IOS 10人机界面设计指南

1.1 设计原则（Design Principles）

作为一名移动应用设计师，意味你有机会发布一款具有杀伤力的产品能够登上苹果商店的榜首。为了实现这个目标，你的产品必须在质量和功能上达到高标准。

以下三大原则让iOS系统有别于其它平台：

清晰（**Clarity**）：

纵观整个系统，任何尺寸的文字都清晰易读，图标精确易懂，恰当且微妙的修饰，聚焦于功能，一切设计由功能而驱动。留白、颜色、字体、图形以及其它界面元素能够巧妙地突出重点内容并且表达可交互性。

遵从（**Deference**）：

流畅的动效和清爽美观的界面有助于用户理解内容并与之交互，而不会干扰用户。当前内容占据整屏时，半透明和模糊处理能够暗示其它更多的内容。减少使用边框、渐变和阴影让界面尽可能地轻量化，从而突出内容。

深度（**Depth**）：

清楚的视觉层和生动的动效表达了层次结构，赋予了活力，并有助于理解。

易于发现的且可触发的界面元素能提升体验愉悦感，让用户在成功触发相应功能或者获得更多内容的同时还能掌控当前位置的来龙去脉。当用户浏览内容时，流畅的过渡提供一种纵深感。

要想扩大影响力和涉及范围，在设计你的独特应用时，请熟记以下几点原则：

美学完整性（**Aesthetic Integrity**）

美学完整性代表了一款应用的视觉表象和交互行为与其功能整合的优良程度。例如，一款协助用户完成重要任务的应用应该使用不易察觉且不引人注目的图形、标准化控件和可预知的交互行为从而让用户保持专注。反之而言，一款沉浸式体验的应用（比如游戏），就需要吸引人的视觉表象，在鼓励用户探索的同时带来无穷的乐趣和刺激。

一致性（**Consistency**）

一款内部一致的应用能够贯彻相同的标准和规范：使用系统提供的界面元素、风格统一（众所周知）的图标、标准的字体样式和一致的措辞。应用所包含的特征和交互行为是符合用户心理预期的。

直接操作（**Direct Manipulation**）

对屏幕上的对象直接操作（而不是通过一堆控件）能够提升用户的参与度并有助于理解。直接操作指包括用户旋转设备或者使用手势控制屏幕上的对象。通过直接操作，他们的交互行为能够得到即时可视的反馈。

反馈（**Feedback**）

反馈认证交互行为，呈现结果，并通知用户。系统自带的iOS应用对每一个用户行为都提供了明确的反馈。可交互的元素被点击时会被临时高亮，进度指示器（progress indicator）显示了需要长时间运转的操作的进度，动效和声音加强了对行为结果的提示。

隐喻（**Metaphors**）

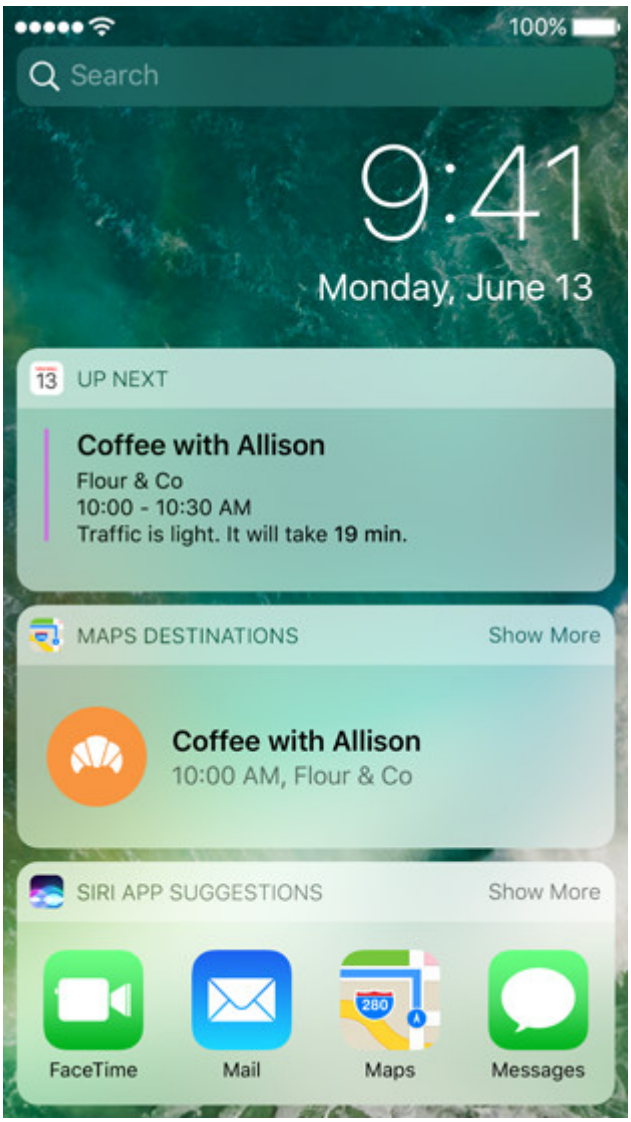
当一个应用的虚拟对象和行为与用户熟悉的体验相似时——无论这种体验是来源于现实生活或是数字世界，用户就能更快速地学会使这款应用。隐喻在iOS中能够起作用是因为用户与屏幕进行物理上的交互。他们通过将视图移出屏幕来显示下方的内容，他们拖曳（drag）和滑动（swipe）对象，他们拨动（toggle）开关，移动（move）滑块，滚动（scroll）数值选择器，他们甚至通过轻扫（flick）来翻阅书籍和杂志。

用户控制（**User control**）

在iOS内部，是用户——而不是应用——在控制。应用可以对一系列用户行为提供建议，或对可能造成严重后果的行为发出警告，但不应该替用户做决定。好的应用会在让用户主导和避免不想要的结果中找到平衡。为了让用户感觉到是他们在控制，应用应该使用熟悉且可预知的交互元素，让用户二次确定有破坏性的行为，并且让即使在运行中的操作也能够被轻易取消。

