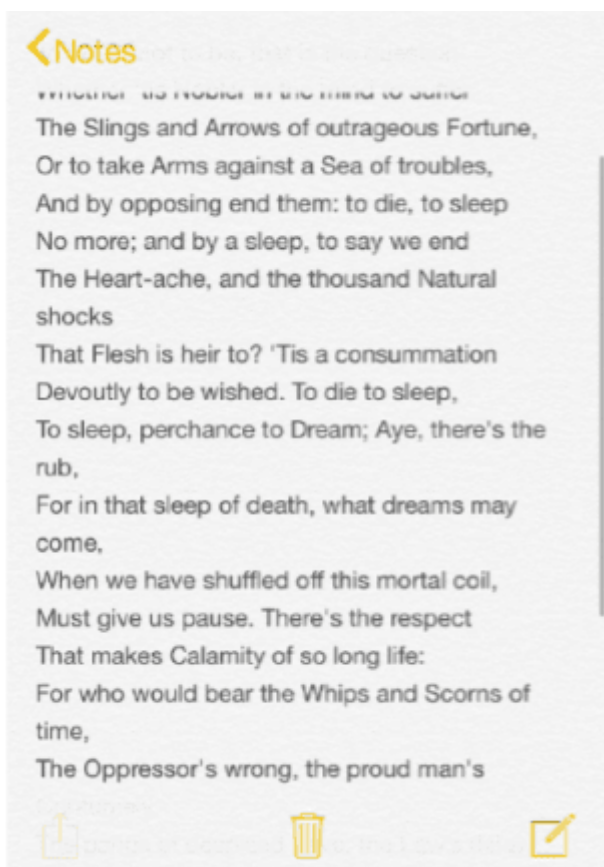
保证清晰度

保证清晰度是另一个方法，以确保你的app中内容始终是核心。

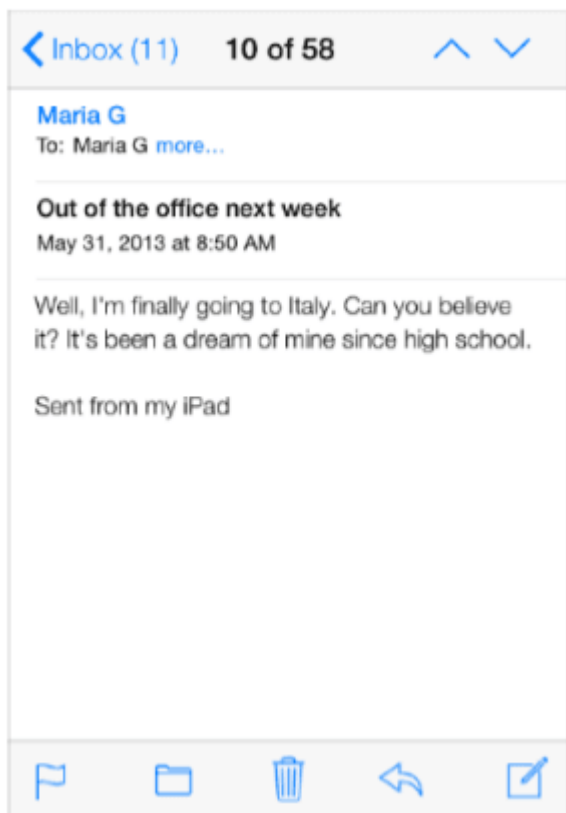
以下是几种方法，让最重要的内容和功能清晰，易于交互。



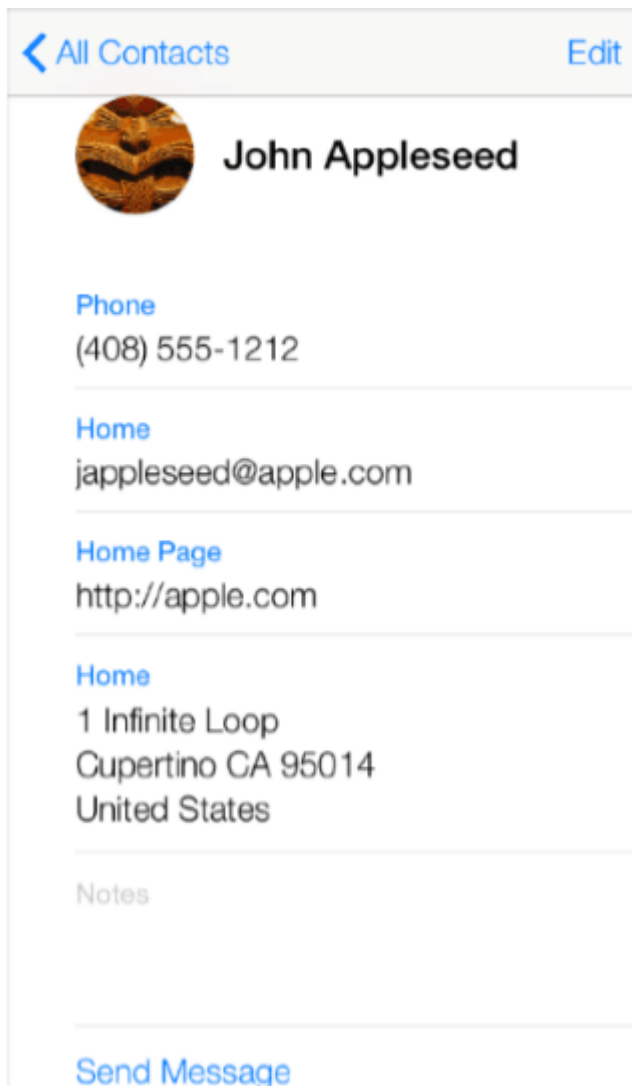
大量留白。空白让重要内容和功能显得更加醒目。此外，空白可以传达一种平静和安宁的视觉感受，它可以使一个app看起来更加聚焦和高效。



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。这也给了一个app一个统一的视觉主题。



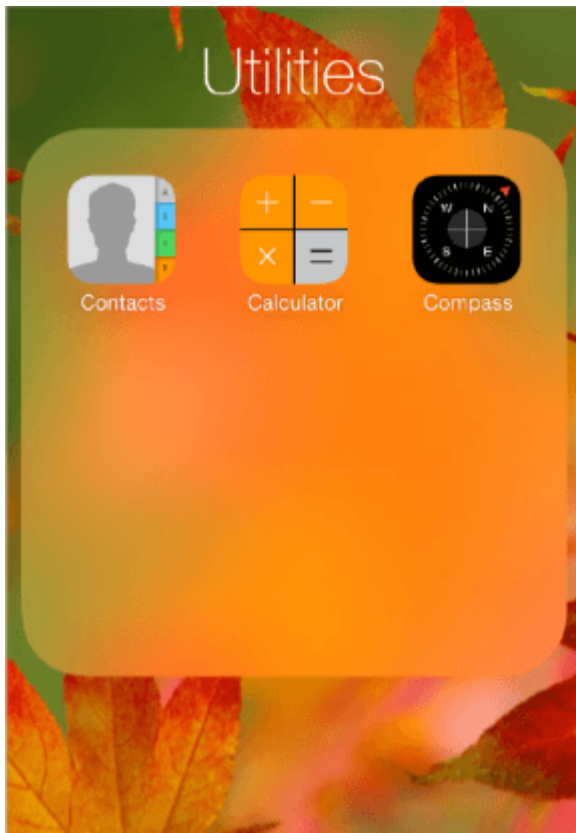
通过使用系统字体确保易读性。iOS7的系统字体自动调整行间距和行的高度，使阅读时文本清晰易读，无论用户选择何种大小的字号都表现良好。



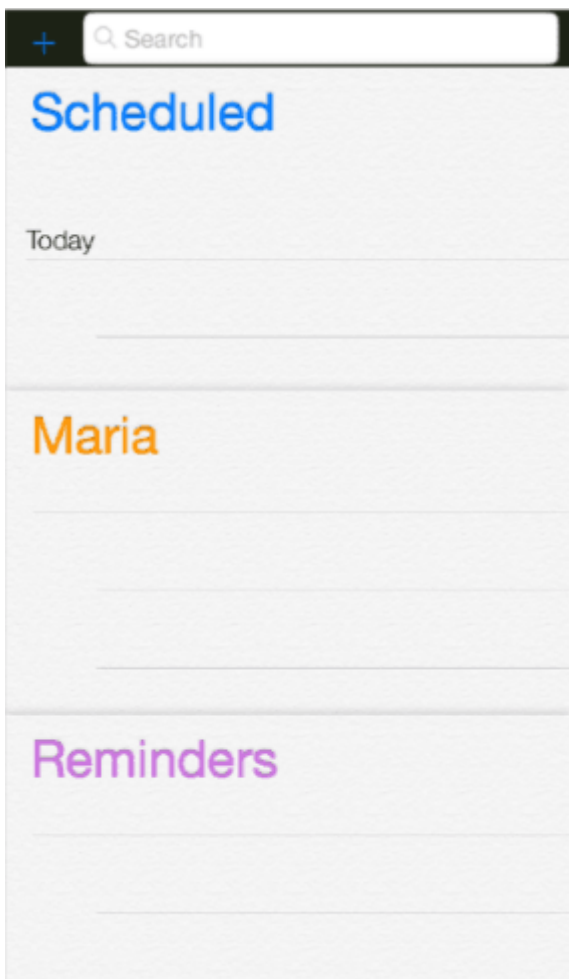
使用无边框的按钮。按钮名称、箭头以及系统颜色向用户展示了交互元素，这些内容替代了原先的带有形状的按钮。联系人界面使用了系统色蓝色箭头文字展示了按钮的导航性。

用深度来体现层次

iOS7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。



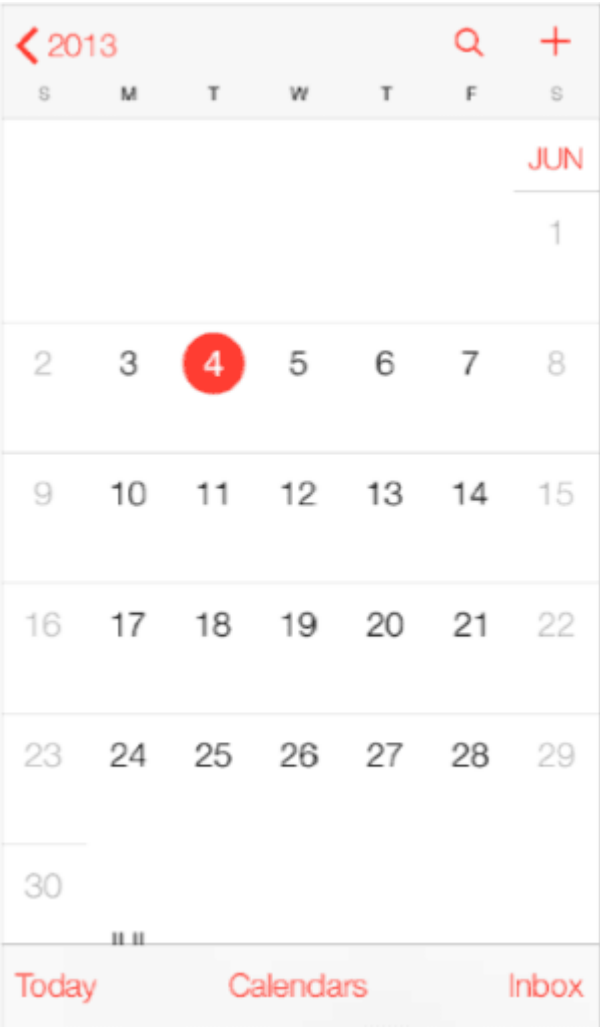
通过使用一个在主屏幕上方的半透明背景浮层来区分文件夹和其余部分的内容。



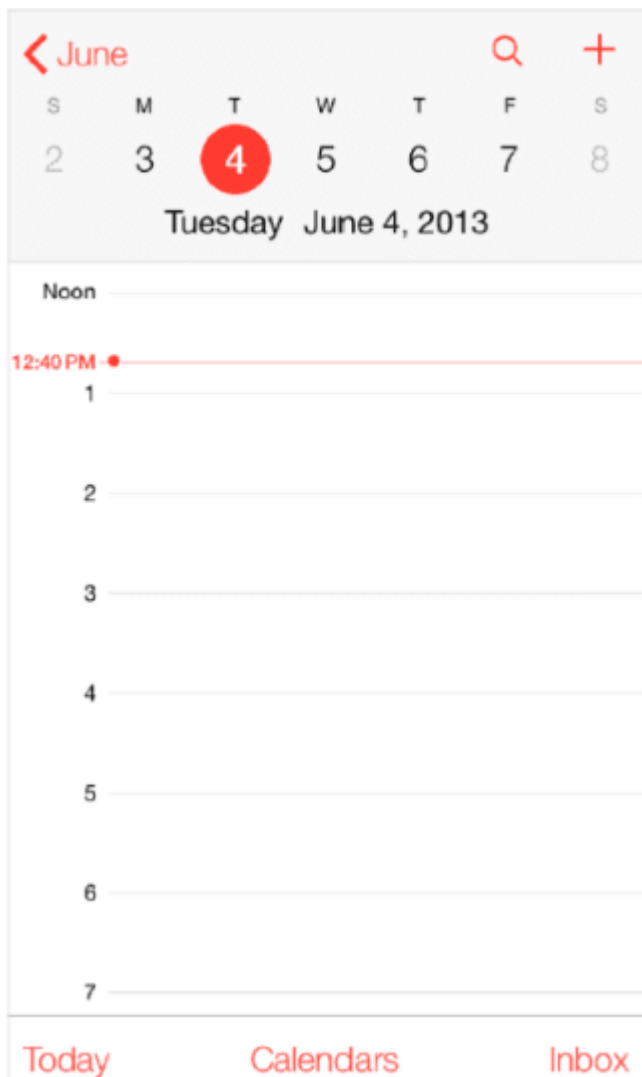
用户在使用备忘录里的某个条目时，其他的条目呈现在屏幕底部的其他分层上，这样用户可以通过滑动展开所有分类条目（译者按：算是一种提示手段）。



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。



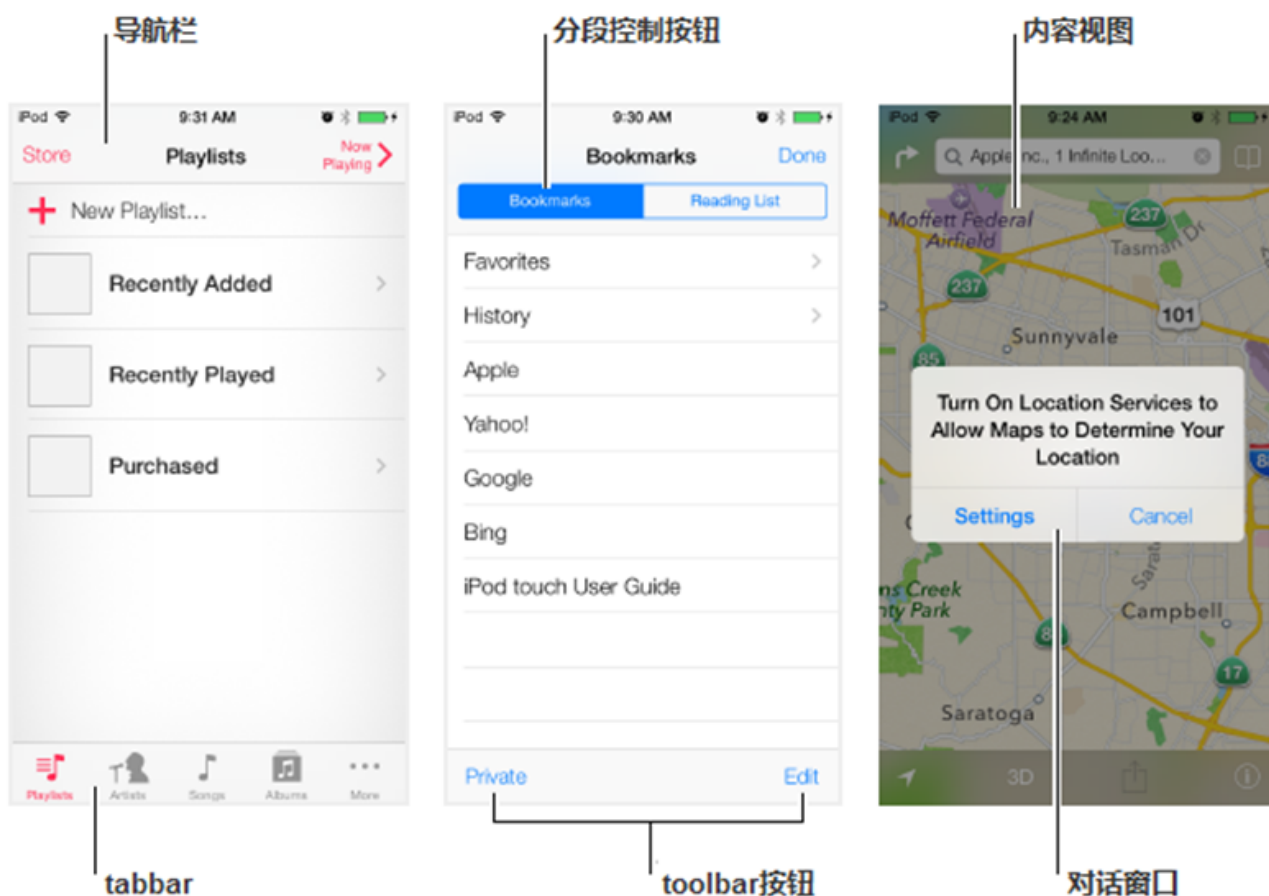
当用户处于月份视图时，点击年份视图按钮，月份会缩小至年份视图中的所处位置。



类似的过度出现在月份和日期视图的切换时，当用户选择某个日期时，月份视图向外扩展，显示出日期视图。

iOS应用解析

几乎所有的iOS app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app过程中做出更好的决定。



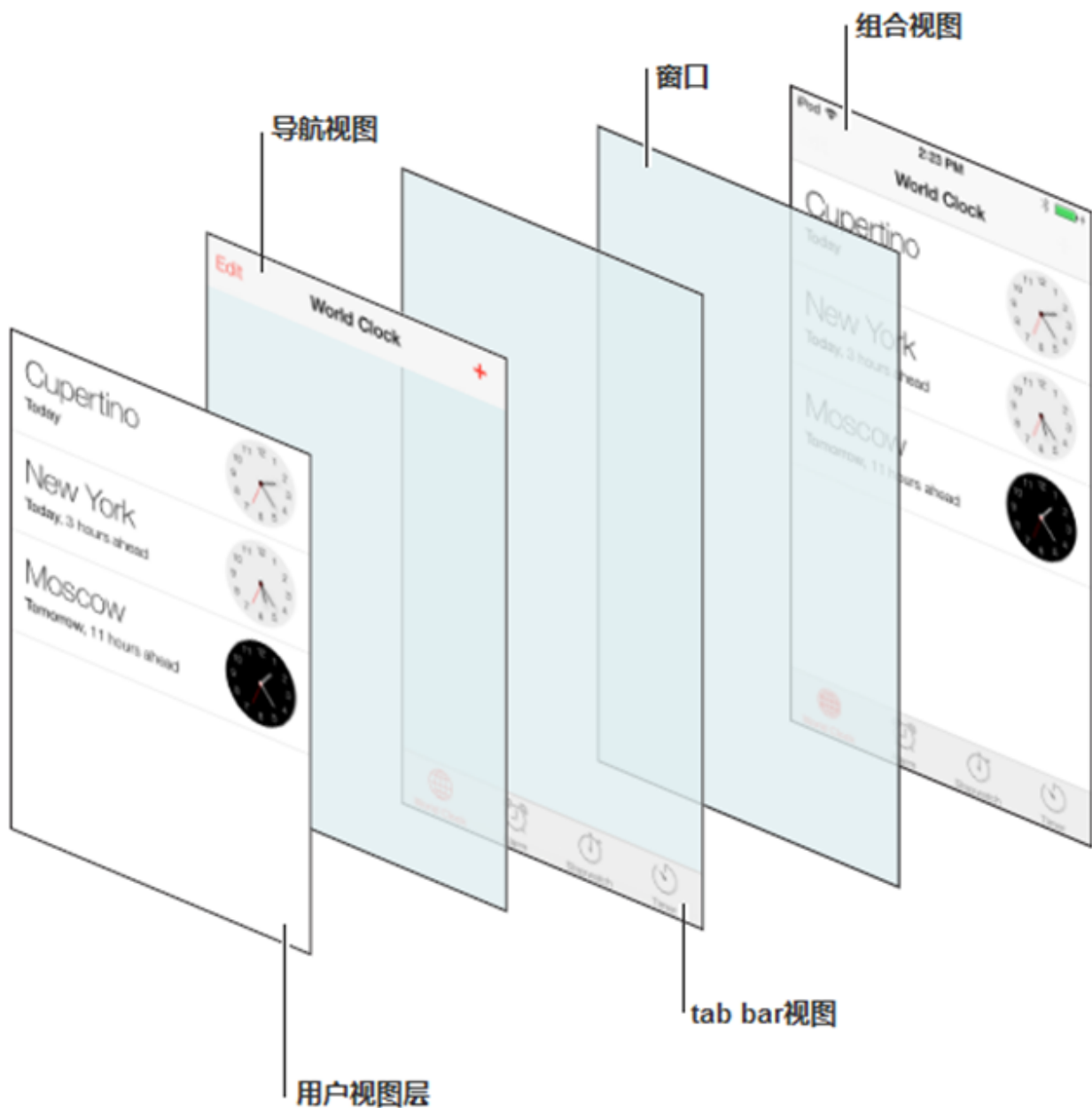
UI组件大致分成以下4种大类：

- **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app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（对话框）**：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，打印支持。

从编程的角度说，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。要在app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不同屏幕内容之间切换。

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 app的UI。



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

起始与停止

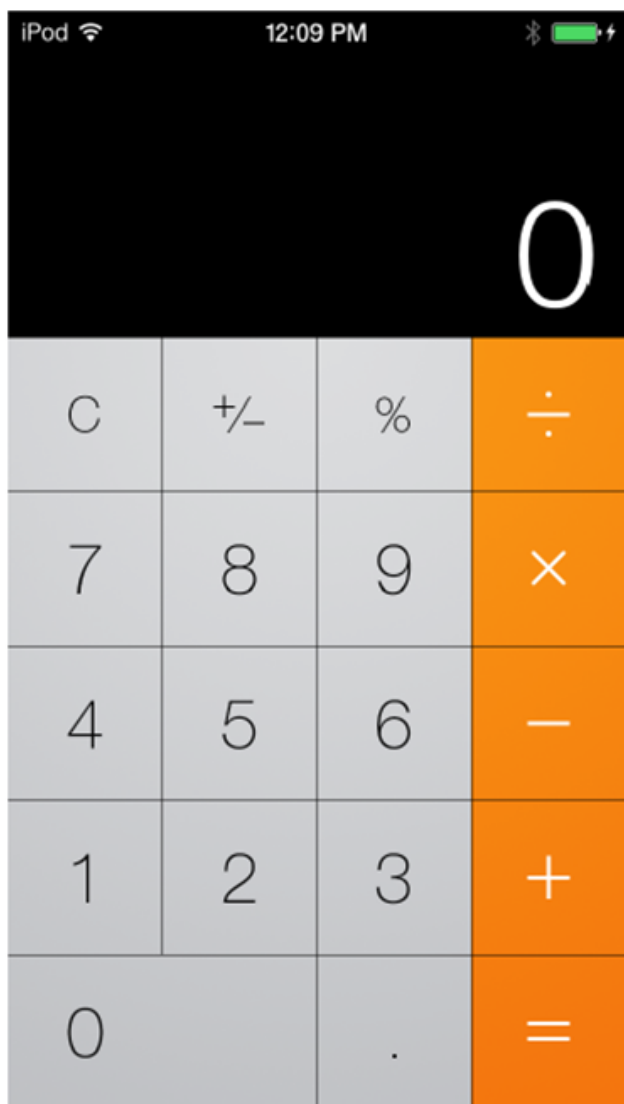
即时启动

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，当你将软件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。

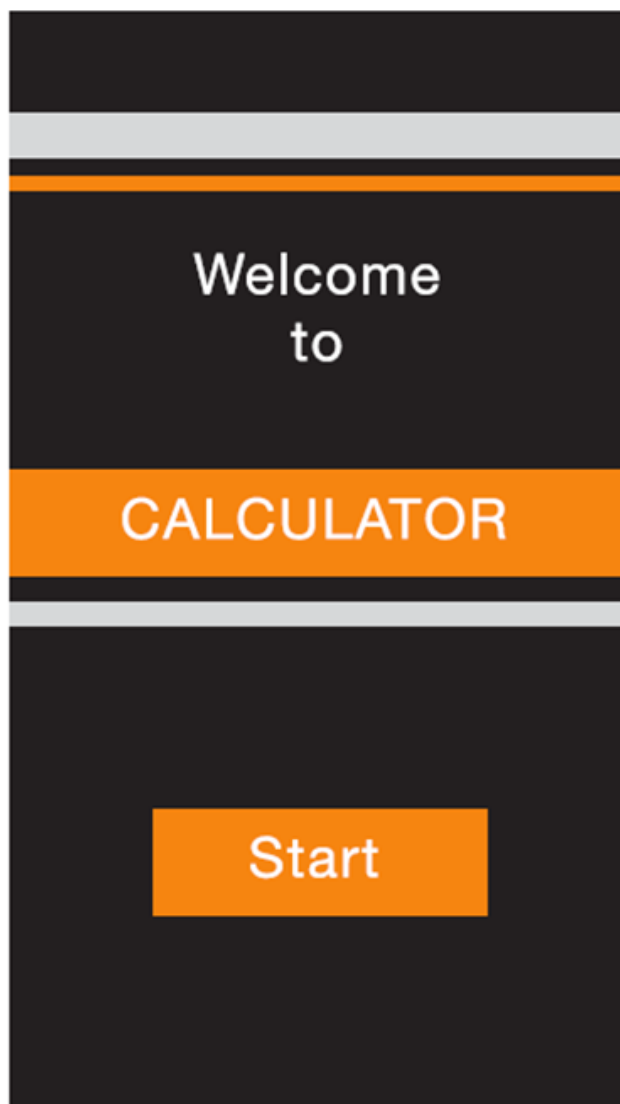
重要：不要在安装过程结束后告诉用户需要重启设备之类的。重启需要时间并且会让人觉得你的应用看上去不可靠而且很难使用。如果你的应用将使用存储空间，或者不重启机器就无法流畅运行，你必须声明这些问题。

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软件是最好不过的。

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**app**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软件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（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**app**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**app**的设置选项进行修改。

尽可能让用户晚一些再登录。让用户在无需登录的情况下就能尽量多的浏览内容并使用部分功能是最理想的状态，。如果用户在熟悉你的**app**之前就被强迫需要登录，那么启动流程就会变得拖沓繁琐。

一般来说，按照屏幕默认的定向方式启动你的**app****。******对于iPhone，竖向是设备默认定向，而iPad则是设备当前所处的方向。如果你的**app**只能在横向模式运行，那么就始终以横向状态启动，让用户在他们自己需要时改变设备方向。

最好让横屏app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app，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app（译者按：大部分人习惯使用右手）。

可以准备一张与app**首页看上去一样的闪屏**，iOS会在启动app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。

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。

时刻准备好停止

iOS app无需关闭或退出选项。当用户切换app或回到主屏幕或者将他们的设备调至睡眠模式的时候，其实就是停止了当前app的使用。

当用户切换app时，iOS的多任务系统将其放置到后台并将新app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。
- 当程序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。

不要强制让app**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：

- 使用吸引注意的屏幕内容描述出现的问题并给出建议的方案。如此可让用户了解到app本身没有问题，并将主动权交给用户，让他们决定是解决问题并继续使用还是切换到其他应用。
- 如果只是某些app*功能无法使用，可以在用户使用这些功能时弹出一个对话框。*只有在用户使用的功能确实无法工作时再继续弹出警告提示。

布局

布局远比UI组件的样式重要。布局能让你向用户展示什么是最重要的，他们是如何选择的，内容是如何相关的。取决于app运行的设备——以及设备的方向——布局可能会有所不同。

让用户尽可能容易地与内容交互并控制好每个控件的间距。需要点击的控件大小至少要有44x44像素。

通过平衡重要内容或者功能，让用户专注于主要任务之上。将重要的组件放置于屏幕上半部分是最常用的方法之一——同时也要遵循从左到右的原则——放置在靠左侧的屏幕上。

利用视觉重心和平衡向用户展示屏幕元素之间的相对重要性。大型部件——以及那些看起来比较重的——更加吸引眼球并且让人感觉比小的部件重要些。

一般来说，避免你的UI不一致。尽可能地让有相似功能的组件有相似的外观。人们经常认为不一致一定有某些原因，并尝试花时间去搞清楚（这样其实是浪费用户的时间）。

确保默认大小的内容（文字、图片）用户能够看清楚。比方说，不要让用户滚动屏幕来阅读（屏幕以外的部分）文本或者无需通过放大操作来看清图像。

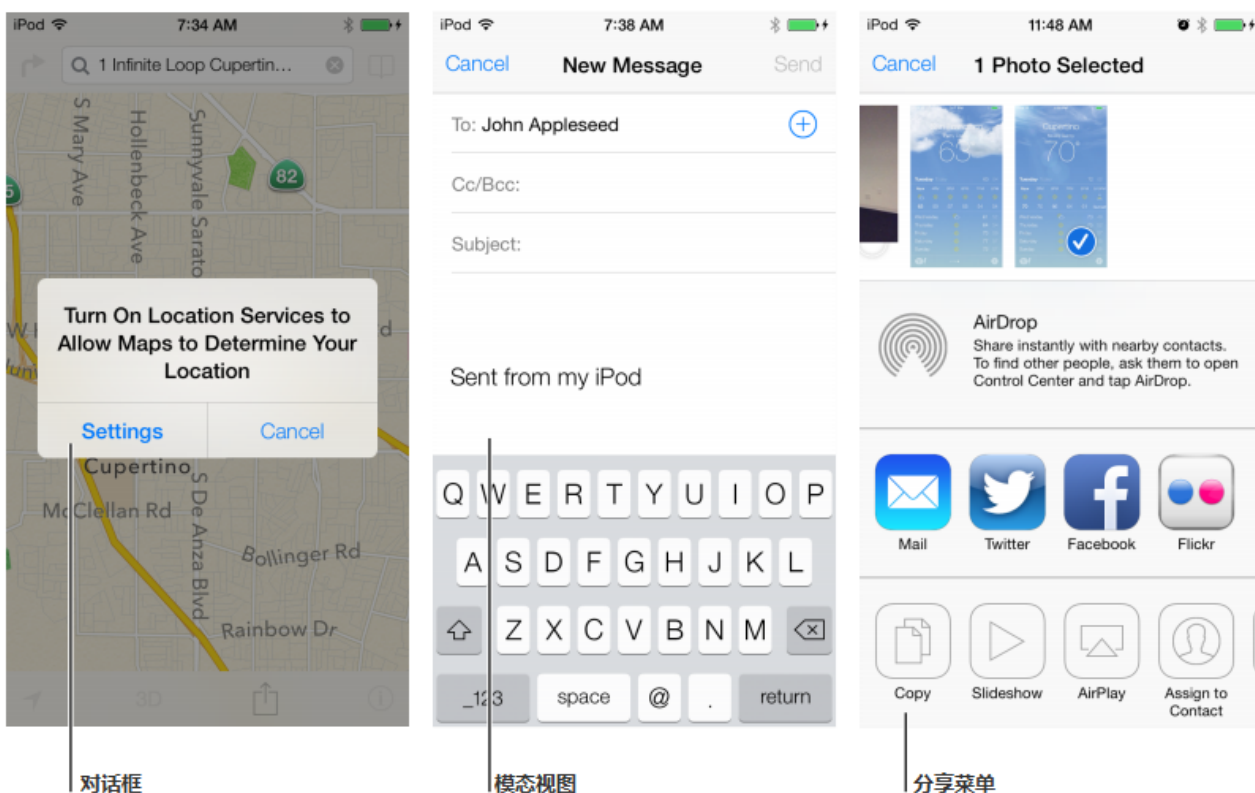
导航

用户很少察觉到一个程序中的导航体验除非它不符合他们的期望。放置导航到一个能够支撑你的app整体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app结构：分层、扁平、内容或经验驱动。在某些情况下，几种导航形式可以出现在同一个app里并且各司其职。无论你的app适合使用哪一种导航结构，最重要的是用户所体验的内容必须被有逻辑的、可预测并易于遵循的呈现在你的app中。用户需要始终很清楚他们在哪儿，并且如何到他们想去的地方。

模态对话

模态视图是一个优缺点并存的模式，承载某些连贯操作或内容，可以给用户在不脱离主任务的情况下完成某些任务或获取某些信息；但是这些操作都是临时的，以防止用户此时与应用程序其余的功能发生交互。



在理想状态下，用户可以与iOS app发生非线性的交互，所以模态视图下最好的做法是最大程度的减小模态操作的数量。大体上说，当以下情况出现的时候，考虑使用模态对话：

- 必须引起用户关注的时候
- 某个任务必须被完成，或者明确被放弃，以避免在模棱两可的状态下遗漏用户信息（操作）。

保持模态任务简单，简短并且高度聚焦。你肯定不希望用户像使用一个mini app那样使用一个模态视图。如果一个模态对话中的子任务太复杂，用户会从暂停的主任务上迷失。创造一个包含一系列视图的模态任务时要特别注意这一点。如果一个模态任务必须在独立视图包含子任务，务必给用户一个独立、清晰的导航路径，并避免迂回。

总是提供一个显眼并安全的方法用以退出模态任务（cancel按钮）。

一个包含一系列视图的任务，必须让用户明白不同步骤中“完成”按钮的作用。

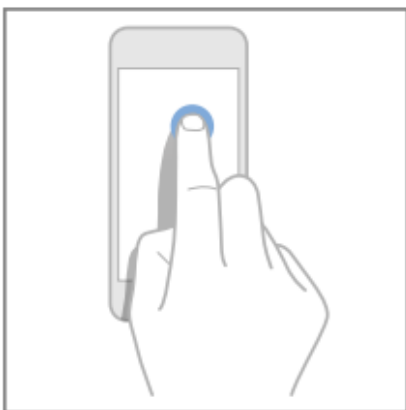
保证对话框提供的信息都是必要的并是可操作的。对话框是打断用户预期的，并且需要点击才会消失，所以让对话框所提示的信息必须是值得中断用户操作的，这对于用户体验来说很重要。