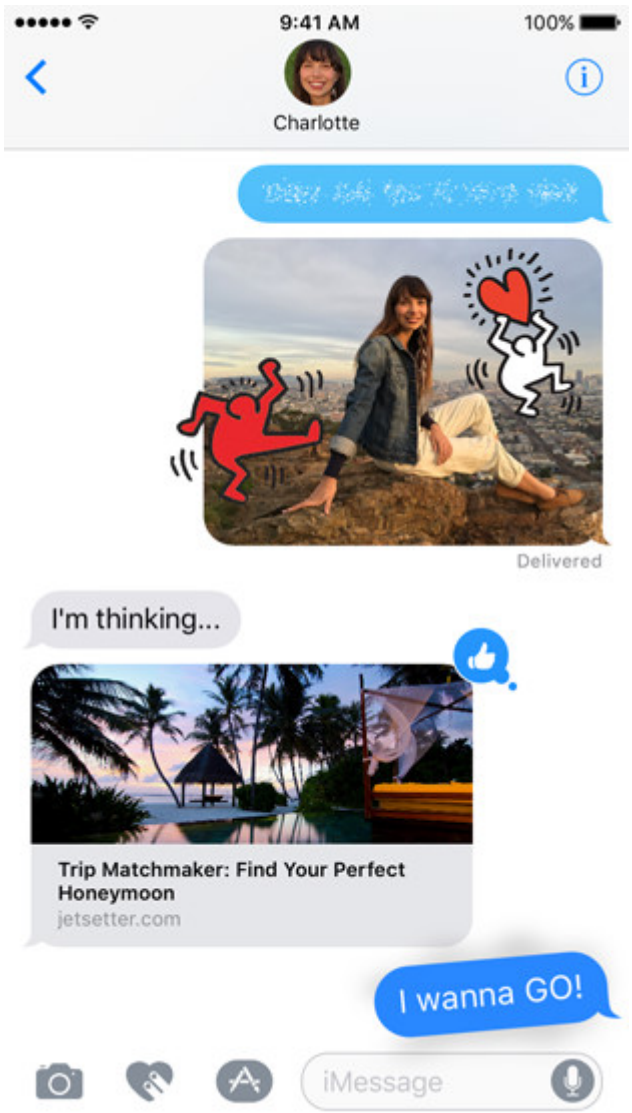
1.2 iOS 10 新特征介绍

在iOS 10，你可以创造前所未有的更强大的应用。当你浏览这些新改变并在思考他们将如何帮助你的应用时，请特别关注设计指南。



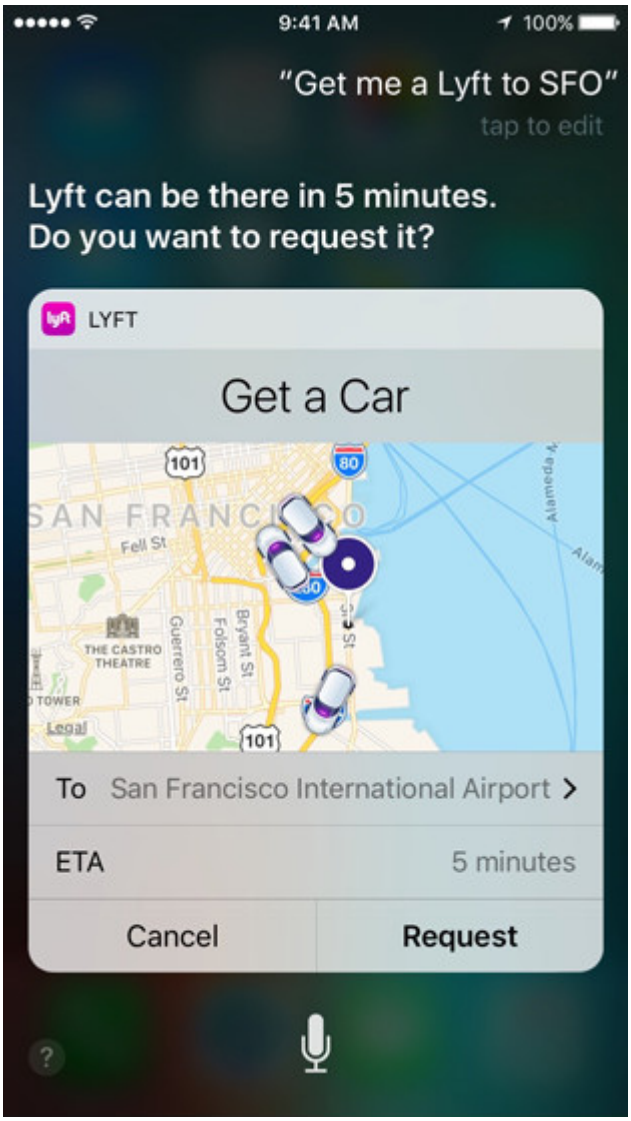
搜索屏幕和主屏幕的窗口控件（**widget**）

Widget 提供即时且有用的信息，或是应用特有的一些无需打开应有就能使用的功能。在过去，用户在消息中心添加widgets当作快速入口。现在，用户在搜索屏幕添加widgets，用户可通过在主屏或是锁屏右滑进入搜索屏幕。你也可以在主屏通过3D Touch触发的某个应用的快捷操作菜单（quick action list）上方添加widget。Widget的设计和交互方式也改变了。请注意更新你现有的设计。



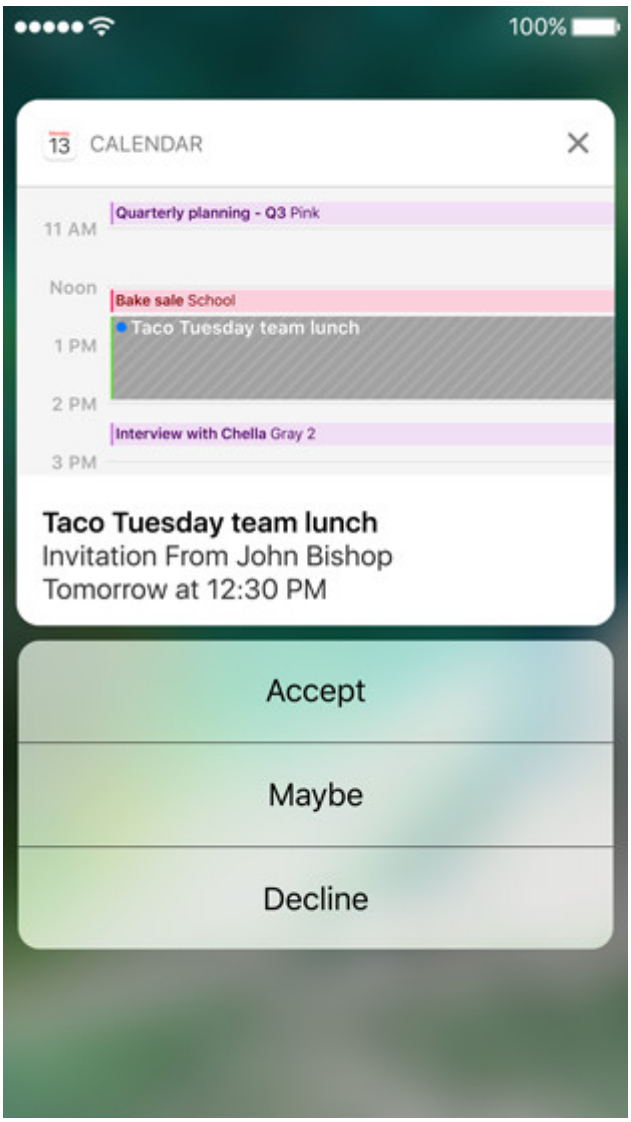
与**Messages**的联动

应用可以通过执行一种出现在对话下方的messaging插件让用户与朋友分享来自该应用的内容。应用可以通过Message分享文字，图片，视频，贴纸，甚至可交互的内容，譬如信息内的游戏。



与**Siri**的联动

应用能够与Siri联动，从而让用户使用声音来执行相应的应用操作，譬如打电话、发短信和开始锻炼。



可拓展的通知栏

你可以通过拓展详情视图来强化你的通知栏，用户可以在未锁屏状态下通过3D Touch功能点击或是下滑你的通知来打开拓展视图。使用这种视图能够让用户快速浏览更多信息，并允许他们在不离开当前界面的情况下对该消息进行快速操作。

1.3 界面基本元素

大多数的iOS应用使用了来自UIKit的部件，这是一个定义了基本界面元素的编程框架。这个框架让各种应用在视觉上达到一致的同时还提供了高度的个性化。UIKit元素是灵活且常见的。它们是可适配的，让你能够设计一个在任何iOS设备上都看起来很棒的应用，而且能够在系统发布新版本的时候自动更新。由UIKit提供的界面元素可以分为以下三种：