尊重用户关于接收推送通知的选择。在设置界面，用户可以设置app是否接收推送。必须遵循用户的设置，否则会触怒用户导致关闭所有推送通知。

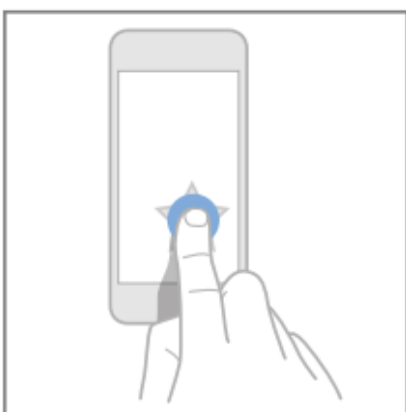
交互性和反馈

标准手势让用户感到舒适

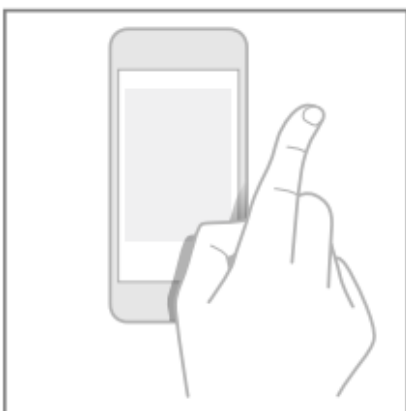
手势操作拉近了设备与用户之间的距离并提高他们的直接操纵感。在app中经常使用的统一手势操作如下：



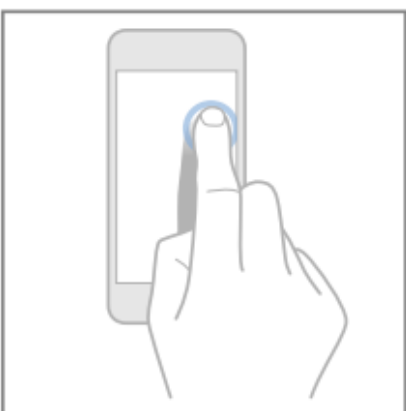
点击 按压或选择一个控制按钮或物件



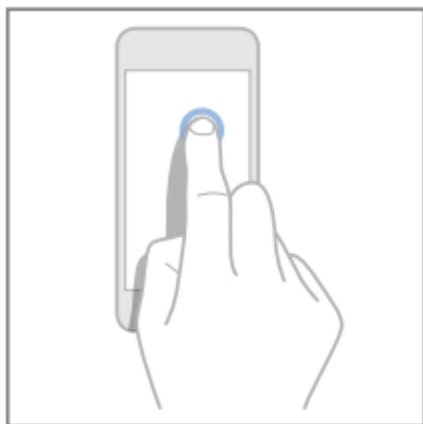
拖动 滚动或平移 从一边移到另一边 拖动某个控件



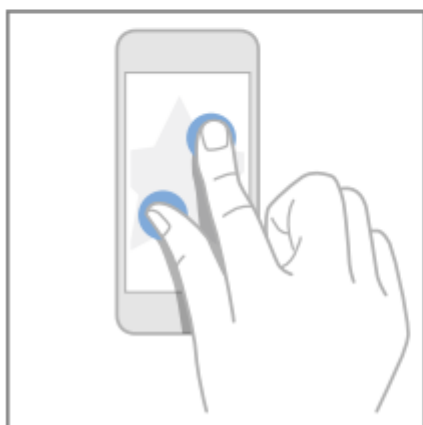
滑动 快速滚动或平移



轻扫 用一个手指反向滑动呼出删除按钮 四指切换任务等



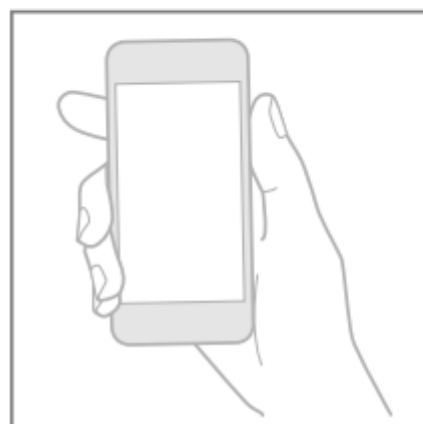
双击 放大缩小，图像中心定位等



捏掐 放大缩小，图像中心定位等



长按 呼出编辑状态或隐藏菜单



摇晃 撤销或重做

避免使用与常规手势操作含义不同的动作。

避免创造与常规手势功能雷同的新手势。

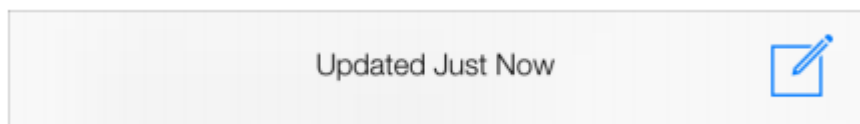
避免让用户用复杂手势完成某种任务。手势操作务必保持简单，直接。

避免创造新的手势，除了在游戏中。

对于 **iPad** 来说，尝试使用多指手势。**iPad** 较大的屏幕给多指操作带来空间。虽然并不是每款 **app** 都需要复杂手势操作，但复杂手势可以丰富用户体验，比如多人同乐的游戏等等。

反馈能帮助用户理解

iOS 用户习惯于得到反馈，帮助他们知道 **app** 正在做什么，下一步可以做什么，并了解他们的操作结果。尽可能地将状态或其他相关的反馈信息集成到 **UI** 上。例如，将邮件的更新状态显示在工具栏上：

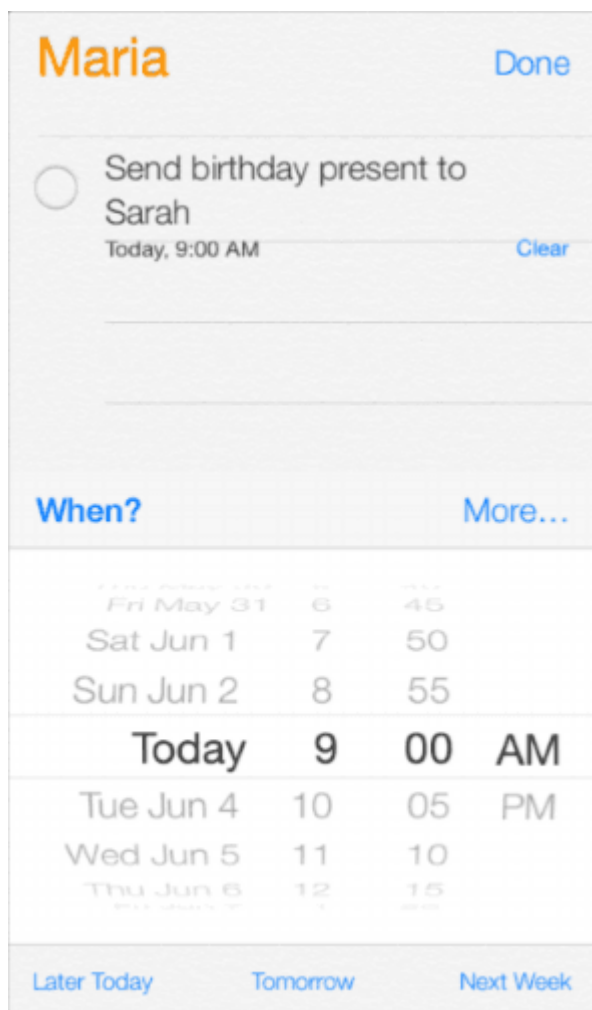


避免不必要的对话框。对话框属于强提示机制，但应仅用于传递重要和有预期的操作信息。如果用户看到太多的对话框，没有包含太重要的信息，那么用户很快就学会忽略所有对话框提示。

输入信息过程需要简易

（在手持设备上）用户利用触摸软键盘输入信息很花时间和精力。如果你的 **app** 因为在输入前出现一堆冗余操作，用户会感到崩溃。

让用户更容易的做出选择。使用选择器或表格视图替代输入操作可以让用户的使用体验更好。



尽可能利用*iOS*获取信息。用户储存了很多信息在他们的设备中。开发者可以自由使用用户已经在系统中输入的信息，比如联系人，日历信息等（当然要获得用户的许可）。

在输入和回报上做出平衡。（在输入后）给出反馈或回报，让用户感觉到他们的行为有价值。

术语和措辞

App中呈现的每一个词都是与用户的一次对话，利用这个机会让用户在使用过程中感到舒适。

使用术语时确保用户能理解。针对用户群确定使用何种短语，例如一些技术术语某些高端用户能理解，但是普通用户就不太熟悉。

可以使用非正式的友好语气，避免太正式又不能太虚假或低三下四。请记住，用户在使用过程中会反复阅读文本，所以有些起初看上去很乖巧的语句多看几次就有可能让人厌烦。

当你的UI文本简短直接，用户可以快速轻松地理解。像新闻编辑一般遣词造句，确定最重要的信息，并强调显示，这样人们就不用看大段文字就知道下一步该怎么做。

给按钮短标签或者易于理解的图标，用户可以一目了然的知道该做什么。

描述时间的时候注意要准确。“今天”、“明天”这样的词听上去是比较友好，但是有的时候会让人混淆，比如你不清楚用户所在的环境（时区不同）。

潜在用户在逛APP Store时，app描述就是最好的沟通机会。除了描述app的品质，你还需要做以下事情：

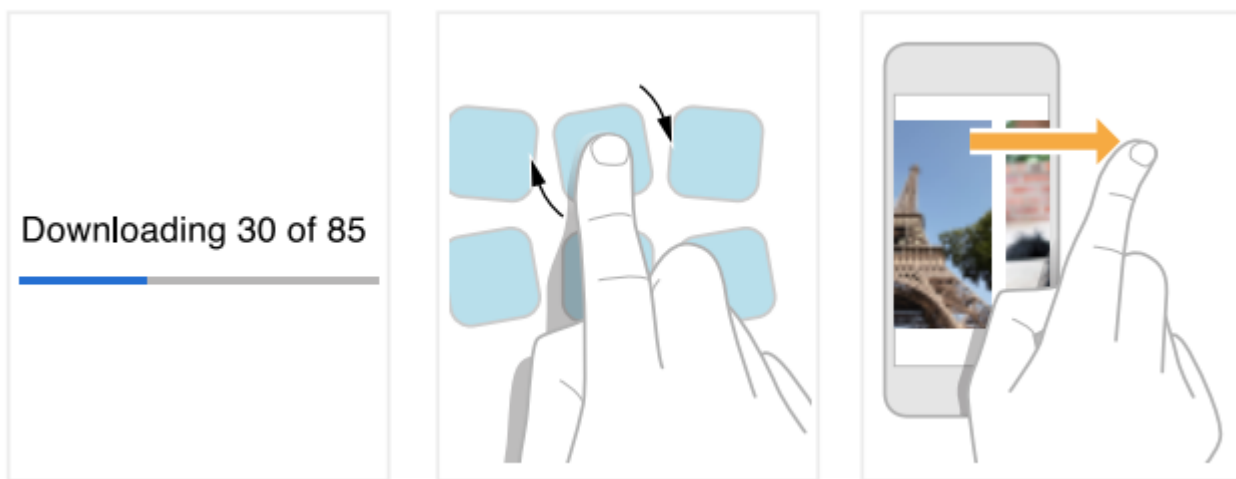
- 修正所有的拼写、语法和标点符号错误。虽然这些小错误不影响使用，但是会给人带来负面印象。

- 尽量少地使用全大写的词汇。虽然有时大写单词可以吸引人们注意，但是全大写的段落不适合阅读，而且有一直朝用户扯着嗓子吼叫的感觉。
- 可以描述bug修复情况。如果您的app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。

动画

美妙优雅的动画贯穿于iOS的用户界面，让app使用体验更加动感和引人注目。微妙和恰当的动画可以：

- 表示状态
- 增强直接操作的意图
- 帮助人们可视化操作的结果。



添加动画的时候需要谨慎，尤其是在app不提供沉浸式体验的时候。app使用过程中，如果在执行主要任务时，过分的、无明显作用的动画往往会分散用户注意力，也影响app的性能表现。

使用与系统内置app一致的动态表现。用户比较熟悉内置应用的动画表现，恰到好处。事实上，用户往往把视图之间的切换、方向变化时的响应、物理感应的滚动等这些效果看做iOS带来的使用体验，除了那些沉浸式app——比如游戏——某些动画可以与系统内置动画相媲美。（译者按：其实就是建议开发者使用系统原生动画效果）

在app中使用的动画效果要统一。如同其他类型的订制，使用统一的自定义动画很重要，这可以让用户在使用不同的app时累计经验。

在大多数情况下，将自定义动画尽量做的真实一点是没问题的。人们往往愿意接受外观表现上的艺术创意，但违反物理定律的动画体验也会让人崩溃（译者按：不要为了开发炫酷动画而违背物理常识）。

排版和颜色

排版必须保持清晰

清晰是排版第一要务。如果用户无法阅读文字内容，再漂亮的文字设计都没有意义。

在你的app中只使用一种字体。不同字体混合使用会让你的app看起来零散拖沓。

Your Game

Play
Directions
Mute
Share
About

颜色可以增强沟通性

尝试定义 **key color**。内置软件使用了 **key color**——比如备忘录中的黄色——用来表明交互和元素状态。

颜色有代表性但人们对颜色的感知并不都相同。不同文化和个体对颜色都有不同理解，这值得花时间去钻研，以免使用的颜色在不同国家和文化中产生禁忌。

在大部分情况下，不要让颜色分散用户的注意力。除非颜色在你的 **app** 里扮演着必不可少的角色，颜色常常在不经意间带来增强效果。

图标和图形

App 图标

每个 **app** 都需要一个漂亮的图标。每个人对图标都有先入为主的印象，关系到 **app** 的品质、目的和可靠性。



有几点关于图标的指引务必记住：

- **app**图标是**app**品牌形象很重要的部分。让图标的设计成为一个机会，给用户讲设计背后的故事，并建立情感链接。
- 最好的应用程序图标应该是独一无二的，整洁的，打动人心的。
- 一个好的**app**图标在每种尺寸和不同背景下看起来都应该合适。细节设计在大尺寸下能丰富图标，但可能在小尺寸时会让图标显得浑浊（译者按：暗示图标图形设计需要简洁）。

其他图标

iOS提供了许多代表了常见任务和操作的小图标，常被用在分页栏、工具栏和导航栏上。建议使用容易被用户理解的内置图标。当然，可以使用自定义图标，如果需要表达自定义的操作或者内容。设计这些小的线型图标与**app**图标有很大区别。

图形

iOS应用大多是图形丰富的。无论显示用户照片还是提供自定义的图片，都有一些应该遵循的准则：

支持**Retina**显示屏。请确保提供两种规格的图片资源。

显示照片或图片时不要超过原始尺寸的**100%**，如果你不想在**app**中看到拉伸变形的图形的话。让用户来选择他们是否要放大或者缩小图片。

不要使用带有苹果符号和版权的图片。这些图形和版权产品的设计经常被修改。

品牌

品牌塑造不止是展示一个品牌的颜色或者**logo**。理想情况下，为你的**app**创造独特的外观和感觉并带给用户难忘的体验，进而打造出特有的品牌形象。

当你需要在应用中展示已有的品牌形象时，请记住下面的要点：

巧妙自然（非强迫式）地融入品牌的颜色或图形。人们使用你的**app**解决问题或者娱乐；他们并不想感觉像被迫看一个广告似的看到你的品牌宣传。最好的体验是，让你的**app**自己说话：比如iOS7通过品牌关键色来表现**app**的互动性和选择状态。

避免侵占主要内容的空间（用于展示品牌信息）。例如，在屏幕上方展示一个用于承载品牌形象的信息栏，这样做反而占用了内容显示的空间。考虑使用其他低干扰的方式来宣传品牌，例如巧妙地定制背景图片（译者按：常见的方式是将品牌**logo**以淡淡的水印形式呈现在背景上）。

重要：对于以上的要点来说，**app**图标是例外，它应该完全聚焦在品牌塑造上。因为用户经常看到**app**图标，更应该花时间来设计它，从而让其在具备品牌辨识度的基础上更加吸引眼球。

与iOS的整合

使用标准UI元素

尽可能用UIKit提供的标准UI元素。当你使用标准而非自定义元素时，你和你的用户都将受益：

- 标准UI元素会自动更新，如果iOS有了重新设计——而自定义元素就不会被升级。
- 使用标准元素对于用户来说没有学习成本。
- 为了充分利用标准UI元素的优点，以下几点比较关键：
- 遵循每个UI**元素的设计规范。**
- 大体来说，请避免创造自定义UI**元素用于表现标准交互行为。**
- 不要用系统自带的按钮和图标表达其他含义。
- 如果你的app**是沉浸式体验，那么创造完全自定义的UI才有足够的意义。**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app。

对切换设备方向的响应

人们通常希望在各种方向都能使用iOS设备，所以在转换方向时时设备应该有合适的响应。

不管设备处于什么方向，请聚焦于主要内容，这是最重要的。人们需要在使用app的过程中与其关心的内容交互。如果主体内容随着设备方向改变而丢失聚焦，那么用户就会感到迷茫并觉得丢失了对app的控制权。