栏：

告知用户现在在应用的哪里，提供导航，而且还可能包含按钮或者其它用来触发功能和交流信息的元素。

视图：

包含用户在应用内最关注的信息，例如文本、图形、动画和交互元素。视图允例如滚动、插入、删除和排列之类的行为。

控件：

触发功能和传递信息。控件包括按钮、开关、输入框和进度指示器。

为了进一步定义iOS界面，UIKit规定了你的应用能够采用的功能。通过这个框架，你的应用可以对触摸屏上的手势作出应答，还可以包含一些例如绘画、辅助和打印的功能。

iOS也和其他编程框架和技术紧密结合，譬如Apple Pay、HealthKit 和ResearcKit, 它帮助你设计出一个强大地惊人的应用。

2. 交互（Interaction）

> 2.1 3D 触摸（3D Touch）
>
> - 2.1.1 主屏幕交互（Home Screen Interaction）
> - 2.1.2 轻压（Peek）和重压（Pop）
> - 2.1.3 Live Photos
>
> 2.2 辅助功能（Accessibility）
>
> 2.3 音频（Audio）
>
> 2.4 身份验证（Authentication）
>
> 2.5 数据输入（Data Entry）
>
> 2.6 反馈（Feedback）
>
> 2.7 文件处理（File Handling）
>
> 2.8 启动初体验（First Launch Experience）
>
> 2.9 手势（Gestures）
>
> 2.10 加载（Loading）
>
> 2.11 模态（Modality）
>
> 2.12 导航（Navigation）
>
> 2.13 请求许可（Requesting Permission）
>
> 2.14 设置（Settings）
>
> 2.15 用辞（Terminology）
>
> 2.16 撤销和重做（Undo and Redo）

2.1 3D 触摸（3D Touch）

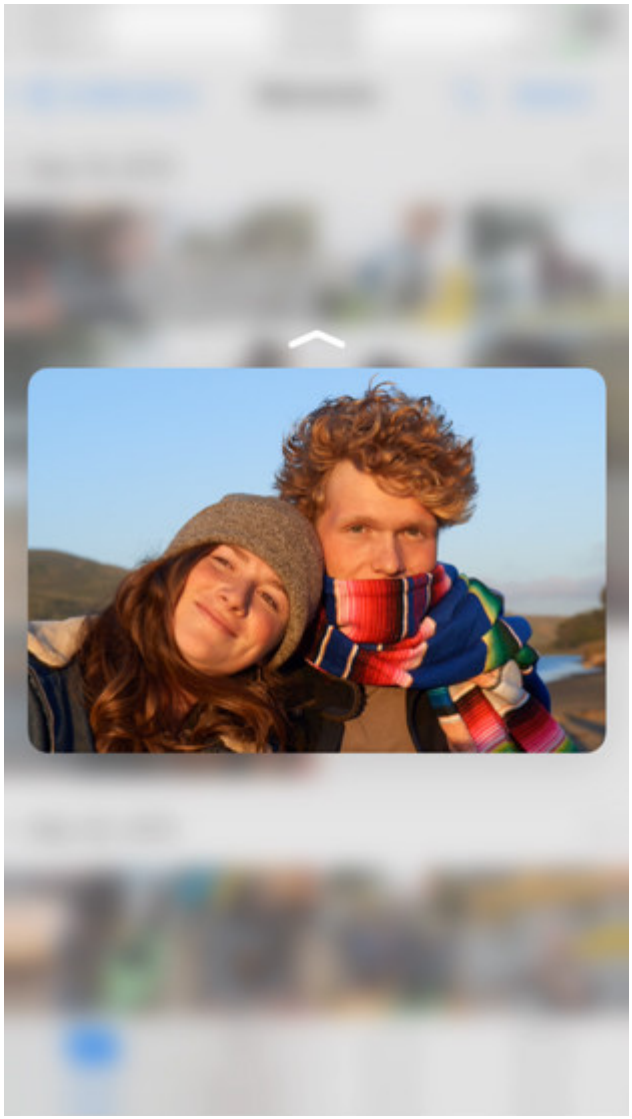
3D Touch 为触碰式交互增加了一个维度。在支持3D Touch 的设备上，用户通过对触摸屏施加不同的力度来实现更多的功能，譬如触发菜单、显示更多的内容或是播放动画。用户无需学习新的手势来使用3D Touch。当他们轻压屏幕并且获得应答的时候就能立即发现这一新的交互维度。

2.1.1 主屏幕交互（Home Screen Interaction）

在支持3D Touch的设备的主屏按压应用图标会触发相应的操作视图。该视图让你能够快速地执行常用的应用任务和预览有趣的信息，譬如日历应用，它能够提供创建新事件的快捷操作，同时显示日程表上的下一个事件。了解相关设计指导，请参阅Home Screen Action和Widgets。

2.1.2 轻压（Peek）和重压（Pop）

轻压允许用户使用3D Touch在当前环境上预览一个临时视图内的对象，譬如一个页面、链接或者文件。要想在支持该功能的设备上实现预览，只需用手指对应用施加一点压力，而抬起手指就能退出预览。要想打开对象来浏览更多的内容，请更重地按压屏幕直到对象放大到填满屏幕。在一些轻压视图上，你可以通过上滑来显示相应的操作按钮。譬如，在Safari打开了某个链接的轻压视图时，你可以通过上滑展开相应的操作按钮——打开链接，添加至阅读列表和复制链接。



利用轻压视图提供实时的，内容丰富的预览

理想情况下，轻压视图为该项提供足够的信息以补充说明当前任务，或者帮助你决定是否完全地打开该项。例如，预览邮件（Mail）信息中的链接，从而决定是否在Safari浏览器中打开或者分享给朋友。轻压视图一般被利用于表单视图中,提供一个行项的详细信息，从而决定是否选择该项。