通常，要让app能够在不同屏幕方向下正常运作。人们期望设备在不同方向时都能正常使用app，能满足这一点是最好的。iPad用户常常期望在当前把持设备的方向（正常）使用app，但某些app只能在横屏下使用。如果确实是这样，请注意以下几点：

- 按默认支持的方向启动app**，忽略设备当前朝向。**
- 避免在UI**中告知用户需要调转设备方向。**
- 横屏或竖屏模式时，支持水平调转设备。例如：在横屏app中，无论home键在左或右，app都能正常使用，即支持设备调转180度，app会自行响应。

如果你的app将方向变化当做一种交互的手段，那你可以将方向响应针对app进行特殊处理。比如某些方向响应的游戏通过改变设备方向来移动游戏中的物件，那此时app就无法响应改变设备方向原本应有的变化。这种情况下，可以让用户在进入主线任务前选择改变设备方向；一旦开始主线任务，则按用户此前选择的方向为基准进行响应。

在iPhone上，预测用户什么时候会需要旋转屏幕方向。比如在用户浏览时，旋转方向是为了看到更多内容。如果此时app仅仅放大内容尺寸，你无法达到用户的预期，应该要重新调整内容布局，行间距等等直到更多内容能够很好的被呈现在屏幕上。

在iPad上，尽量支持所有方向来达到用户期望。iPad的大屏幕（能呈现更多内容）减轻了用户在“看到更多”这方面的需求。iPad也很少被用户认为有默认方向，如果可以，尽量满足在任何把持方向下都能正常让用户与你的app进行互动。遵循以下几点规范：

- 考虑改变显示辅助信息或功能的方式。以iPad内置的邮件为例，账户和邮箱属于次要信息（选中的邮件是主要内容）。在横屏时，账户和邮箱被放在左侧面板中，而竖屏时出现在弹出面板上。在某些游戏中，不同方向下的UI也许需要重绘，从而在边界上留下额外空间，这种情况下可以在这些地方展示游戏中的辅助信息或对象。
- 避免无意义的布局变化。尽可能在不同方向下提供一致的体验，从而让用户在旋转屏幕时维持他们的操作习惯。如果你的iPad app在横屏时以网格形式展现图片，在竖屏时就完全没必要改成列表式。
- 避免重新定义信息或文本的方向。对于文字内容来说，尽量保持相同的格式。这样可以避免用户在旋转屏幕时丢失文章定位。如果某些样式必须发生改变，使用动画来帮助用户感受到变化过程。
- 为不同方向都准备一张独立的启动图片。无论用户在何种方向启动应用，体验到的是平滑的启动过程。与iPhone的主屏幕不同，iPad的主屏支持所有方向，以使用户退出并继续打开同一个app。

淡化文件和文档处理

iOS可以帮助用户创建并管理文件，但并不代表用户必须考虑iOS设备的文件系统如何运作。

在iOS中没有类似于OS X系统中的Finder（管理软件），用户无法像在电脑上那样操作。特别是不应该让用户考虑文件所在的位置之类的东西，比如：

- 打开或保存文件的对话框
- 文件许可状态的信息

尽可能允许用户无需在电脑上打开**iTunes****就能管理文档。**考虑使用iCloud帮助用户访问不同设备上的内容。

如果你的**app**能帮助用户创建并编辑文档，可以提供文件选择器让用户打开已有文件或者创建新文件。

给用户信心，他们的工作成果会被随时保存除非主动取消或者删除。iOS应用应该承担起帮助用户保存输入内容的责任，无论是打开另一个文档或切换应用的时候。

如果你的**app**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需要保存修改。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处于编辑模式。“保存”可以保留改变，“取消”可以退出编辑模式。

必要时提供可设置选项

某些应用需要安装或设置选项，但是大部分应用不需要这么做。一个成功的**app**可以让大部分用户上手迅速并通过主界面给用户调整体验的方式。

避免让用户去（系统）设置中（寻找解决方法）。请记住，用户必须关闭你的**app**才能进入系统设置，相信你也不希望用户这么做。

当你的**app**（的默认状态）满足大部分用户的期望，（用户）对设置项的需求就减少了。

如果有必要，让用户在你的**app**内进行设置。提供设置选项可让你的**app**的变化直接体现，并让用户看到，也无需离开你的**app**再去设置。

尽可能在主界面提供设置选项。用户在执行主线任务时如果想频繁改变设置，放置在主界面的设置项就很有意义。如果用户只是偶尔用到设置项，就将其放在独立的视图中。

UI元素

重要：这是一份针对API或其它相关技术开发而准备的预备文档。尽管文档在专业精确程度上已经过多次审查，它仍不是最终版本。文档仅供已注册苹果开发者计划的开发者使用。苹果提供这份文档的目的，是帮助开发者根据文档来规划自身应用的开发技术与界面设计。这些信息将可能发生变化，您的应用也应当根据最新的操作系统与最终文档进行相应的调整。该文档可能会由于API与相关技术的发展而更新版本。

栏(Bars)

状态栏(Status Bar)

状态栏展示了关于设备及其周围环境的重要信息。



你可以将状态栏风格设计为全应用统一，或者为应用里不同的视图定义不同的状态栏风格。你可以通过阅读[UIApplication Class Reference](#)与[UIViewController Class Reference](#)来分别了解更多关于UIStatusBarStyle常数和preferredStatusBarStyle属性的内容。

外观和行为

状态栏是透明的。不管设备处于横屏还是竖屏，状态栏始终固定在整个屏幕的上边缘，承载用户所需要的如网络连接，时间，电量等信息。

指南

尽管你不会像使用其它UI元素一样编辑状态栏，理解它在应用中的功能仍然很重要。

隐藏状态栏时请慎重。由于状态栏是透明的，通常情况下不需要隐藏它。始终隐藏状态栏意味着用户必须退出你的应用才能知道现在的时间，或者了解是否当前环境下是否有Wi-Fi连接。

在用户全屏观看媒体时，考虑隐藏状态栏以及所有页面UI。当你这么做的时候，请确保用户在轻击屏幕时即可重新唤起状态栏以及相关的UI。而除非你有充分的理由，否则最好不要重新定义一个手势来让用户唤起状态栏，因为用户不会发现，就算发现了也难以记住。

不要创建自定义状态栏。用户依赖系统默认状态栏的一致性。就算你在应用中隐藏了它，也优于定制一个新的UI来代替它。

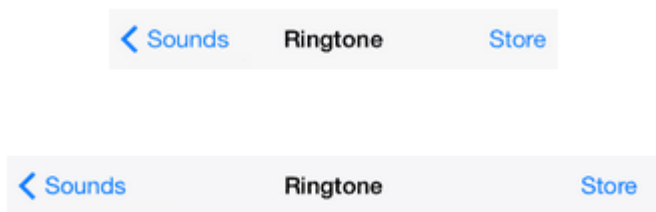
为你的应用选择配色协调的状态栏颜色。内容是深色的状态栏的在浅色应用中效果出色，而相应的浅色状态栏则更适用于颜色较深的应用。

千万千万，避免在状态栏后面叠加会分散注意力的内容。尤其是，你不能让用户觉得轻击状态栏之后可以获取内容或激活你的应用中的控件。

在适当的时候展示网络活动指示器(**network activity indicator**)。这可以提醒用户当前远程网络连接(**lengthy network access**)正在工作。更多详情请参考本章第三节控件(Control)部分的Network Activity Indicator.

导航栏(Navigation Bar)

导航栏能够实现在应用不同信息层级结构间的导航，有时候也可用于管理当前屏幕内容。



导航栏包含在导航控制器（**navigation controller**）中，该控制器是一个用于管理自定义视图中信息层级展示形式的编程对象。想要了解如何在代码中定义你的导航栏内容，请参考[Navigation Controllers](#)和[Navigation Bars](#).

外观和行为

导航栏通常位于屏幕的上方，状态栏正下方。导航栏居中展示当前屏幕或当前视图的标题。当用户在信息层级中穿梭时，也可以通过点击导航栏中的返回按钮，或轻扫屏幕的边缘来回到上一层。另外，用户可以使用导航栏上相应的控件来管理当前的屏幕内容。

导航栏是半透明的，上面所有的控件都是无边界的(borderless)。

在iPhone上，导航栏始终与屏幕等宽以通栏显示，当用户改变屏幕方向时，导航栏的高度也将自动发生改变。

指南

你可以用导航栏在不同视图间提供导航，或在上面放置管理当前视图内容的相关控件。

使用当前视图的标题作为导航栏标题。若觉得标题冗余，你也可以将标题留空。

当用户到达一个新的层级，你需要做以下两件事：

- 将导航栏标题改为当前层级的标题
- 在当前标题左侧放置返回按钮，按钮的标题应为前一层级的标题

确保导航栏上的文字容易阅读。系统默认字体的可读性最好，但合适的话，你也可以使用其它的字体。

考虑在应用最高层级的导航栏中放置一个分段控件，它能够帮助你更好地扁平信息层级，也会让用户更容易找到所需内容。如果在导航栏中使用了分段控件，请确保返回按钮标题命名的准确。更多详情请参考本章第三节中的 **Segmented Control**。

即使空间充足，也应当避免让过多的控件填满你的导航栏。导航栏上应该不多于以下三个元素：当前视图的标题、返回按钮和一个针对当前的操作控件。相反的，当你在导航栏中使用了分段控件，就不要再放标题以及其它多余控件了。

根据控件的标准含义来选择系统提供的按钮。详情请参考下文中工具栏与导航栏标准按钮(**Toolbar and Navigation Bar Buttons**)。如果想自定义导航栏控件，请参考文档第五章中 **Bar Buttons Icons** 给出的建议。

根据你应用的配色来定义你的导航栏颜色。举个例子，你可以为导航栏自定义背景图片，也可以指定它的色调与透明度。有时候使用可缩放的背景图(**resizable background image**)是个好主意。想要了解更多可缩放背景图的使用方法，请参考文档第五章 **Creating Resizable Images**。请提供适用于iOS 7应用的图片高度，更多详情请参考[iOS 7 UI Transation Guide](#)中的 **Navigation Bar** 部分。

确保你的导航栏与你的应用的外观和风格是协调的。举个例子，不要在同一个应用中使用不透明导航栏和半透明工具栏。在屏幕处于同一方向时，最好不要改变不同屏上导航栏的背景图片、颜色和透明度。

确保你自定义的返回按钮长得像返回按钮。用户知道系统默认的返回按钮能帮助他们在信息层级中追踪自己的路径，如果你想重新设计它，请仍然使用自定义模版图片(**custom mask image**)，它可以在iOS7中让这些按钮标题在系统各转场中出现或者消失。

在iPhone上，要考虑到由于屏幕方向的变化将会导致导航栏高度自动变化。确保你自定义的图标可以适应导航栏高度变小的情况。不要将导航栏高度写死，可以利用 **UIBarMetrics** 常数来确保图标的适应性。

工具栏(Toolbar)

工具栏上放置着用于操作当前屏幕中各对象的控件。



工具栏包含在导航控制器(navigation controller)中，该控制器用于管理定制视图中信息层级的展示形式。想要了解如何在代码中定义工具栏，请参考[View Controller Catalog for iOS](#)中的[Displaying a Navigation Toolbar](#)和[Toolbar](#)部分。

外观和行为

在iPhone上，工具栏始终位于屏幕底部，在iPad上则有可能出现在顶部。

工具栏是半透明的，栏中各项以等距方式排列。因为不同界面所对应的操作不同，工具栏中的控件可能随着界面的切换而进行相应调整。

在iPhone上，当用户从竖屏转换为横屏时，状态栏的高度将自动发生改变。在iPad上，工具栏的高度则不会因设备方向而发生变化。

指南

你可以在工具栏上放置可让用户对当前视图内容进行操作的工具。

在工具栏里放置用户在当前情景下最常用的指令。你也可以在工具栏里放置分段控件以方便用户快速到达不同视图或模式；更多使用指南，请参考本章第三节——控件(Controls)中的分段控件(Segmented Control)。

如果需要在工具栏上展示3个以上的项目，可以使用图标。由于文本按钮通常会比图标更占空间，所以用图标可以避免文字标题们挤在一起。

保证工具栏标题按钮之间有足够的间距。控件过于拥挤会让用户觉得它们难以区分。如果工具栏中按钮的标题看起来太接近，可以用UIBarButtonItemSystemItemFixedSpace常数来增加他们之间的间距。（更多关于这个常数的内容可以参考[UIBarButtonItem Class Reference](#)）

在iPhone上，要考虑到由于屏幕方向的变化将会导致工具栏高度的自动变化。确保你自定义的控件可以适应横屏模式下工具栏高度变小的情况。不要将工具栏高度写死，可以参考UIBarMetrics 常数来确保工具栏中各项的适应性。

工具栏与导航栏标准按钮(Toolbar and Navigation Bar Buttons)

iOS提供了一系列用于工具栏与导航栏的标准按钮。想要了解如何设计自定义图标，请参考文档第五章Bar Button Icons部分。工具栏和导航栏图标的颜色可以通过tintColor属性来设定。

想要了解每一个按钮所对应的标志名称及其含义，请参阅[UIBarButtonItem Class Reference](#)中的[UIBarButtonItem](#)部分。

重要：跟所有标准按钮和图标相同，应当根据文档中说明的图标含义，而不是只凭图标外观来使用这些工具栏图标和导航栏图标。这样能够保证在关联特定意义的按钮改变了外观的情况下，你的应用中的UI仍然是可用而有意义的。

表 34-1 工具栏与导航栏标准按钮(Standard buttons available for toolbars and navigation bars)

按钮	名称	含义
	分享(Share)	呼出分享操作列表，列表中应包含系统操作和针对应用自定义的服务或动作
	相机(Camera)	呼出相机拍摄的操作列表，列表中应当包含在相机模式下的照片选择器
	编写(Compose)	打开一个新的消息编辑视图
	书签(Bookmarks)	展示书签(app-specific bookmarks)
	搜索(Search)	显示搜索字段
	添加(Add)	新建一个项
	回收站(Trash)	删除当前项
	整理(Organize)	将某个项移动到应用内的其它位置，比如另一个文件夹内
	回复(Reply)	将某个项发送到另外一个位置
	刷新(Share)	刷新当前内容（请在必要时才使用它，尽量自动刷新）
	播放(Play)	播放当前媒体内容
	快进(FastForward)	快进当前内容
	暂停(Pause)	暂停当前内容
	回退(Rewind)	回退当前内容

除了以上展示的标准按钮之外，你还可以使用系统提供的编辑、取消、保存、完成、撤销、重做等等按钮来支持编辑或其它操作。这些按钮的标题即是按钮的操作内容。想要了解每一个按钮的名称及其含义，请参阅[UIBarButtonItem Class Reference](#)中的[UIBarButtonItemSystemItem](#).

另外，你还可以在工具栏中放置系统提供的信息按钮(info button).



标签栏（Tab Bar）

标签栏用于让用户在不同的子任务、视图和模式中进行切换。



标签栏包含在标签栏控制器中，该控制器用于管理自定义视图的展示形式。想要了解如何在代码中定义标签栏，请参考[Tab Bar Controllers](#)和[Tab Bars](#)。

外观和行为

标签栏位于屏幕底部，并应该保证在应用内任何位置都可用。标签栏是半透明的，展示图标和文字内容，每一项均保持等宽。当用户选中某个标签时，该标签呈现适当的高亮状态。

标签栏是半透明的，始终与屏幕等宽以通栏显示。因为不同界面所对应的操作不同，工具栏中的控件可能随着界面的切换而进行相应调整。

在iPhone上，一个标签栏一次最多可承载5个标签；多于5个标签的时候，可以展示前4个标签，把剩余的标签以列表的形式收在“更多”之中。iPad的标签栏则可以承载5个以上的标签。

你可以在标签上加上红底白字，显示数字或者省略号的小气泡（**badge**），用以展示与应用相关的信息。

标签栏的高度不随着屏幕方向的改变而改变。

指南

你可以使用标签栏来切换对同一组数据的不同视图模式，或者整体功能下不同的子任务。当你使用标签栏时，请遵守以下指南：

一般而言，使用标签栏来组织整个应用层面的信息结构。标签栏非常适合用于应用的主界面中，因为它可以很好地扁平信息层级，并且同时提供多个进入不同信息种类的入口。

不要使用标签来执行对于当前屏幕内容的操作。如果你需要给用户提供的操作控件，请使用工具栏。

即使标签当前不可用，也不要把它从标签栏中删除。如果某个标签所代表的部分功能在当前场景下不可用，可以将它标识为不可用状态，但不要删除它。让某些标签时而出现时而隐藏，会让用户觉得应用UI不稳定而且难以预测。最好的解决方式是确保每个标签都可用，并解释当前标签不可用的原因。举个例子，当用户没有在设备中保存任何歌曲，在系统音乐应用的歌曲标签页里就可以教育用户如何去下载一首歌。

考虑在**tab**上加入红色的小气泡(**Badge**)以传达信息。你可以通过添加小气泡来告知用户该标签中包含新的内容。

根据控件的标准含义来选择系统提供的图标。详情请查看下文中的标签栏标准图标(**Tab Bar Icons**)。如果想自定义标签栏图标，请参考文档第五章中**Bar Buttons Icons**里给出的建议。

可能的话，自定义你的标签栏外观。举个例子，只要你的图标是系统默认的标准图标或遵循了系统模版的图标，你都可以为你的标签栏以及上面的图标设计特有的视觉风格。有时候使用可缩放的背景图是个好主意，想要了解更多可缩放背景图的使用方法，请参考文档第五章**Creating Resizable Images**。

在iPad上，你可能会在对分视图(**split view pane**)或者浮出层(**popover**)内使用标签栏以切换或筛选视图中的内容。然而通常情况下，在对分视图和浮出层底部使用分段控件效果会更好，因为视觉上看起来更为协调。更多详情请参考文档本章第三节——控件(**Controls**)中的**Segmented Control**。

在iPad上，避免让过多的标签填满你的标签栏。放置太多控件将导致用户难以点击，同时每添加一个标签，意味着你的应用程序又复杂了一分。一般来说，在主界面(main view)和对分视图的右窗格上来说，标签数量应该控制在7个左右；而对于浮出层和对分视图的左窗格来说，标签数以5个左右为宜。

在iPad上，无论横屏还是竖屏情况下都应展示相同数量的标签，以提高应用的视觉稳定性。竖屏视图中我们推荐标签个数在7个左右，在横屏中，你应该将相同数量的标签居中展示。这个建议同样适用于在对分视图或者浮出层中的标签栏。举个例子，如果你在竖屏模式下的一个浮层中使用了标签栏，那么横屏时它应该也能很好地展现在对分视图的左窗格中。

标签栏标准图标(Tab Bar Icons)

iOS提供了一系列标签栏标准图标。想要了解如何设计自定义图标，请参考文档第五章Bar Button Icons部分。标签栏图标的颜色可以通过tintColor属性来设定。

想要了解每一个图标的名称及其含义，请参阅[UIBarItem Class Reference](#)中的UIBarButtonItemSystemItem部分。

重要：跟所有标准按钮和图标相同，应当根据文档中说明的图标含义，而不是只凭图标外观来使用这些图标。这样能够保证在关联特定意义的按钮改变了外观的情况下，你的应用中的UI仍然是可用而有意义的。

表 34-2 标签栏标准按钮(Standard icons for use in the tabs of a tab bar)

按钮	名称	含义
	书签(Bookmarks)	展示应用指定书签(app-specific bookmarks)
	通讯录(Contacts)	展示通讯录
	下载(Downloads)	展示下载内容
	个人收藏(Favorites)	展示用户的个人收藏
	精选(Featured)	展示应用推荐的精选内容
	历史记录(History)	显示用户的历史记录
	更多(More)	显示更多标签项
	最新动态(MostRecent)	显示最新动态
	浏览最多(MostViewed)	显示所有用户近期浏览最多的内容
	最近使用(Recents)	显示用户在程序指定的时间内访问过的项
	搜索(Search)	进入搜索模式
	评分最高(TopRated)	显示用户评分最高的项

搜索栏(Search Bar)

搜索栏获取用户键入的文本，用以作为搜索的关键字(下图中显示的文本为占位符，非用户输入文本)。



想要了解更多的详情，请参考[Search Bars](#)。

外观和行为

搜索栏的外观类似文本框。默认状态下左侧有搜索图标，当用户点击搜索栏时会自动出现键盘；用户输入完后，系统将按照应用程序定义的方式来处理输入的文本。

搜索栏可能包含以下这些可选元素：

- 占位符文本(**Placeholder text**)。占位符文本通常会写明控件的功能——如“搜索”，或者提示用户输入的文本将在哪里搜索，如“Google”。
- 书签按钮(**The Bookmarks button**)。书签按钮可以让用户方便地找到他们需要的内容。例如在地图中搜索时，用户可以通过书签按钮快速选中书签地址、最近搜索记录、或联系人。书签按钮只有当搜索栏中没有占位符或用户输入内容时才会出现，当搜索栏中已有文本时，书签按钮会被清除按钮(**Clear button**)所代替。
- 清除按钮(**The Clear button**)。大多数搜索栏都会提供清除按钮，方便用户一键清空输入内容。一旦用户在文本框中输入内容，清空按钮就会出现；而当搜索框中没有任何文本内容时，清空按钮将被隐藏。
- 结果列表图标(**The results list icon**)。结果图标说明此次搜索有搜出结果。当用户点击它时会出现用户最近一次搜索的搜索结果。
- 描述性标题，我们称为提示 (**Prompt**)。描述性标题通常放在搜索栏上方。举例来说，它可以是一个为搜索栏提供指引信息的短语。

指南

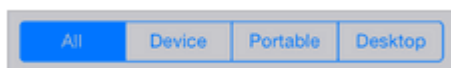
在你的应用中使用搜索栏而非文本框来让用户进行搜索。因为文本框的外观不符合用户对搜索的预期。

你可以为搜索栏指定颜色或设计背景图片。在iOS 7里，你也可以把搜索栏放在导航栏中。请参考[UISearchBar](#)。

如果你决定在搜索栏里使用背景图片，最好采用可调整尺寸的背景图，想要了解更多可调整尺寸背景图的使用方法，请参考文档第五章中[Creating Resizable Images](#)的内容。

范围栏(Scope Bar)

范围栏只有在与搜索栏一起时才会出现（通常出现在搜索栏下方），它让用户可以定义搜索的范围。



想要了解如何在代码中定义搜索栏与范围栏，请参考[Search Bar](#)。

外观和行为

当搜索栏出现时，范围栏会出现在它的附近。范围栏的外观与你所指定的搜索栏的外观兼容。

指南

当用户想在明确的分类范围内进行搜索时，使用范围栏是非常有用的。然而，更好的选择是优化您的搜索结果，让用户不需要使用范围栏对搜索结果进行筛选，便可以找到他们所需要的内容。

内容视图(Content Views)

活动菜单(Activity)

每个活动菜单表示一个系统提供的服务或定制服务——它可以通过访问活动视图控制器(Activity view controller)来作用于某些特定的内容。

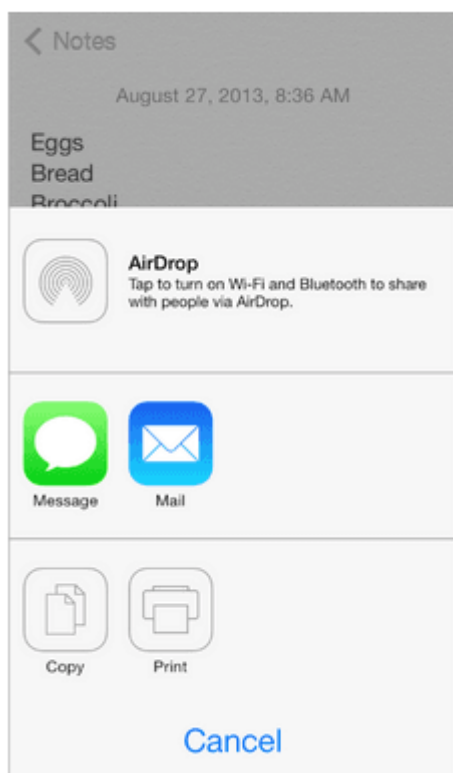


想要了解如何在代码中定义标签栏，请参考[UI Activity Class Reference](#)。

外观和行为

活动菜单指的是指一种代表当前应用所支持服务的对象。比如说，当用户点击分享按钮，应用程序将呈现活动视图控制器来告诉用户当前可以进行什么操作。想要了解如何将活动视图控制器整合进应用中，请参考[Activity View Controller](#)。

每个活动菜单都会以一个图标加图标下方文字的形式呈现。系统提供的活动菜单可以使用以下两种图标风格的任意一种：看起来更像一个应用图标的，或是看起来像工具栏标准图标，而第三方的活动菜单图标通常会选择后一种。这两种风格可以从下面的活动视图控制器中看到。



用户通常通过点击控制器中的活动图标来启动某样活动。点击之后该项服务通常会立刻执行，除非这项服务过于复杂，此时系统将会进一步索取更多的信息之后才会为用户执行该服务。

指南

使用活动菜单来让用户执行你的应用所提供的服务。请注意，iOS本身提供了若干内置的服务，如打印，转发到Twitter，发送信息和Airplay等等，你不需要再另外创建它们。

为你应用的各种服务设计一套精简的模版图标(**Template image**)。如果想制作出好看的模版图标，设计的时候可以遵循以下原则：

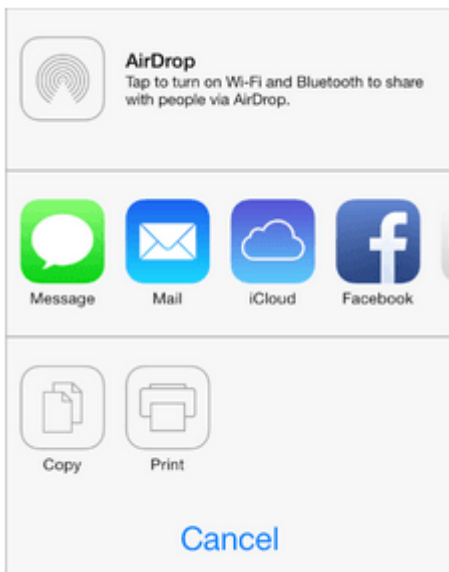
- 使用透明度适当的黑色或白色
- 不要使用阴影
- 进行抗锯齿处理

一个活动模版图大小应该保持在70×70像素左右(高分辨率下)，在区域里居中显示。

为每一个活动菜单设计清晰简练的文字标题。标题将会出现在活动菜单图标的下方。短标题通常效果最好，因为它在屏幕上的显示效果更好并且更容易本地化。如果你的标题文字过长，iOS会将缩小文本，仍然过长的话则会被截断。一般而言，最好能避免在活动标题中提及你的公司或产品名称。

活动视图控制器(**Activity View Controller**)

活动视图控制器是一个临时视图，当中罗列了一系列可以针对页面特定内容的系统服务和自定义服务。



想要了解如何在代码中定义活动视图控制器，请参考[UIActivityView Class Reference](#)。

外观和行为

活动视图控制器中列出了让用户操作当前内容的一系列可配置的服务。用户可以通过轻击分享按钮来查看活动视图控制器的内容。

活动视图控制器需要配合一系列的活动菜单来使用，每个活动菜单代表了一个特定的服务。想要了解如何设计一个自定义活动菜单，请查看上文的活动菜单(**Activity**)。

在iPhone和iPod Touch上，活动视图控制器在操作列表(**action sheet**)中出现；在iPad上则会出现在浮出层(**popover**)中。

指南

使用活动视图控制器来为用户提供一系列针对当前内容的服务。这些服务可以是系统自带的，比如复制，分享到**twitter**，打印等等，也可以是自定义的。活动视图控制器通常用作让用户把他们选中的内容复制到他们的社交媒体账户上。

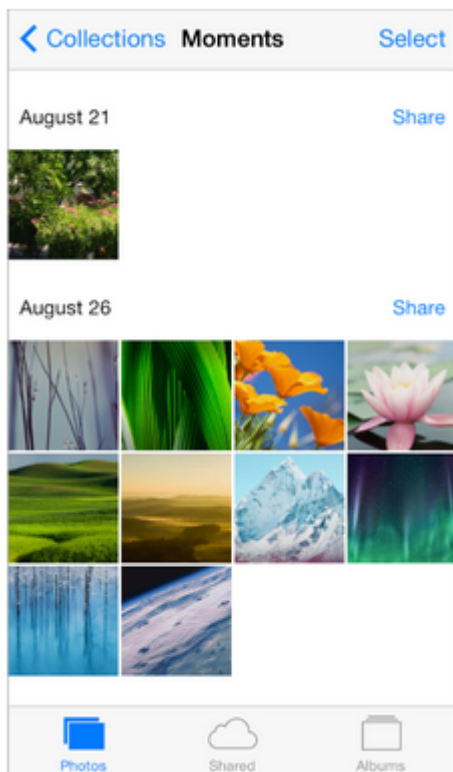
不要创建一个自定义按钮来触发活动视图控制器。用户更习惯点击分享按钮后使用系统提供的服务。你应该学会如何更好地利用用户这一既定习惯，而不是强迫他们以一种全新的方式来完成同样的事情。

确保控制器中的操作适用于当前场景。你可以适当地在活动视图控制器中增减系统操作，或增加自定义操作。例如，如果你不希望用户打印某张图片，你可以把打印功能从控制器中删除。

注意：你不能改变系统默认服务在控制器中的顺序。同时，所有系统服务都应出现在自定义服务之前。

集合视图(Collection View)

集合视图用于管理一系列有序的项，并以一种自定义的布局来呈现它们。



想要了解如何在代码中定义集合视图，请参考[Collection View Programming Guide for iOS](#).

外观和行为

集合视图是可以自定义的滚动视图，视图中罗列了用户可以对内容进行操作的一系列项。用户用手势来与视图中的各项进行交互，也可以进行导入、移动和删除等操作。

集合视图的整体布局与其中各项的外观样式是由集合视图与若干其它对象的代码共同定义的。这些对象中，布局对象（**layout object**）居于首位，它是[UICollectionViewLayout](#)的标准或自定义子类，定义了这些项的具体位置以及视觉属性。为了方便使用，UIKit提供了[UICollectionViewLayout object](#)，它为一组网格排列的项定义了可调整的线性排序。

在集合视图中，可选补充视图(**optional supplementary views**)可以在视觉上区分各项的子集。集合视图也支持装饰视图，也就是说你可以自定义它的背景和外观。

当用户在集合视图中导入、移动或者删除项的时候，会出现系统默认动画效果。集合视图同样支持开发者额外定义手势识别来执行自定义操作。默认情况下，集合视图可以识别轻击(**tap**)某项以选中，和长按(**touch-and-hold**)某项进行编辑。

在iOS 7里，集合视图支持开发者自定义各个布局间的转场动画。更多的详情可以参考[UICollectionViewTransitionLayout Class Reference](#).

指南

使用集合视图来让用户查看和操作一系列不适合以列表形式呈现的项。由于集合视图的布局不是一个严格的线性布局，因此尤其适合用来展示一些尺寸不一致的项。

集合视图支持广泛的自定义，因此我们要尽量避免把心思都放在进行全新的设计上。集合视图是用来帮助用户更好地完成任务的，视图本身并不是用户体验的焦点所在。

以下指南可以帮助你设计出用户体验更好的集合视图：

- 可以使用表格视图(**Table View**)的时候，不要使用集合视图。有时候用户会觉得以列表呈现的信息更容易阅读和理解，例如将文本信息放在滚动列表中滚动列表中的时候，用户阅读和处理起来会更为简单和高效。
- 让视图中的项更容易点击。如果用户很难点击集合视图中的项，他们是不会愿意用你的应用的。跟所有用户可以点击的UI对象一样，请确保你的集合视图中每一个项的最小点击区域有44×44pt，尤其是在iPhone上。
- 当你要让整个布局进行动态变化时，请务必谨慎。集合视图允许你在用户浏览和操作项的时候调整视图的布局。但当你决定调整它的时候，请确保这个动态变化是有意并且容易理解的。没有明确目的而贸然改变集合视图的布局会让用户对应用留下难用、不符合预期等负面的印象。更有甚者，如果用户此时关注的项在变化中消失了，用户会觉得这个应用超出了他们的控制能力。

容器视图控制器 (Container View Controller)

容器视图控制器采用自定义的方式来管理和呈现它的视图控制器或一系列子视图。系统定义的容器视图控制器包括[标签栏视图控制器\(Tab bar view controller\)](#)、[导航视图控制器\(navigation view controller\)](#)和[对分视图控制器\(split view controller\)](#)。

想要了解如何在代码中定义容器视图控制器，请参考[UIViewController Class Reference](#)。

外观和行为

容器视图控制器不存在任何预先定义好的外观或者行为。如果你在自定义容器视图控制器对象的时候把UIViewController归为子类，你可以自己规定它里头应该包含多少子类，以及它们将如何展现出来。

指南

用容器视图控制器来呈现内容，用户可以通过控制器来以自定义的方式进行导航。

先问问你自己是不是必须用到容器视图控制器。用户会更习惯诸如对分视图、或者是标签栏视图这类他们所熟知的东西。你必须确保你设计的控制器的优点不会由于用户不熟悉、不认识、不会用而白费功夫。

确保你的容器内容控制器在横屏与竖屏模式都可用。你的容器视图控制器无论在横屏还是竖屏中，体验都应该是一致的。

一般来说，避免太过花哨的转场动画。如果你采用了故事板(storyboard)的设计方法来设计你的视图内容控制器，你往往自然而然地会为它自定义一些动画。但绝大多数情况下，这些花哨的转场动画会让用户分心，让他们忘记了当前要做的事，还可能降低你的应用整体的美感。

图片视图(Image View)

图片视图用以展示一张单独的图片，或者一系列动态图片。

想要了解如何在代码中定义图片视图，请参考[image views](#)。

外观和行为

图片视图不存在任何预先定义好的外观，同时在默认状态下它不支持用户的交互行为。图片视图可以检测图片本身及其父视图(parent view)的属性，并决定这个图片是否应该被拉伸、缩放、调整到适合屏幕的大小，或者固定在一个特定的位置。