设计足够大的轻压视图

设计一个足够大的轻压视图从而保证手指不会遮挡到内容。确保轻压视图能够提供足够详细的信息，以便用户决定是否按地更重来完全地打开该项。

统一使用轻压和重压功能

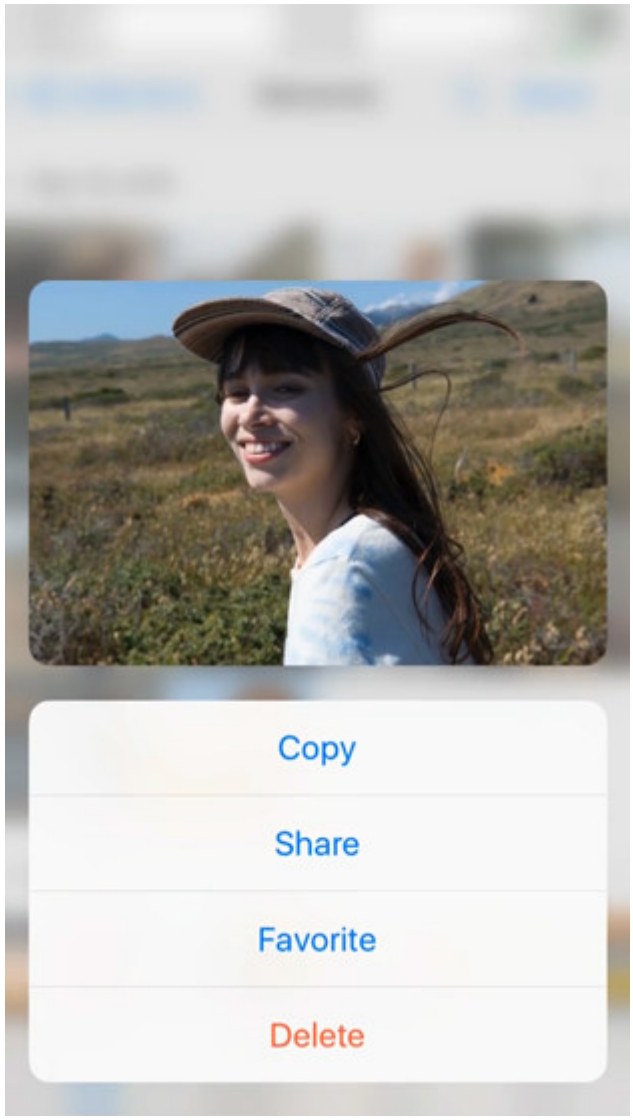
如果你只在某些地方使用轻压和重压，而不在另一些地方使用，用户就不会知道到底哪里可以使用这个功能，而且可能会认为你的应用或是他们的设备出了问题。

允许每个轻压视图都能够被重压

虽然轻按视图能够提供给用户他们所需的大部分信息，但如果他们想离开当前任务并转移注意力至该项时，应该允许他们过渡到重压。

避免在轻压视图中呈现按钮式元素

如果用户抬起手指去点击类似按钮的元素，轻压就会消失。



不要让同一项具备轻压和编辑菜单（**Edit menu**）两个功能

当一个项目同时启用两个功能时，不但会让用户感到困惑，也会让系统难以判断用户目的。了解更多指导，请参阅Edit Menus。

适当时提供操作按钮

不是每个一轻压都需要操作按钮，但这是一个为常用任务提供快捷操作的好方式。如果你的应用已经为项目提供了自定义的点击并长按（touch-and-hold）动作，那么最好在轻压里包含同样的操作。

避免为打开被轻压的项目提供操作按钮

用户一般都通过更重的按压来打开他们轻压的项目。所以，没有必要再提供一个明显的打开按钮。

不要让轻压成为唯一的执行项目操作的操作

并不是所有设备都支持轻压和重压，甚至有的用户会关闭3D触摸功能。你的应用为这些情况考虑其它触发项目操作的方式。譬如，你的应用可以将轻压的快捷操作映射到一个视图中，该视图会在点击和长按时出现。

2.1.3 Live Photos

应用可以通过支持Live Photos，并在照片中加入压感用来查看动态回忆。当你按压它们时，Live Photos死而复生，通过动作和声音再现拍照的前后时刻。了解相关设计指导，请参阅Live Photos。

2.2 辅助功能（Accessibility）

iOS 提供了大量的辅助功能来帮助失明、失聪以及其他残疾群体。大部分以UIKit为基础的应用能够轻易地具有辅助性，让更多的用户来使用你的应用，因为你为所大众提供了平等的使用体验。

为图片、图标和界面元素提供可选择的文字标签

可选择的文字标签在屏幕上是不可见的，但是他们让VoiceOver能够通过声音描述屏幕上有什么，让失明用户能够轻易地使用导航。

相应辅助功能的偏好设置

如果你的应用使用UIKit来实现用户界面，文字、界面元素就会自动调整至相应辅助功能的偏好设置，譬如加粗并且更大的文字。你的应用也应当在适当的时候检查并相应辅助功能的偏好设置，譬如当减弱动态效果（reduce motion）的开关被打开时。采用自定义字体的应用应该力图和系统字体的辅助特性保持一致。

测试应用的辅助功能

除了文字和动态效果的变化，辅助功能选项还能改变对比度，反转颜色，降低透明度以及更多。为那些需要这些功能的用户启用设置并观察你的应用将会变成什么样并且如何运作。

包含隐藏式字幕和口述影像

隐藏式字母帮助失聪以及重听用户明白视频中的对话和其它音频内容。口述影像为视觉受损的用户提供了关键视频内容的口头解说。

了解更多信息，请查阅iOS Accessibilty和Accessibility Programming Guide for iOS。

2.3 音频（Audio）

无论声音是你应用体验的要素或只是一个点缀，你都应该知道用户对声音有什么要求并且满足他们的期待。

用户通过音量键、静音键、耳机声控和屏幕上的音量调节滑块控制声音。非常多的第三方配件也包含声控功能。音频可以通过内部和外部的扬声器、耳机输出，甚至通过支持AirPlay或是蓝牙设备无线输出。

静音：用户将他们的设备调节至静音来避免被意外的声音（比如电话铃声和短信提示声）打扰。他们也想要关闭没有意义的声音，包括按键声、音效、游戏配乐以及其它音频反馈。当设备被设置成静音，只能出现被明确被打开的声音，比如媒体播放中的声音、闹铃和音频/视频信息。

音量：无论是使用物理的设备按键或是屏幕上的滑块，用户都希望系统的所有音量都能够被改变，包括音乐声和应用内的音效。但是铃声音量是唯一例外，它只能在没有任何声音播放的情况下被单独调节。

耳机：用户使用耳机来私密地听声音并且能够释放他们的双手。当用户插入耳机时，他们希望声音能够自动继续播放而不被打断。当拔掉耳机时，他们希望播放能够立即停止。

必要时自动调节不同层级的声音，但不是整体音量

为了达到更好的混合音效，你的应用可以单独调节不同层级音频间的相对音量。但是，最终的音量输出应该由系统音量决定。

恰当的时候允许音频重选路由（**rerouting**）

用户会经常想要选择一个不同的音频输出设备。比如，他们会想要通过客厅的立体音响、车载收音机或是苹果电视来听音乐。请支持这个功能除非你有令人信服的理由不这么做。

使用系统提供的音量视图来调节音量

音量视图（volume view）是最好的能提供调节音量的界面控件。这个视图是自定义的，包含一个音量调节滑块，甚至包含一个用来替音频输出重选路由的控件。了解实现方法，请参阅MPVolumeView Class Reference。

短音和振动请使用系统声音服务

了解实现方法，请参阅System Sound Services Reference。

如果声音对你的应用十分重要请设置音频类别

不同的音频类别允许声音被静音按钮静音、与其它声音混响、或是当你的应用在后台时播放。根据类别的含义和当前设备的音频播放情况来选择一个类别，然后将其分配给你音频对话（audio sessions）。比如，非必要情况下，请不要打断用户正在收听的来自其它应用的音乐。总的来说，尽量不要在你的应用运行时更改所属的音频类别，除非应用需要经常地录制然后播放音频。了解实现方法，请参阅Audio Session Programming Guide。

在适当时候继续播放被干扰打断的音频

正在播放的音频有时会受来自其它应用的声音干扰。暂时性干扰（比如来电铃声）被认为是可恢复的。永久性干扰（比如被Siri打开的播放列表）被视为不可恢复的。当一个可恢复的干扰出现时，你的应用应该在干扰结束时恢复音频播放（假设音频在干扰出现之前就已经开始播放了）。比如，一个在播放配乐的游戏和一个在播放音频的媒体应用都应该恢复声音的播放。当干扰发生时应用没有在播放任何音频，那么它也就不需要恢复任何对象。