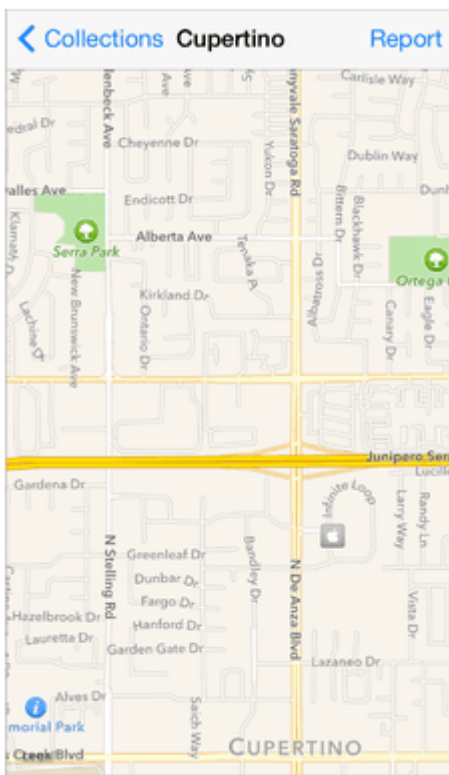
在iOS 7里，包含了模版图片(template image)的图片视图会把当前的色调(tint color)应用到图片上。

指南

请务必确保图片视图中的每一张图片都保持相同的尺寸和比例。如果你的图片尺寸各不相同，图片视图将会逐一对它们进行调整；而当你的图片比例不一，渲染的时候很可能会出错。

地图视图(Map View)

地图视图呈现地理数据，同时支持系统内置地图应用的大部分功能（如下图所示）。



想要了解如何在代码中定义地图视图，请参考[*Map Kit Framework Reference*](https://developer.apple.com/library/ios/documentation/MapKit/Reference/MapKit_Framework_Reference/index.html#//apple_ref/doc/uid/TP40008210)_。

外观和行为

地图视图通常以标准地图、卫星图像、或两者结合的形式来展示地理区域。地图样式还会标注单一的地点(annotations)，描绘路径和二维区域的轮廓。

用户可以缩放和移动地图视图——除非你在应用中禁用了这些动作——你也可以在编程时自定义地图视图的缩放和移动。

指南

利用地图视图可以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应用(routing app)，可以使用地图视图来展示你给用户的路径。

一般来说，允许用户在视图中进行交互行为。用户习惯了在系统内置地图中进行交互，因此他们会有预期，能在你所提供的地图中进行类似的行为。

使用标准的地图标注颜色。地图上标注了一系列地点。因为用户习惯了内置地图的各个标注的颜色，所以最好避免在你的应用中重新定义这些颜色的含义。定义颜色时，请遵循以下这些标准：

红色——表示目的地

绿色——表示起点

紫色——表示用户指定的地点(User-Specified Point)

页面视图控制器(Page View Controller)

页面视图控制器通过滚动(Scrolling)或翻页 (Page-curl transition style)来处理长度超过一页的内容。



想要了解如何在代码中定义页面视图控制器，请参考[Page View Controller](#).

外观和行为

带滚动条的页面视图控制器没有默认的外观。带翻页效果的控制器可以在两页中间增加书脊(book spine)的效果——当用户翻页时，它看起来就像一本真实的书在翻页一样。

页面内容控制器可以根据指定的切换效果来模拟出页面切换时的动画。使用滚动条效果的时候，当前页面将滚动到下一页；而使用翻页效果时，页面上会出现一个模拟实体书或笔记本翻页效果的翻页动画。

指南

使用页面视图控制器来展示那些线性的内容——例如一个故事的文本，或者是一些可以被自然地拆分成块的内容——比如说，日历。

页面视图控制器让用户从一页移动到前一页或者后一页，而并不支持用户在并不相邻的页面间快速切换。如果你希望在页面视图控制器中展示一些非线性的内容——比如说字典，或者书籍的目录——那么你就需要自定义一种方式，让用户可以随意地到达不同的内容区块。

滚动视图(Scroll View)

滚动视图方便用户浏览尺寸超越滚动视图边界的图片（下图中地球的图片无论是长度还是宽度都超过了）。

想要了解如何在代码中定义滚动视图，请参考“Scroll Views”。



外观和行为

滚动视图刚出现或者当用户在对它进行操作的时候——纵向和横向的滚动指示器会短暂地闪烁，以提示用户在当前视图外仍有更多内容。跟瞬态滚动条(transient scroll indicator)不一样，滚动视图没有预定义的外观。

滚动视图的响应速度和对各个操作手势的识别都应当让用户感到自然。当用户在视图中拖拽内容，内容随之滚动；当用户轻扫屏幕时，内容将快速滚动——直到用户再次触摸屏幕或内容已经到达底部时停止。滚动视图同样可以应用在页模式(paging mode)中，在此模式下用户可以通过拖拽和轻击等手势来浏览一页的内容。

指南

你可以使用滚动视图来允许用户在固定的空间内浏览大尺寸或大量的视图。因为在iOS中用户常常会用到滚动视图，所以请确保你的应用中滚动视图的操作与他们的预期相同。

适当地支持缩放操作。如果放大和缩小对于当前内容是有用的话，你可以支持用户通过捏或者双击来对当前视图进行缩放。而若是支持了缩放操作的话，你还应当设定在当前情景下允许缩放的最大值和最小值。如果你允许一个字符被放大到充满整个屏幕的话，用户会很难阅读当前内容。

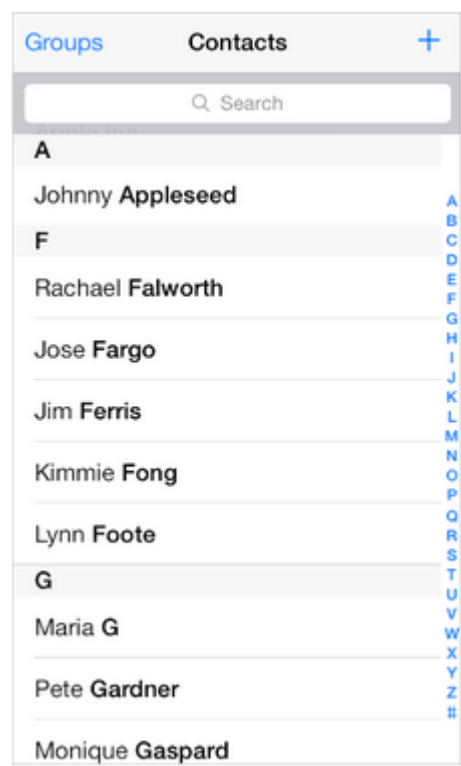
在页模式滚动视图中，可以考虑使用页面控件(page control)。当你想要展示分页、分屏或者分块的内容，最好是让用户知道当前内容一共有多少块，以及他们当前浏览的是第几个——页面控件以圆点的形式来把这个信息告诉用户。同时由于页面控件被广泛用于Safari，股票，天气以及其它系统内置应用中，用户很容易理解它的用途。

当你在滚动视图中使用页面控件的时候，最好禁用同一方向的滚动指示器(scroll indicator)。这样一来用户就有了一种唯一且清晰的方式来浏览当前内容。想要了解更多请参考网络视图(Web View)。

一般来说，一次只展示一个滚动视图。如果在一屏中同时存在不止一个滚动视图，由于用户滚动屏幕时动作幅度经常都会很大，他们很容易会碰到另一个。如果你确实要在同屏中放两个滚动视图，可以考虑给他们设定不同的滚动方向，来避免用户想要滚动一个视图的时候误操作。比如iPhone上的股票应用，纵向滚动上半部分会展示股票报价，横向滚动下半部分时则展示该公司的特定信息。

表格视图(Table View)

表格视图以单列多行的形式来展示数据。



想要了解如何在代码中定义表格视图，请参考[Table View Programming Guide for the iOS](#)以及[Table Views](#).






外观和行为

表格视图中的数据是以单列的方式展示的，因此也很容易将它们分段或者分组。用户通过轻击或者拖拽来浏览不同组的信息。用户可以点击来选中某行，通过控件来添加、移除、多选、查看详情或者展开另一个表格视图。当用户选中某一行时，该行会短暂地高亮。

当选中某行将展开另外一屏内容的时候，该行会短暂地高亮，然后新一屏内容滑入。当用户回到前一屏时，之前选中的那一行同样会短暂地高亮，提醒用户他们先前选中了什么。

iOS定义了两种表格样式。

平铺型（**Plain**）表格可被分为若干带标签的段落，表格右侧可能会出现垂直的表格索引。每行开头可以有页眉，尾部可以有页脚（也可以没有）。

More		Edit
	Audiobooks	>
	Tones	>
	Genius	>
	Purchased	>
	Downloads	>

分组型（**Grouped**）中至少含有一组列表，而每一组中至少包含一项内容。分组可以有页眉和页脚。与平铺型不同，分组型表格没有索引。

< Settings	Maps
DISTANCES	
In Miles	✓
In Kilometers	
MAP LABELS	
Always in English	<input checked="" type="checkbox"/>
PREFERRED DIRECTIONS	
Driving	✓
Walking	

iOS提供了若干表格视图元素(table-view elements)来扩展表格视图的功能。除了特别标明外，这些元素只适用于表格视图。

表 35-1 表格视图元素(Table View Elements)

按钮	名称	含义
✓	选中(Checkmark)	表示当前行已被选中
>	展开(Disclose Indicator)	在新视图中展示关于当前行的更多信息(适用于表格视图以外)
i	展 开 详 情 按 钮 (Detail Disclosure button)	在新视图中展示关于当前行的更多信息(适用于表格视图以外)
≡	行记录器(Row Recorder)	表示当前行可以被拖拽到表格的另一位置
+	增加行(Row insert)	在表格中新增一行
-	删 除 按 钮 控 件 (Delete button control)	在编辑态中，显示和隐藏删除按钮
	删除按钮(Delete button)	删除当前行

除了以上表格中列举的元素外，iOS定义了刷新控件，让用户可以刷新当前的表格内容。想要了解更多关于刷新控件的用法，可以参考文档本章第三节控件中的Refresh Controls。

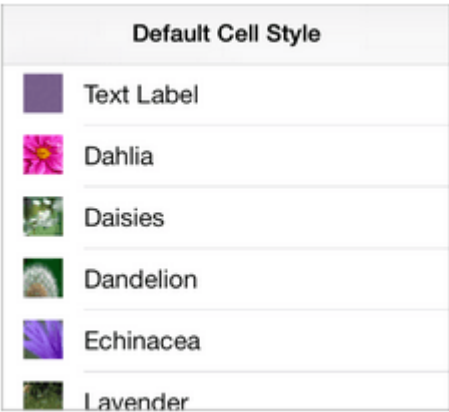
iOS定义了 在平铺型表格和分组型表格中最常用到的四种单元格布局样式。每种单元格样式都有最适合展示的信息类型。

注意：从编程角度来说，这些样式应用于单元格中，用以控制表格里每一列的绘制方式。

默认型（Default）([UITableViewCellStyleDefault](#))

默认型样式包括左侧的图标（可选），和图标右边左对齐的文字标题。







默认型样式适合展示一系列无须通过附加信息便可以区分的项。



副标题型（Subtitled）([UITableViewCellStyleSubtitle](#))







副标题型包括左侧图标（可选），图标右边左对齐展示的文字标题，以及在标题下方同样左对齐展示的副标题。

左对齐的文本标签让用户可以更快速地扫视表格。这种样式适用于列表各项较为相似的情况，用户可以通过副标题中的详细信息来区分列表中的各项。

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 ([UITableViewCellStyleValue1](#))

在Value 1样式下，标题左对齐，副标题用较细的字体右对齐。

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 2 ([UITableViewCellStyleValue2](#))

在Value 2样式里，蓝色标题右对齐，副标题左对齐，混排在同一行中。这种样式通常不包含图片。

Value 2的布局中，文本和副标题中间的垂直间距会让用户专注于副标题的第一个单词。

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：以上四种单元格样式均支持添加表格视图元素，如勾选或展开标志。添加这些元素会缩小标题以及副标题单元格的可用宽度。

指南

使用表格视图可以简洁而高效地展示少量或者大量信息。举例来说，你可以通过表格视图来：

展示用户可选的选项列表。你可以使用选中标记来告知用户当前选中了哪些项。

展示层级信息。平铺型表格样式非常适合展示层级信息。表格中的每项都指向不同的子信息，这些子信息承载于另一个列表中。用户可以沿着这些层级结构的路径来点击每一层列表中的项。展开标志告知用户点击这一列中的任何位置，都将展开新的列表以展示其子类信息。

展示可以在概念上进行分组的信息。平铺型和分组型列表都允许你通过提供页眉和页脚来对信息进行分组和分段。

对于iOS 6.0以上的版本，你可以用页眉页脚视图(header-footer view)——即UITableViewHeaderFooterView中的一个实例——来展示页眉和页脚的文本，或图片。想要了解如何在代码中定义页眉页脚视图，请参考[UITableViewHeaderFooterView Class Reference](#)。

展示索引来方便查找。平铺型列表支持索引，可以让用户快速找到需要的内容。索引信息包含一个浮在屏幕右侧的、纵向罗列的条目，内容则通常是字母表中的字母。

如果你使用了索引，要避免在表格右侧使用其它表格视图元素——比如展开指示符——因为它们与索引是冲突的。

使用表格视图时可遵循以下指南：

用户选择列表项时，始终给与反馈。当用户点击可选的列表项时会认为被点击的项都应短暂地高亮一下。在点击后，用户期望出现新的视图，或者出现一个复选标记以表明先前点击的项已经被选中或激活。

如果表格的内容庞大而且复杂，不要等数据都加载完之后才一起显示出来。可以首先展示文字信息，图片等较为复杂的内容则在加载完后再显示。这样可以将有用的信息立即传达给用户，同时也提高了应用的响应能力。

在等待信息加载的时候，可以考虑展示“过期”信息。尽管我们并不推荐在数据频繁变化的应用中这样做，它还是可以帮助更多的静态应用程序立即给到用户有用的信息。当然在你这么做之前，请认真衡量你应用中数据的变化频率，并弄清楚你的目标用户有多需要立即获取最新的信息。

如果信息加载速度很慢或者非常复杂，你需要告诉用户加载正在进行中。如果表格中所有内容都很复杂，我们很难即时地给用户展示任何内容。在这种极端情况下，切勿显示空白的表格，因为这会让用户以为应用挂了。此时应当在屏幕中央展示一个活动指示器(activity indicator)和一个信息标签(information label)，比如“加载中...”，让用户知道加载仍然在进行。

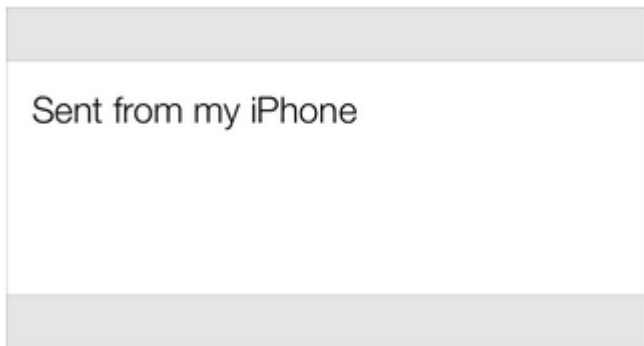
如果合适的话，为删除按钮自定义一个名称。如果这能让用户更好地理解应用的相关功能的话，你可以创建一个合适的标题，来取代“删除”这个字样。

尽量使用简洁的文字标签，以避免被截断。繁冗的文字和词组不方便用户浏览和理解。以上所有单元格样式均会自动截断文本，而文本截断所造成的问题可大可小，取决于你采用的单元格样式，以及被截断了哪一部分文字。

如果你想以一种非标准的形式来布局你的表格，最好是自定义一种单元格样式，而不是在现有的表格样式上进行改动。如何创建自定义单元格样式，请参考[Table View Programming Guide for iOS](#)中的[Customizing Cells](#)部分。

文本视图(Text View)

文本视图可以接收和展示多行文本。



想了解如何在代码中定义文本视图，参考[Text Views](#)。

外观和行为

文本视图是矩形，可定义为任何高度。当内容太多超出视图的边框时，文本视图支持滚动。

如果文本视图支持用户编辑，当用户轻击文本视图内部时，将唤起键盘。键盘的布局 and 类型取决于用户的系统语言设置。

指南

你可以控制文本视图中文字的字体、颜色和对齐方式。文本视图的默认字体是系统字体，默认字色是黑色。默认文字对齐方式是左对齐（你可以改为居中或右对齐）。

始终确保文字的易读性。虽然你可以使用属性字符串将不同的字体、字色和对齐方式串联在同一个文本视图内，但保持文本的可读性是必不可少的。最好是可以支持动态文本(Dynamic Type)和UIFont method `preferredFontForTextStyle`来展示文本框中的文本。

根据输入内容的类型来指定不同的键盘类型。举例来说，你希望用户能更方便地输入网址、密码或者电话号码。但请注意，由于键盘的布局以及输入方法是由用户的系统语言设置决定的，这是你不能控制的。iOS提供了各种不同的键盘类型，以便用户输入不同类型的文本。想要了解可用键盘类型，可以参考[UIKeyboardType](#)。想要了解如何在管理你的应用中的键盘，请参考[iOS App Programming Guide](#)中的Managing the Keyboard部分。

网络视图(Web View)

网络视图是一个可以展示丰富的HTML内容的区域。（下图是iPhone自带的邮件应用，网络视图指的是下图中导航栏和标签栏中间的区域）



想要了解如何在代码中定义网络视图，请参考[Web Views](#)。

外观和行为

除了展示网络内容外，网络视图还会自动处理页面中的内容，比如把页面中的电话号码转化成电话链接（译者按：phone link，点击之后iPhone将自动拨打该号码）。

指南

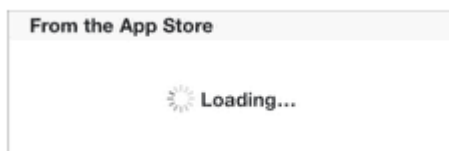
如果你经营一个网页或者网络应用，你大约会用网络视图来实现一个简单的iOS App，来对你的网页或者应用进行一个封装。如果你打算用网络视图来访问你所控制的网页内容，请务必阅读[Safari Web Content Guide](#)。

不要用网络视图来创建一个看起来像迷你网络浏览器的应用。用户期望使用iOS自带的Safari来浏览网页内容，因此我们并不推荐你在自己的应用中复制这种以被广泛应用的功能。

控件(Controls)

活动指示器(Activity Indicator)

活动指示器表明任务或进程正在进行中，如下图所示。



API提示：想要了解如何在代码中定义活动指示器，可以参考[UIActivityIndicatorView Class Reference](#)。

活动指示器：

- 当任务进行和加载时旋转，任务完成后自动消失
- 不支持用户交互行为

在工具栏或主视图中使用活动指示器来告知用户任务或加载正在进行中，但并不提示该过程何时会结束。

不要使用静止的活动指示器。用户会以为该进程停滞了。

用活动指示器来让用户知道进程仍在进行中。有些时候，告诉用户进程没有停止比告诉他们何时完成更加重要。

可以的话，最好可以设计一个与应用的风格协调的活动指示器。

添加联系人按钮(Contact Add Button)

添加联系人按钮让用户将现有联系人添加到文本框或者其它文字视图中。



API提示：想要了解如何在代码中定义添加联系人按钮，参考[Buttons](#)。

添加联系人按钮：

- 展示联系人列表
- 帮助用户将一个联系人添加到当前联系人按钮所在的视图中

使用添加联系人按钮让用户在不需要使用键盘的情况下就可以方便地访问到联系人。举个例子，在新建邮件的界面中，用户可以点击该按钮来添加收件人，而不需要用键盘输入收件人的名字。

由于添加联系人按钮属于键盘输入联系人方法的替代品，我们不推荐在不支持键盘输入的界面中使用添加联系人按钮。

日期时间选择器(Date Picker)

日期时间选择器展示关于日期和时间的组件，比如小时，分钟，天，以及年。

Mon Sep 2	6	57	
Tue Sep 3	7	58	
Wed Sep 4	8	59	
Today	9	00	AM
Fri Sep 6	10	01	PM
Sat Sep 7	11	02	
Sun Sep 8	12	03	

API提示：想要了解如何在代码中定义日期选择器，请参考 [Date Pickers](#)。

日期时间选择器：

- 最多可以展示4个独立的滑轮，每一个滑轮表示一个不同的值，比如月份或小时等
- 在每个滑轮的中央使用深色字体来表示当前选中的值
- 日期时间选择器的大小与iPhone键盘的大小相同，并且不可更改
- 包括四种模式，每一种模式代表了一组不同的值：
 - 日期和时间。日期和时间模式（默认模式）包含日期、小时、和分钟，以及一个可选的AM/PM值。
 - 时间。时间模式包括小时和分钟，以及可选的AM/PM值。
 - 日期。日期模式包括月份，天以及年三个值。

· 倒计时器。倒计时器模式展示了小时和分钟值。你可以精确地设定总共的倒计时时间，倒计时的最大值为23小时59分钟。

使用日期时间选择器来让用户选择时间，而不是让用户自己输入一个包含了日期、时间等多个部分的时间值。

尽量地让用户在当前内容中使用日期选择器。最好避免用户在使用日期选择器的时候要进入另外一个界面。在iPad上，日期时间选择器可能会出现在一个浮层中，或者嵌入在当前内容里。

有必要的时候，改变分钟滑轮的单位刻度。在默认情况下，分钟滑轮包含从0到59共60个值，如果你要展示一个颗粒度较大的时间，你可以让分钟滑轮的单位刻度变大，只要这个刻度可以整除60。比如说你可能会设定每15分钟为一个刻度，此时分钟滑轮就有4个值，0、15、30、45。

详情展开按钮(Detail Disclosure Button)

详情展开按钮展示了与该项相关的更多详细信息与功能描述。



API提示：想要了解如何在代码中定义详情展开按钮，可以参考 [UITableViewCell Class Reference](#) 和 [Buttons](#)。

详情展开按钮以一个单独的视图展示特定项目的更多详情信息与功能。

当详情展开按钮在表格行中出现时，点击表格行的其它区域不会激活此按钮，只会选中该行，或者触发app中其它自定义的行为。

一般来说，你会在一个表格视图中使用详情展开按钮来让用户知道更多关于这个列表项的信息。当然你也可以将这个按钮用在其它类型的视图中来为用户展示更多与特定项目相关的信息和功能。

信息按钮(Info Button)

信息按钮展示了app的配置信息，有时候它会出现在当前视图的背面。



API提示：想要了解如何在代码中定义信息按钮，可以参考[Buttons](#)。

iOS包含了两种信息按钮样式：适用于浅色内容上的深色按钮，以及适用于深色内容上的浅色按钮。

使用信息按钮来显示app的配置信息或选项。你可以根据自己app的UI风格来选择最为协调的信息按钮样式。

标签(Label)

标签用于放置静态文本。

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

API提示：想要了解如何在代码中定义标签，可以参考[UILabel Class Reference](#)。

标签可以：

- 展示任意数量的静态文本
- 禁止除了复制文本外的任何用户交互行为

你可以使用标签来命名或解释你的部分UI，又或者用它来给用户提供一些简单的信息。标签最适合拿来展示相对简单的文本信息。

保证你的标签清晰易读。最好支持动态文本(Dynamic Type)，并使用 `UIFont` 中的 `preferredFontForTextStyle` 来获得标签中的展示文本。如果你要用自定义字体的话，请慎重选择字体种类，不要以牺牲清晰度为代价来换取花哨的颜色和字体效果。（想要了解关于app中字体使用的指南，可以参考 [Color and Typography](#)；想要了解更多动态文本的内容，可以参考 **Text Programming Guide for iOS** 里面的 [Text Styles](#) 部分。）

网络活动指示器(Network Activity Indicator)

网络活动指示器在状态栏中出现，表示网络活动正在进行。



API提示：你可以在代码中使用 `UIApplication` Method `networkActivityIndicatorVisible` 来控制该活动指示器的可见性。

网络活动指示器：

- 出现在状态栏中，当网络活动正在进行时它会旋转，在活动停止时它则消失
- 不支持用户交互行为

当你的app正在链接网络，而这个连接过程将会持续好几秒的时候，你可以通过网络活动指示器来给用户以反馈。如果进程所需时间很短，则不需要用到它，因为很可能在用户注意到它之前，它就消失了。

页面控件(Page Control)

页面控件告诉用户当前共打开了多少个视图，还有他们正处在其中哪一个。



API提示：想要了解如何在代码中定义页面控件，可以参考 [Page Controls](#)。

页面控件：

- 包含一系列圆点，圆点的个数代表了当前打开的视图数量（从左到右，这些圆点代表了视图打开的先后顺序）。
- 默认情况下，使用不透明点来标识当前打开的视图，使用半透明点来表示所有其它视图。
- 不支持用户访问不连续的视图
- 当视图数量超过页面宽度可承载的氛围时，点的大小和间距并不会因此变小，如果需要显示的点超过一定数量，系统会把它截断。

当你的app中所有的视图都属于同级的时候，你可以使页面控件。

当你的app结构存在信息层级，请不要使用页面控件。因为页面控件不能让用户跟踪自己的访问路径，回到上一级。

将页面控件垂直居中放置于当前打开视图的底边与屏幕底边之间，这样可以保证它的可见性而又不会对内容造成干扰。避免展示太多的点，一般来说，iPhone竖屏方向最多可以容纳差不多20个点。

选择器(Picker)

选择器展示了一组值，用户可以从选择一个。



API提示：想要了解如何在代码中定义选择器，请参考 [UIPickerView Class Reference](#)。

选择器：

- 是日期时间选择器的通用模式
- 包括一个或多个滑轮，每个滑轮含有一组值
- 当前选中的值在中间，以深色标识
- 不可以自定义大小（选择器的大小与iPhone的键盘相同）

使用选择器可以让用户更容易从一系列不同的值中间进行选择。

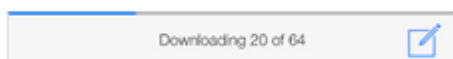
一般来说，当用户对整组值都比较熟悉的时候，可以使用选择器。由于当滑轮静止的时候，大部分的数值会被隐藏，最好是在用户对所有数值均有预期的情况下才使用选择器。当你需要展示一大组用户并不熟悉的选项，此种选择器可能不太适合。

尽可能让让用户在当前视图中使用选择器。不要让他们在使用选择器时还要进入其它的视图。

如果你需要展示的备选项数量很多，考虑使用表格视图(**Table View**)而不是选择器。因为表格视图的高度较大，内容滚动起来会更快。

进度视图（**Progress View**）

进度视图展示了任务或进程的进度（下图是iOS默认邮件App的工具栏）。



API提示：想要了解更多如何在代码中定义进度视图，参考 [UIProgressView Class Reference](#)。

进度视图：

- 是一条轨迹，随着进程的进行从左向右进行填充
- 不支持用户交互行为

iOS定义了两种进度视图样式：

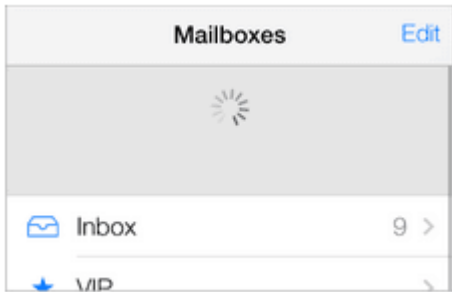
- 默认(**Default**). 默认样式适合用在app的主要内容区中。
- 进度条(**Bar**). 此样式比默认样式细，适合用在工具栏中。

当一个任务存在明确的进程，可以使用进度条来给与用户反馈，尤其是告诉用户这个任务大约需要多少时间才能完成。

可以的话，请根据你的app的风格来设计进度条的外观。你可以自定义进度条的底色以及轨迹颜色，也可以直接使用图片。

刷新控件(Refresh Control)

刷新控件执行用户触发的内容刷新——一个典型的例子，它常在表格中出现（下图展示的是iOS默认的邮件app的mailbox列表页）。



API提示：想要了解如何在代码中定义刷新控件，请参考 [UIRefreshControl Class Reference](#)。

刷新控件：

- 看起来类似活动指示器
- 可以出现在标题中
- 默认状态下不可见，当用户在表格上缘往下拖拽以刷新内容时才出现

就算你使用了刷新控件，也不要因此就不支持内容自动刷新。尽管用户喜欢在执行刷新操作时内容立刻刷新，他们也同样会喜欢内容自动刷新。如果过于一来用户自己执行所有刷新操作的话，那些不会自动刷新的用户就会疑惑，为何你app中的数据永远都不更新。一般来说，刷新控件给了用户多一个选择，让他们可以立刻获得最新的内容，但同时，你也不能奢望用户会主动获取所有的更新信息。

只有在必要的时候才加短标题。特别需要注意的是，不要使用短标题来描述刷新控件怎么使用。

圆角矩形按钮(Rounded Rectangle Button)

iOS 7已经不再使用圆角矩形按钮，而是使用了新的系统按钮——类型为UIButtonTypeSystem的UI按钮([UIButton](#))。使用指南可参考 [System Button](#)。

分段控件(Segmented Control)

分段控件是一组分段的线性集合，每一个分段的作用类似按钮，点击之后将切换到相应的视图。



API提示：想要了解如何在代码中定义分段控件，请参考 [Segmented Controls](#)。

分段控件：

- 由两个或以上的分段组成，每一个分段的宽度相同，与分段的数量成比例（分段数量越多，则宽度越小）；
- 可以包含文字或者图片

使用分段控件来提供密切相关而又互斥的选项。

保证每个分段都容易点击。为了保证每个分段的大小至少有44×44像素，请控制分段的数量。在iPhone上，1个分段控件最多包含5个分段。

尽可能地保持每个分段中的文字长度一致。因为每个分段都是等宽的，当文本长度差异很大时看上去会很失调。

不要在同一个分段控件中混用文字和图片。每一个分段都仅可支持纯文字或纯图片。避免在同一个分段控件中，一些分段里使用纯文字，另一些分段里使用纯图。

如果你自定义了分段控件的外观，请在必要时调整分段控件中文本的对齐方式。如果你给分段控件添加了自定义底图，请确保控件里自动居中的文本依然清晰美观。你可以通过 **bar metrics APIs** 来调整分段控件内文本的对齐方式(想要了解如何定义 **bar metrics**，可以参考 [UISegmentedControl](#) 中关于自定义API外观(**appearance-customization APIs**)的描述)。

滑块(Slider)

滑块允许用户在一个限定范围内调整某个数值或进程(下图展示的是iOS设置中亮度设置的滑块，滑块的左边和右边均为自定义图形)。



API提示：想要了解如何在代码中定义滑块，请参考 [Sliders](#)。

- 由一条水平的轨迹和一个**Thumb**(滑块中支持用户水平拖拽的圆形控件)组成
- 左边和右边支持使用自定义图片来表述相对的最小值与最大值的含义
- 填充轨道左边缘最小值之间到**Thumb**之间的部分

使用滑块来让用户精准地选择自己想要的值，或者控制当前的进程。

如果合适的话，自定义滑块的外观。比如，你可以：

- 定义**Thumb**的外观，让用户一看就知道滑块当前的状态
- 在轨迹的左右两端使用自定义图片来告诉用户滑块的最小值和最大值所代表的含义。比如说，一个图调整图片尺寸的滑块可以在最小值的左边放一张小图，在最大值的右边放一张大图。
- 根据**Thumb**所在的位置和当前滑块的状态来为滑块的轨迹定义不同的颜色

步进器(Stepper)

步进器可以以常数为幅度来增减当前数值。



API提示：想要了解如何在代码中定义步进器，请参考 [Steppers](#)。

步进器：

- 是一个两段控件，其中一段默认显示减号，另一端默认显示加号
- 支持自定义图片
- 不展示用户更改的值

当用户想要对数值进行小幅度调整时，可以使用步进器

当用户需要大幅度调整数值的时候，不要使用步进器。用户可能会在打印机里使用步进器来确定打印份数，因为这个值的变化幅度通常并不大；而当用户需要选择打印的页码范围时，使用步进器就会让操作变得繁琐，因为用户很可能要点很多下才能选定页数。

确保步进器所调整的值明显可见。步进器自身不展示任何数值，所以你需要保证让用户知道他们正在调整哪一个数值。

开关按钮(Switch)

一个开关按钮展示了两个互斥的选项或状态。



API提示：想要了解如何在代码中定义开关，参考 [Switches](#)。

开关按钮：

- 显示了一个项存在二元状态
- 仅在表格视图中可用

在表格中使用开关按钮来让用户从某一项的两个互斥状态中指定一个，比如是/否(Yes/No)，开/关(On/Off)。

你可以使用开关按钮来控制视图中的其它UI元素。根据用户的选择，新的列表项可能出现或者消失，或从激活状态变为不激活状态。

系统按钮(System Button)

系统按钮执行app中定义的行为。

Button

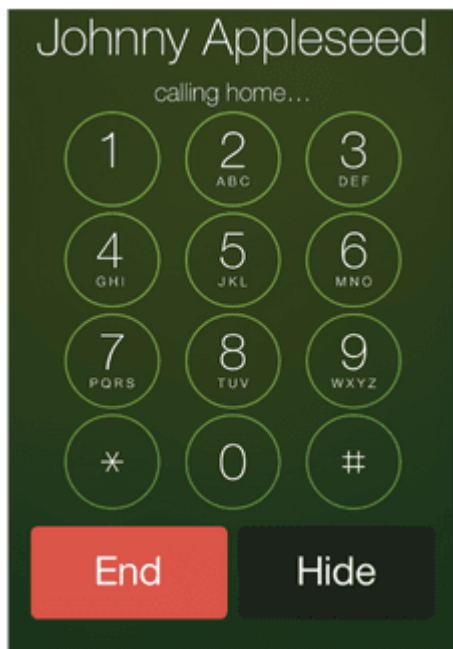
API提示：在iOS 7中，UIButtonTypeRoundedRect已经被重新定义为 `UIButtonTypeSystem`。如果在iOS 6中使用了圆角矩形按钮，在连接到iOS 7的时候会自动替换为新的系统按钮。想要了解如何在代码中定义系统按钮，参考 [Buttons](#)。

系统按钮：

- 默认状态下不含边界，也不含背景图
- 可以是图标或者文字标题
- 支持自定义样式，如描边或者加背景图(想要自定义按钮外观，可以使用 `UIButtonTypeCustom` 类型的按钮，并且提供背景图片)

使用系统按钮来执行某个动作。当你为系统按钮命名时，请遵循以下方法：

- 使用动词或动词短语来描述按钮所代表的动作。这种命名方法告诉用户这个按钮是可交互的，也提示了用户点击之后会执行什么操作
- 使用标题式大写(title-style capitalization，每个单词的首字母均大写)。除了冠词，并列连词以及少于4个字母的介词外，标题中每个单词的首字母均大写。
- 标题不要太长。太长的标题会被截断，让用户难以理解其含义。