类别	含义	特性
Solo ambient	声音不是必要的，但是会使其它声音静音，例如有配乐的游戏。	受静音键控制。 不要与其它音乐混响。 不会在后台播放。
Ambient	声音不是必要的，而且不会使其它声音静音。比如，一个游戏能够允许用户在游戏时播放来自其它应用的音乐，从而替代游戏本身的配乐。	受静音键控制。 与其它声音混响。 不会在后台播放。
Playback	声音是必要的而且可能会与其它声音混响。比如，有声读物或者外国语言的教育应用，用户即使离开应用也希望能听到音频。	不受静音键控制。 可能也可能不与其它声音混响。 会在后台播放。
Record	声音被录制。比如，一个做笔记的应用提供录音模式。这类型的应用可能会想要改变它的音频类别至playback，从而允许用户播放录制的笔记。	不受静音键控制。 不与其它声音混响。 会在后台录制。
Play and record	声音被录制而且被播放，而且可能是同时的。比如，一个语音讯息或者视频对话应用。	不受静音键控制。 可能也可能不与其它声音混响。 会在后台播放和录制。

让其它应用知道何时你的应用将停止播放暂时性的音频

如果你的应用可能会暂时性地干扰到其它应用的音频，那么就应该恰当地标明声音片段，从而让其它应用知道确切的恢复时间。了解实现方法，请参阅AVFoundation Framework Reference 中的AVAudioSessionSetActiveOptionNotifyOthersOnDeactivation。

只有在有意义时才对声音控件作出反应

无论你的应用在前台还是后台，用户都能够通过应用界面以外的东西控制音频的播放，比如在控制中心（Control Center）中，或者耳机声控。如果你的应用正在一个明确与声音相关的环境下播放音频，或是连接到一个支持AirPlay的设备上，那么对声音控件作出反应是合理的。但是，你的应用不应该混淆其它应用的音频，因为它们可能会在控件被激活时播放。

不要重新定义声音控件

用户希望声音控制在任何应用都保持一致性。永远不要重新定义声音控件。如果你的应用不支持某些控件，那么只需不对它们作出反应即可。

2.4 身份验证（Authentication）

要求用户进行身份验证时应该用有价值的东西交换，比如个人化体验、获得更多功能、购买内容或者同步数据。如果你的应用要求身份验证，请保证登陆流程快速简单并且低调，这样就不会减少应用的乐趣。

尽可能地延后登陆

用户经常遗弃应用因为他们在做一些有用的事前被强制登陆。在强制用户前给他们一个爱上你的应用的机会。在购物应用内，允许用户启动应用后能马上浏览你的商品，然后在他们决定购买时才要求登陆。在流媒体应用内，允许用户先探索和了解你能够提供的内容，然后在他们播放时让他们登陆。

解释身份认证的优势以及如何注册

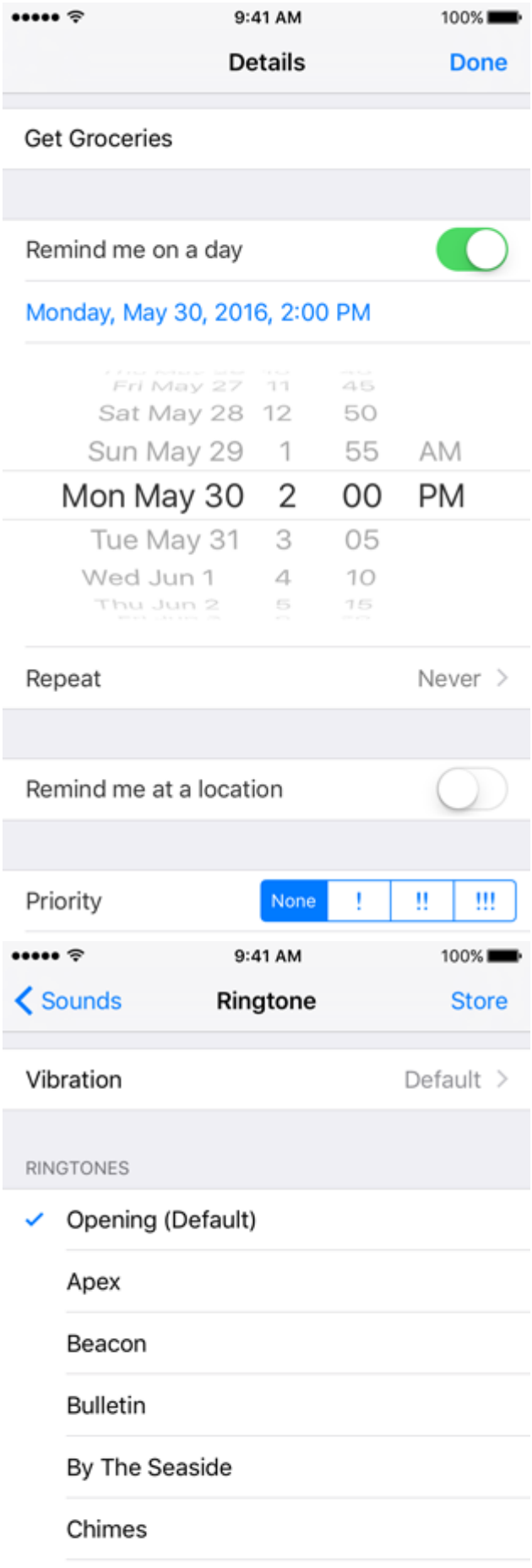
如果你的应用要求身份认证，在登陆界面简要友好地介绍之所以要登陆的原因及其优势。并且请牢记不是每个人在开始使用应用时都拥有一个账号。请确认你解释了如何得到账号，或者提供一个简单的应用内的注册方式。

展示适合的键盘来减少数据输入

比如，当要求填写一个邮箱地址时，请展示包含信息输入所需快捷键的邮件键盘窗口。

2.5 数据输入（Data Entry）

无论是点击界面元素还是使用键盘，信息输入都是一个冗长的流程。当一个应用在做一些有用的事情前要求用户一连串的输入，进而拖慢了流程，那么用户会很快感到失望，甚至会彻底地抛弃这个应用。



可能时展示选项

尽可能地提高信息输入的效率。比如，考虑使用选择器或是列表来替代输入栏，因为从一列提前设定好的选项中选择一个比打字容易。

可能时从系统中获取信息

不要强迫用户提供那些可以自动或是在用户许可内就能获取的信息，比如联系人或是日历信息。

提供可靠的默认值

尽可能地预填最可能的信息值。提供一个可靠的默认值缩短了做决定的时间从而加快了流程。

只有在收集完必需信息之后才能进行下一步

在允许“下一步”或“继续”按钮前，确保所有必要的输入框都有信息。尽可能地在用户输入之后就立马检查输入值，这样他们就能立即改正。

只要求必要的信息

只有系统运行真正必需的信息才使用必填栏。

简化值列表的导航

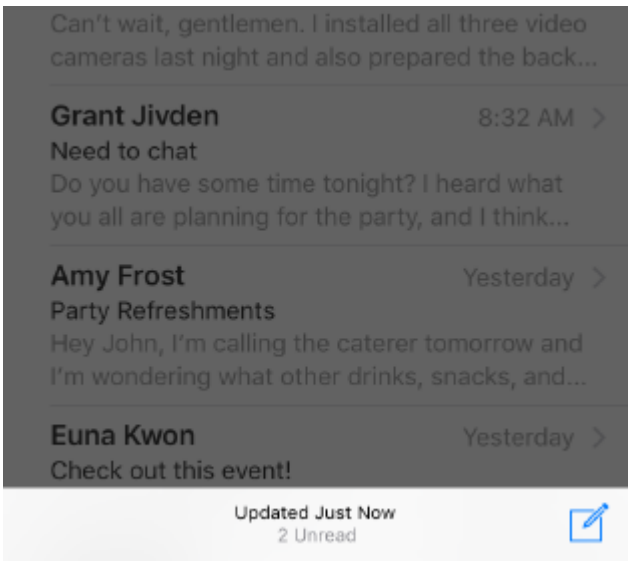
尤其是在列表和选择器中，必需能够简单地选择值。考虑通过将值列表按首字母排序或是其它逻辑排列，从而加快浏览和选择的速度。

在输入栏显示提示以辅助说明

当输入栏没有其它文字时，可以包含占位符文字——比如“邮件”或“密码”。当占位符文字已经足够说明时不要再单独使用标签来描述。

2.6 反馈（Feedback）

反馈让用户知道应用现在在做什么，发现下一步他们应该做什么，并且理解操作的结果。



悄悄地在你的界面中加入状态或其它类型的反馈

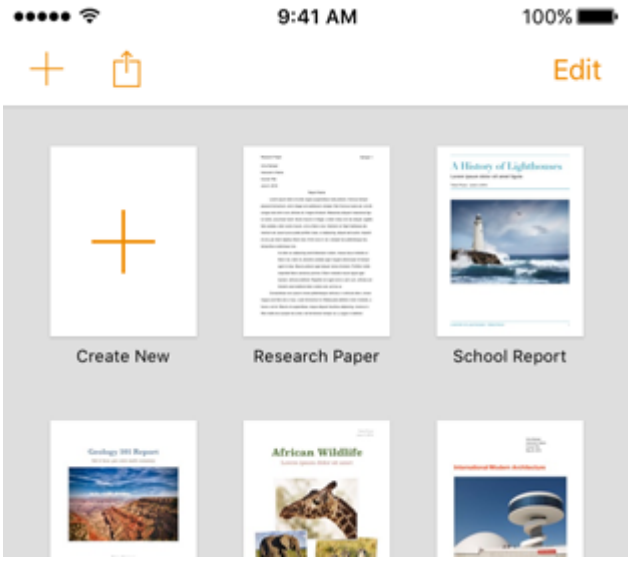
理想中，用户能够在不采取任何操作或是被打扰的情况下得到重要的信息。比如，当用户在邮件应用中查看邮时，状态信息被巧妙显示在工具栏上。这个信息不会和屏幕上的主要内容抢风头，但是用户在任何时候快速一瞥就能查看。

避免不必要的警告

警告是一种有威力的反馈机制，所以它应该只被用于传递重要的并且最好是需要操作的信息。如果用户看到太多包含无关紧要信息的警告框，他们很快就会学会忽略之后的警告。了解更多帮助，请参阅Alerts。

2.7 文件处理（File Handling）

用户在创建、查看和操作文件时无需思考文件系统。如果你的应用需要运行文件时，尽可能地淡化文件处理。



让用户相信**除非主动取消或删除**

文件会随时被保存。总而言之，不要让用户去即时保存文件。反之，在文件被打开、关闭，或是跳转至其它应用时，应该自动定时地替用户保存文件。但在某些情况，比如正在编辑一个已被创建的文件时，保存和取消的选项也是有意义的，因为它们帮助确认何时编辑的内容应该被保存。

不要提供创建本地文件的选项

用户总是希望他们全部的文件都能在任何设备上读取。如果可能，你的应用应该支持文件云储存，比如通过与iCloud类似的服务。

设计一个直观并且图像化的文件浏览界面

理想情况下，使用用户熟悉的系统文档选择器来浏览文件。如果你想设计一个自定义的文件浏览器，请确保它是直观且高效的。最好的文件浏览器应该是高度图像化的，提供了文档的视觉再现。要想加快导航速度，减少手势的使用，并且考虑提供一个添加新文件的按钮，这样用户就无需再到其它地方去创建新文档。

让用户在你的应用内就能预览文件

你可以使用Quick Look 功能让用户查看来自Keynote、Numbers和Pages的内容，以及PDF文档、图片以及某些其它格式的文件，即使你的应用并没有真正打开它们。请参阅Quick Look。

合适时，与其它应用共享文件

如果有意义，你的应用可以通过document provider extension与其它应用共享文件。你的应用也可以让用户浏览和打开来自其它应用的文件。了解实现方法，请查阅Document Picker Programming Guide。

2.8 启动初体验（First Launch Experience）

应用的启动时间是你接触新用户并与老用户再次连接的第一个时机。请设计一个快速、有趣并有教育意义的启动体验。

提供启动画面

启动画面在应用打开时出现，在加载应用初始内容的同时，让人感觉你的应用的响应速度很快。因为这个画面很快就会被应用的首屏替代，所以它应该尽量与首屏相似，除非出现可定位的文字和可交互的元素。了解更多，请参阅Launch Screen。

选择合适的方向启动

如果你的应用同时支持竖屏和横屏模式，那么应该以设备目前的方向启动。如果你的应用只在一个方向运行，那它只能在相同方向启动并在需要时允许用户旋转设备。除非有迫不得已的原因，否则处于横屏模式的应用正确地选择方向，无论Home键是在左侧还是右侧。了解更多信息，请参阅Layout。

快速使用。避免出现延迟用户使用应用时间的启动画面、菜单和说明。反之，允许用户快速进入应用内。如果你的应用需要教学或是介绍步骤，为用户提供一个跳过的选项并且不要对老用户展示这些。

提前设想用户可能会需要的帮助

经常主动地考虑用户何时会遇到麻烦。比如，一个游戏，能够在暂停或是角色很难升级时提供一些诀窍。当用户错过启动画面的内容时，允许他们之后重新观看教程。

只在教程中展示最关键的内容

虽然为新用户提供引导没错，但是教学不能成为优秀的应用设计的代替品。更重要的是，确保你的应用是直观的。如果你的应用需要过多的引导，那么请重新审视你的设计。

让学习变得有趣而且易于学习

通过操作来学习比阅读一长串说明来的更有趣和有效。 在上下文环境中，通过动画和可交互性循序渐进地教导。避免展示看起来似乎可交互的屏幕截图。

避免在最开始要求用户设置信息

用户期待应用马上工作。为大多数人设计你的应用，然后让余下少部分需要不同配置的人自己调整参数来满足他们的需求。尽可能地，从设备设置和默认中或许设置信息，或者通过同步服务，比如iCloud。如果应用一定要求设置信息，那么在最初在应用内提示用户，然后允许用户稍后在应用设置中修改。

避免展示应用内的接受许可协议和免责声明

在你的应用被下载之前直接在苹果商店展示接受许可协议和免责声明。如果你必须将这些东西放在你的应用里，那么以和谐融入它们，以避免干扰用户体验。

在你的应用重新启动时恢复之前的状态

不要让用户重新操作来回到之前的应用定位。保存并且复原应用的状态，这样用户就能从他们上次离开的位置继续。

不要太快或是太频繁地要求用户对你的应用评分

太快或是太频繁地要求评分会让用户恼怒，并且减少最终收到的有用反馈的数量。为了鼓励考虑周到的反馈，在要求评分之前，给用户足够的时间直到他们形成对应用的看法。总是提供跳出评分提示的选项，并且永远都不要强迫用户对你的应用评分。