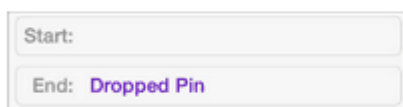


合适的话，为内容区域内的系统按钮描边或者加入背景。大多数情况下，你可以通过定义一个清晰的按钮名称、选择一个不一样的标题颜色或提供上下文情景提示来让用户知道这是一个按钮而非普通文本。但在某些特定的内容区域内，为按钮描边或者添加背景颜色，让用户迅速地把注意力放到按钮上，也是必要的。

以iPhone为例，给数字按键添加圆形边框强化了用户拨电话号码时的心理模型，而结束(End)和隐藏(Hide)按钮的背景色让用户拥有了更大的点击范围。

文本框(Text Field)

文本框支持用户输入单行的文本。



API提示：想要了解如何在代码中定义文本框，以及在文本框中支持图片和按钮，请参考 [Text Fields](#)。

文本框：

- 高度固定，包含圆角
- 当用户点击它时，自动唤起输入键盘
- 可以包含系统提供的按钮，如书签按钮(Bookmarks)
- 可以展示多种文字样式(了解更多请参考 `UITextView`)

使用文本框来获取用户输入的少量信息。

你可以自定义一个文本框，帮助用户更好地理解如何使用它。举个例子，你可以在文本框的左侧或者右侧加入自定义图形，或者加入系统按钮，如书签按钮等。一般来说，文本框的左侧用于表述文本框的含义，而右侧用于展示附加的功能，如书签。

合适的话，在文本框右侧加入清除按钮。轻击清除按钮便可清空当前框内输入的全部内容，无论你原本打算在这个按钮上面展示什么其它图片。

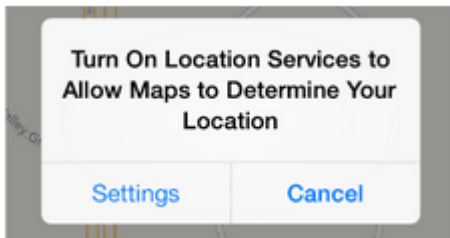
如果可以帮助用户理解的话，可以在文本框中加入提示文字。当文本框里没有任何其它提示文字时，会展示占位符文本(**placeholder text**)，如名字、地址等。

根据输入内容的类型来指定不同的键盘类型。举例来说，你希望用户能更方便地输入网址、密码或者电话号码。iOS提供了各种不同的键盘类型，以使用户输入不同类型的文本。想要了解可用键盘类型，可以参考 [UITextInputTraits Protocol Reference](#) 中的 [UIKeyboardType](#)。想要了解如何在管理你的应用中的键盘，请参考 [iOS App Programming Guide](#) 中的 Managing the Keyboard 部分。但请注意，由于键盘的布局以及输入方法是由用户的系统语言设置决定的，这是你不能控制的。

临时视图(Temporary Views)

警告框(Alert)

警告框用于告知用户一些会影响到他们使用app或设备的重要信息。



API提示：想要了解如何在代码中定义警告，参考 [UIAlertView Class Reference](#)。

警告框：

- 必须包含标题，有时候会包含正文文本
- 包含一个或多个按钮

一般来说，警告框警告出现的频率较低，也正因为如此，警告的出现通常会让用户额外重视。请严格控制你的app中警告的个数，并且保证每一个警告都能提供重要的信息，或者有用的选项。

避免出现不必要的警告框。一般来说，在以下情景中，是不需要用到警告框的：

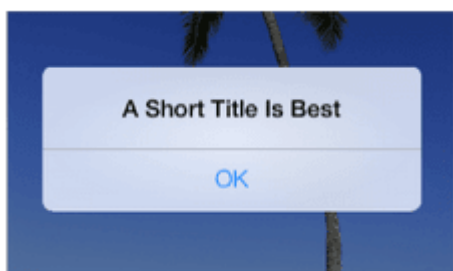
若你使用警告来做以下这些事	可以这样做来代替警告
提供与 app 的标准操作相关的信息	设计一种与你的应用风格相协调，但又引人注目的信息展示方式
更新用户正在进行的任务	使用进程视图(Progress View, 可参考上文"控件"中的进程视图部分)或者活动指示器(可参考上文"控件"中的活动指示器部分)，或者将此状态信息融入 app 的 UI 之中
让用户确认触发的任务	使用操作列表(Action Sheet,可参考下文 Action Sheet 部分)
告知用户此时产生了无法解决的问题	如果问题并非决定性的，可以把信息融合到 app 的 UI 中 ;如果问题是决定性的，则使用警告

当你在设计警告文案的时候，了解以下这些定义非常有用：

- 标题式大写 (**Title-style capitalization**)指的是除了冠词，并列连词以及少于4个字母且不处在第一个单词位置上的介词外，标题中每个单词的首字母均大写。
- 句子式大写 (**Sentence-style capitalization**)指的是第一个字母大写，其余除了专有名词和专有形容词外的字母均小写

简明扼要地描述当前情景，并告诉用户他们可以做什么。理想情况下，警告框中的文字应该给与用户足够的情景和上下文联想，让他们可以清楚地知道为什么警告会出现，同时帮助他们判断自己应该点哪个按钮。

保证标题足够简短，最好在一行之内。过长的标题让用户很难快速理解它的意思，还可能会被截断。



如果可以的话，使用句子片段而非完整的句子。一个简洁清晰的状态描述往往比一个完整的句子更容易理解。

尽可能的精炼你的标题文字，让警告框即使没有下面的正文信息也能完全让用户理解。举个例子，当你使用一个问题，或者两个短句来作为警告框标题的话，很可能你并不需要添加文本信息。

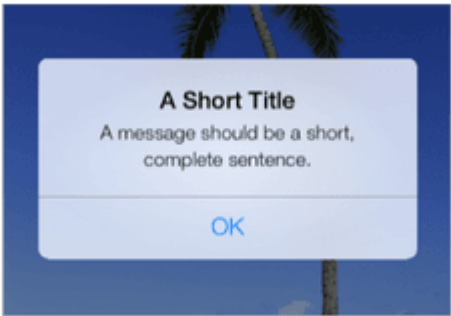
不用刻意避免在警告框中使用消极负面的文案。用户们理解大多数警告框是为了告诉他们发生的问题，或者对他们目前的状态作出警告。因此消极但清晰直接的文案优于积极但晦涩间接的文案。

尽可能地避免使用"你"，"你的"，"我"，"我的"这类字眼。有时候，这些直接指向的字眼容易引起歧义，有时候甚至会被误认为是一种冒犯。

适当地使用大写和标点符号，尤其是在以下这些场景中：

当警告的标题符合下面这些情况	使用以下方法
句子片断或非疑问句	标题式大写，句末不需要加标点
疑问句	句子式大写，句末加问号
由两个或以上句子组成	句子式大写，每句的末尾添加适当的标点符号

如果你必须为警告框添加正文文本，请使用一个完整的短句。可能的话，尽量保证句子在1到2行之间。如果句子太长，用户会需要滚动才能看完，这样的体验很糟。使用句子式大写，并在句末加上适当的标点符号。



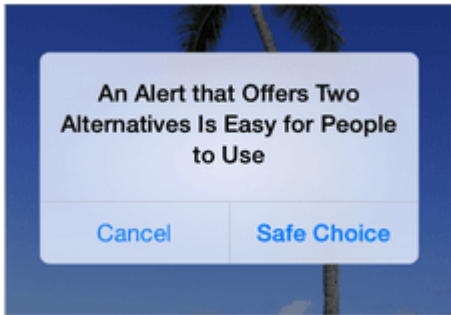
**** 避免在文本中详细描述“该按哪个按钮”而导致文本过长。****理想情况下，表意明确的警告文案和逻辑清晰的按钮文案已经足以让用户正确判断自己该按哪个按钮了。但如果你一定要在文案中描述这些内容，请遵循以下原则：

- 确定使用轻击(**tap**)来描述这个选择操作，不要用触摸(**touch**)、点击(**click**)或者选择(**choose**)这类字眼。
- 不要用引号，但保证大写

确保警告框在竖屏和横屏中均显示正常。横屏模式下警告框的高度会受到限制，其大小与竖屏下可能会有区别。我们推荐您限定好警告框的最大高度，保证在竖屏和横屏模式下文字均能不需要滚动便可完整地显示。

一般情况下，使用两个按钮的警告框。两个按钮的警告框是最为常见和有用的，因为它最便于用户在两个按钮中做选择。单按钮警告框不那么有用，因为它通常只是起到告知的作用，并未给予用户控制当前状态的能力。多于两个按钮的警告框太过复杂，应该尽可能地避免使用。如果你在警告框中设计了太多按钮，它也许会导致警告框被强制滚动，这也是一个非常糟糕的体验。

提示：如果你需要在警告框中给与用户超过2个选项，可以考虑使用操作列表来代替警告框。



正确地放置按钮。理想情况下，最容易点击也最不容易点错的按钮符合两个条件：它代表了用户最可能会选择的操作，即使用户一时不注意误点了它，也不会造成严重问题。尤其是：

- 如果这个按钮不会造成损害性结果，又是用户最有可能会选择的操作，那么它应该放在右边，取消按钮则应该放在左边
- 如果这个按钮会造成损害性后果，又是用户最有可能会选择的操作，那么它应该被放在左边，取消按钮应该放在右边

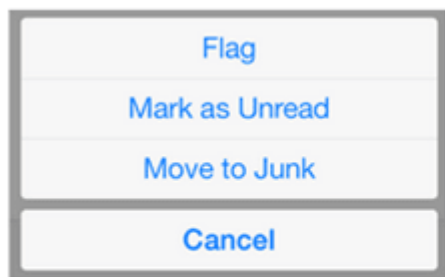
提示：一般来说，当警告框出现的时候，按Home键将会从该app里切回主屏幕，此时Home键的效果类似于取消按钮——当用户回到app中的时候，警告框将消失，操作也不会被执行。

为按钮设计简短而逻辑清晰的文案。好的按钮文案一般只有1到2个单词，描述用户点击按钮后的结果。设计文案时可以遵循以下指南：

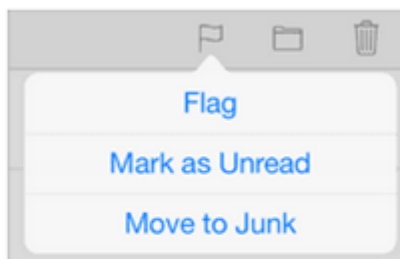
- 跟其它所有按钮一样，使用标题式大写，而且不需要标点符号
- 尽可能的使用与警告文案直接相关的动词或动词词组，如"取消(Cancel)"，"允许(Allow)"和"回复(Reply)"等。
- 当没有更好的选择的时候，可以使用"OK"。避免使用"是(Yes)"或"否(No)"。
- 避免使用"你"，"你的"，"我"，"我的"这类字眼。含有这些字眼的文案可能会指代不清，还有可能造成冒犯。

操作列表(Action Sheet)

操作列表展示了与用户触发的操作直接相关的一系列选项。



iPhone 里的操作列表会从页面底部向上浮出



iPad 里操作列表总是以弹出层的形式出现

API提示：想要了解如何在代码中定义操作列表，可以参考[Action Sheets](#)。

操作列表：

- 由用户某个操作行为触发
- 包含两个或以上的按钮

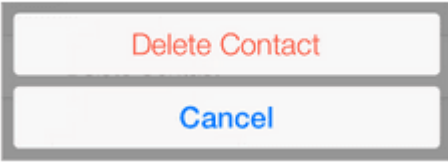
使用操作列表来：

提供完成一项任务的不同方法。操作列表提供一系列在当前情景下可以完成当前任务的操作，而这样的形式不会永久占用页面UI的空间。

在用户完成一项可能有风险的操作前获得用户的确认。操作列表让用户有机会停下来充分考虑当前操作可能导致的危险结果，并为他们提供了一些其它的选项，尤其是在以下这些情景下：

如果一个任务是从以下情景中触发的	使用这样的操作列表	是否需要添加取消按钮？
在弹出层(popover)以外	不需要过场动画，操作列表与浮出层同时出现	不需要，因为用户只要点击浮出层以外的区域，浮出层变为自动消失
在弹出层里面	需要过场动画，操作列表滑出，出现在弹出层上方	需要，此时需要提供一个方法，让用户在不关闭浮出层的情况下仍能收起操作列表

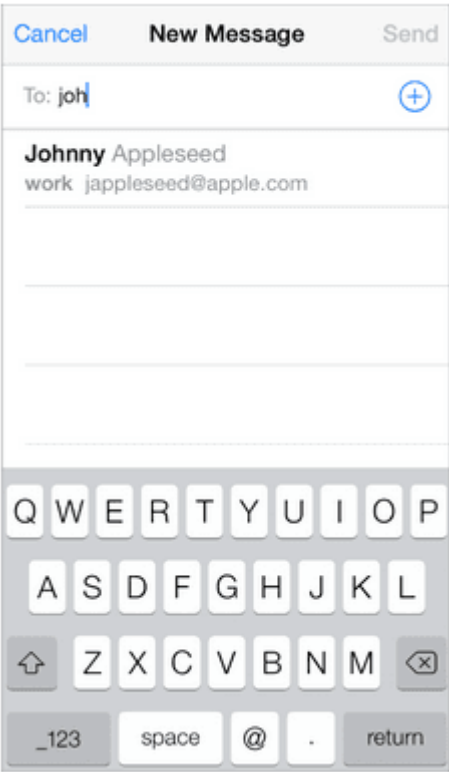
无论在哪种设备中，使用红色文字来表示可能存在破坏性的操作。在操作列表的顶部使用文字颜色为红色的按钮，因为越靠近列表顶部的操作越容易引起用户注意。在iPhone里，潜在风险的操作离列表底部越远，用户在关注Home键的时候就越不容易误点它。



避免让用户滚动操作列表。如果你的操作列表中存在过多按钮，用户必须要滚动才能看完所有操作。这样的体验是可能让用户不安，因为他们要花更多的时间来充分理解每个选项的区别。此外，用户在滚动的过程中将很有可能误点其它按钮。

模态视图(Modal View)

模态视图是一个以模态形式展现的视图，它为当前任务或当前工作流程提供独立的、自包含的(self-contained)功能。



API提示：想要了解如何在场景中定义模态视图，请参考*UIViewController Class Reference*。

模态视图：

- 占据整个屏幕，在iPad中，它也可能占据整个父视图(parent view)的区域
- 包含完成当前任务所需的文字和控件
- 通常也会包含一个完成任务的按钮（点击后即可完成任务，当前模态视图也会消失）， 和一个取消按钮（点击后即放弃当前任务，同时当前模态视图消失）

当需要用户完成与你的app中的基础功能相关的、独立的的任务的时候，可以使用模态视图。模态视图尤其适用于那些所需元素并非常驻在app主要UI中、又包含多个步骤的子任务。

在iPad上，根据当前任务的种类和你的app的整体视觉风格来选择适当的模态视图。你可以使用以下定义的任何一种模态视图样式：

模态视图样式	外观	适用情景
全屏(Full screen)	占据整个屏幕	
页列表(Page sheet)	宽度固定为 768 像素，高度与当前屏幕的高度相同；横屏状态下，模态视图左右两边的可见区域均变暗	用于承载在当前模态视图中可以完成的、较为复杂的任务
表格列表(Form sheet)	固定大小为 540×620 像素，居中显示与屏幕中。横屏状态下，当输入键盘唤起时，表格列表上移到状态栏下方	收集来自于用户的结构化信息(Structured information)
当前上下文(Current context)	与父视图的尺寸相同	在对分视图、弹出层或其它非全屏视图中展示模态内容

在iPad上，不要让模态视图覆盖在弹出层之上。除了警告框外，没有任何元素应该覆盖在弹出层上面。除非极其少有的情况下，用户在弹出层内进行的操作结果必须要以模态视图的形式展现，即便是这个时候，也请先将弹出层关闭，再出现模态视图。

在iPhone上，确保你的模态视图看起来与你的app的整体视觉风格相协调。举个例子，如果一个模态视图含有导航条和取消或完成任务的按钮，这里的导航条样式应该与你的app中导航条一样。

无论是哪种设备，合适的话，在模态视图里加入可以说明任务内容的标题。你可能还需要在模态视图里加入一些补充文字，来清楚地阐明任务内容，并提供一些任务指南。

无论是哪种设备，选择一个适当的过渡动画来展示模态视图。使用与你的app一致的过渡动画，让用户可以准确地理解当前页面内容的转变与模态视图的出现。关于这一点，你可以指定以下任意一种过渡动画：

- 垂直出现(Vertical).模态视图从底部边缘滑入屏幕，也同样从屏幕底部滑出（默认模式）。
- 弹出(Flip).当前视图从右往左水平滑动，露出模态视图。从视觉上看，模态视图好像原来就处于当前视图的下面，当前视图移开时，它便出现了。离开模态视图时，原先的父视图从左边滑回屏幕右边。

如果你要改变当前的过渡动画样式，请确保这种改变对于用户而言是有用而且有意义的。用户很容易便能感知到这些改变，还会认为这些改变存在特别的意义。最好能设计出一种符合逻辑并始终保持一致的过渡方式，让用户容易感知并且记忆。在没有充分理由支持的情况下，最好不要改变这些默认的过渡方式。