不要鼓励重启

重新启动耗费时间并且让你的应用看起来即不可靠又不可用。如果你的应用出现储存或者其它问题，导致它无法运行只能系统重启，那么你应该解决这些问题。

2.9 手势（Gestures）

用户通过在触摸屏上使用手势来与iOS设备交互。这些手势表现了一种亲密的人与内容之间的联系，并且加强了对屏幕上对象直接的操作感。用户普遍地希望一下的标准手势能够在操作系统和每一个应用内保持一致。

点击（Tap）：激活一个控件或者选择一个对象。

拖曳（Drag）：让一个元素从一边移动到另一边，或者在屏幕内拖动元素。

滑动（Flick）：快速滚动或是平移

横扫（Swipe）：单指以返回上一页，呼出分屏视图控制器（split view controller）中的隐藏视图，滑出列表行中的删除按钮，或在轻压中呼出操作列表。在iPad中四指操作用来在应用间切换。

双击（Double tap）：放大并居中内容或图片，或者缩小已放大过的。

捏合（Pinch）：向外张开时放大，向内捏合时缩小。

长按（Touch and hold）：在可编辑或者可选文本中操作，显示放大视图用以光标定位。在某些与集合视图类似的视图中操作，进入对象可编辑的状态。

摇晃（Shake）：撤销或重做

一般使用标准手势

用户已熟悉了标准手势，并不喜欢在做相同事情时被强迫去学习不同的方式。在游戏等沉浸式体验的应用中，自定义的手势能够成为体验的有趣要素。但是在其它应用中，最好使用标准手势，这样用户就无需花费多余的力气去学习和记忆它们。

不要禁止系统性的手势

除了标准手势，还有一些手势会触发系统性的操作，譬如呼出控制中心或是通知中心。在每个应用中，用户都依赖使用这些手势。

避免使用标准手势来执行非标准的操作

除非你的应用时一个极具可玩性的游戏，否则重新定义标准手势会变得混论和复杂。

为基于界面的导航和操作提供补充性的快捷手势，而不是取而代之

可能时，提供简单明显的方式来导航或是执行操作，即使它可能意味着额外的点击。非常多的系统应用包含一个提供了清晰可点的返回上一页的按钮的导航栏。但是用户也能通过在屏幕边缘右滑来返回。在iPad，用户能够点击Home键退出到主屏幕，或是使用四指捏合的手势。

使用多指手势来加强某些应用的体验

虽然涉及多个手指同时操作的手势不适用于每一个应用，但是他们能够丰富一些应用的体验，譬如游戏和绘画应用。比如，一个游戏可能包含多种屏幕上的控件，比如同时操作的的控制杆和发射键。

2.10 加载（Loading）

当内容在加载时，一片空白静止的屏幕好像应用被冻住了，让人感到困惑和失望，而且很可能让用户离开你的应用。

明确加载的状态

至少，展示一个活动旋转器（activity spinner）来表明有任务在进行中。更胜一筹的是，显示明确的进度，这样用户就能知道他们还需等待多久。

通过教育或娱乐用户来填充**加载的时间**

尝试展示游戏诀窍、令人愉悦的视频序列或者有趣的占位图。

自定义加载画面

尽管标准的活动指示器还不错，但他们有时会感觉是脱离上下文环境的。尝试设计符合你的应用或游戏的自定义动画和元素，以实现一个更沉浸式的体验。

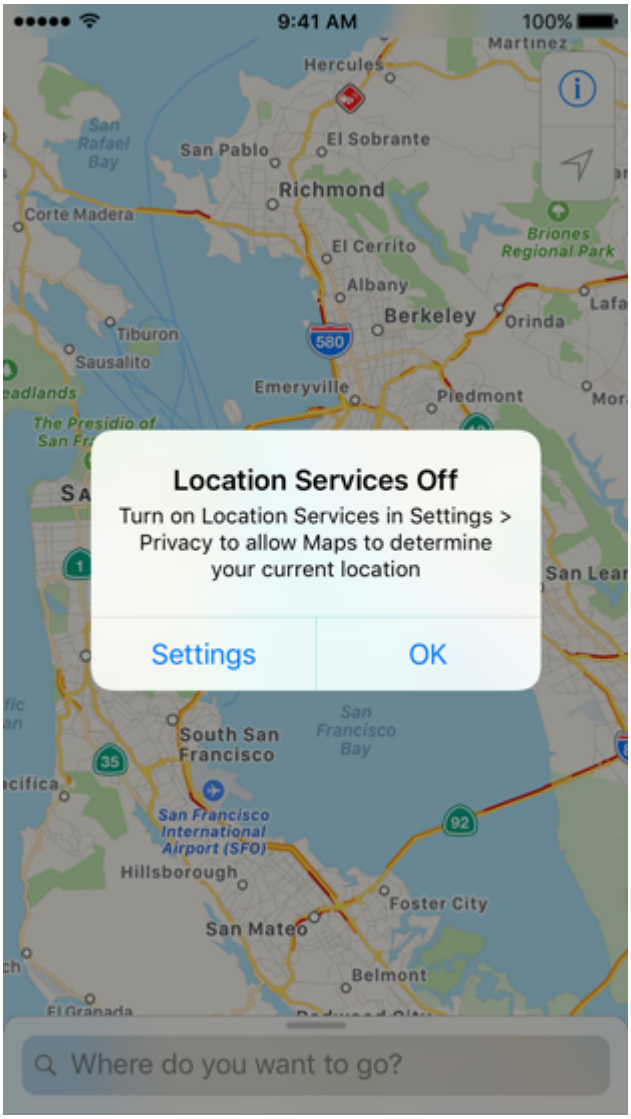
尽快显示内容

不要让用户在看到屏幕画面前去等待内容的加载。立马显示屏幕画面，然后通过占位符、图片或者动画明确告知用户哪个范围的内容还未显示。当内容加载成功之后再把占位元素替代掉。可能时，比如当动画在播放时或是用户在某个层级或菜单导航时，在后台预加载接下来要出现的内容。

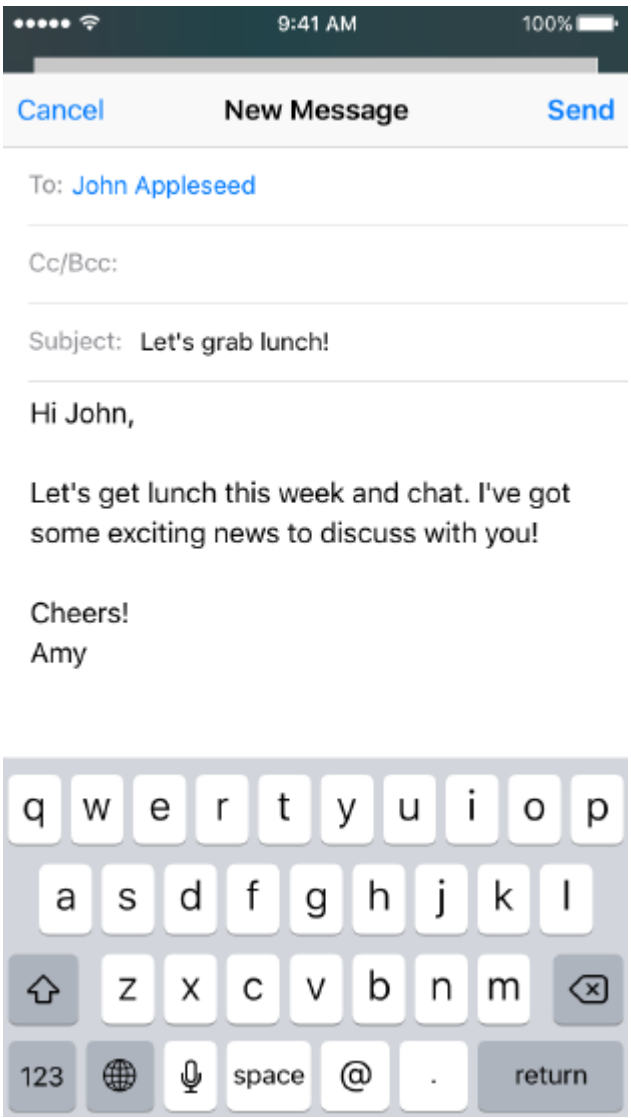
了解更多指导，请参阅Progress Indicators。

2.11 模态（Modality）

模态突出焦点，因为用户只有在完成当前的任务或关闭一个信息或视图之后才能去做其它事情。操作列表、警告框和活动视图都提供了模态化的体验。当屏幕上出现一个模态视图时，用户必须采取一个决定（点击按钮或是其它）才能退出模态化体验。在日历（Calendar）中编辑事件或是在Safari浏览器中选择书签都是模态视图在应用中被采用的例子。一个模态视图可以占据整个屏幕、整个父视图（比如浮出层）或者屏幕的一部分。一个模态视图一般都含有“完成”和“取消”按钮来退出视图。


9:41 AM
100%
Martinez
Hercules
San Rafael Bay
San Pablo
El Sobrante
Richmond
Corte Madera
Briones Regional Park
El Cerrito
Albany
Berkeley
Orinda
Tiburon
Sausalito
580
Emeryville
Piedmont
Location Services Off
Turn on Location Services in Settings > Privacy to allow Maps to determine your current location
Settings
OK
South San Francisco
San Francisco Bay
35
92
San Francisco International Airport (SFO)
Hillsborough
Foster City
San Mateo
Belmont
Where do you want to go?

Δ 警告框



△ 模态视图

减少模态的使用

一般来说，用户更喜欢与应用进行非线性的交互。只在必须要引起用户注意时、某个任务必须被完成或是确认关闭时，或保存重要数据时才考虑使用模态视图。

提供一个明显并可靠的退出模态任务的方式

确保用户总是知道他们关闭一个模态视图将导致的结果。

保持模态任务简单、简短并且高度集中

不要在你的应用中创建一另一个应用。如果一个模态任务太过复杂，用户在进入模态视图时就会看不到视他们本想执行的任务。当创建一个包含多层级视图的模态任务时请格外谨慎，因为用户可能会在多个视图中迷失并不知道如何返回。如果一个模态任务必须含有次视图，那么请提供单级的跳转路径以及清楚的完成路径。除非完成任务否则不要使用标有“完成”的按钮。

如果合适的话，请使用能够明确说明任务的标题

你也可能在视图的其它部分提供详细描述任务的文字或是提供指导。

只有在传达关键以及需要操作的信息时才使用警告框

警告框干扰体验，并且需要单击才能关闭，所以必须要让用户认为这个打断是有理由的。了解更多，请参阅Alerts。

尊重用户的通知偏好设置

在设置里，用户明确规定了他们想要如何地接受来自你应用的通知。遵循这些个人偏好，这样他们就不会想要完全地关闭来自你应用的通知推送。

不要让模态视图盖在在浮出层上

除了警告框，任何元素都不应该覆盖在浮出层之上。在极少数情况下，你需要让模态视图在用户完成浮出层内的任务之后弹出，那么请先关闭浮出层再展示模态视图。

让模态视图的视觉风格与你的应用相符

一个模态视图可能包含一个导航栏。在这种情况下，请使用与你应用内的导航栏一样的视觉风格。

选择合适的模态视图样式

你可以使用到以下任何一种样式：

样式	外观	适用于
全屏（Full screnn）	覆盖整个屏幕	可以在模态视图中完成的较复杂的任务
页列表（Page sheet）	在大屏设备或者横屏时遮盖了底下的大部分内容。未被遮盖的区域被模糊处理，避免用户与之交互。在小屏设备或是竖屏时遮盖全屏。	可以在模态视图中完成的较复杂的任务
表格列表	在屏幕中心显示，但是当键盘同时出现时可能会改变位置。所有未被遮盖的区域被模糊处理，避免用户与之交互。可能会在小屏设备中遮盖全屏。	收集信息
当前上下文（Current context）	与父视图大小一致	在对分视图、浮出层或者其它非全屏视图中展示模态内容

为展示模态视图选择一个合适的过渡方式

使用与应用风格相符的过渡方式来加强用户对当前内容转变的认知。默认的过渡方式让模态视图垂直地从屏幕底部向上滑出，然后在被关闭时下滑。弹出样式的过渡是指当前视图水平滑出，显示出模态视图，看起来就好像模态视图藏在当前视图的背后。当模态视图被关闭时，原先的视图便重新滑回来。在你的应用内容部使用统一的模态过渡方式。

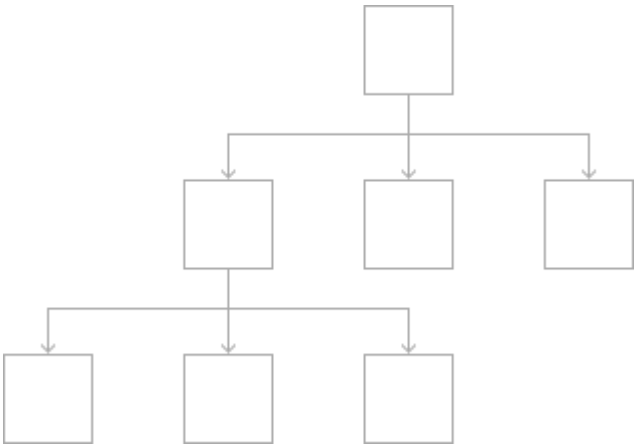
了解更多模态视图的实现方法，请参阅UIViewController Class Reference和UIPresentationController Class Reference。

2.12 导航（Navigation）

用户往往意识不到一个应用的导航，除非它没有达到他们的预期。你的工作就是实现一种能够支持应用结构和目的的导航，并且让人们注意到到导航的存在。导航应该让人觉得自然和熟悉，并且不应该主导界面或者抢走内容的风头。在iOS，主要有三种导航结构。

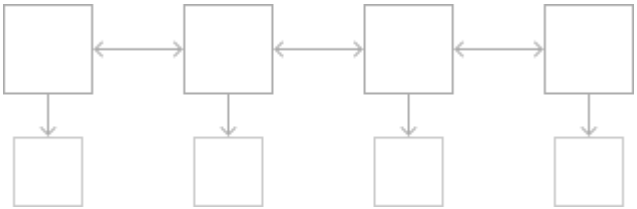
分层导航：

在每屏都做一次选择，直到你到达目标位置。要想到达另外的目标位置，你必须原路返回一些层级或是从头开始重新选择。原生应用设置（Settings）和邮件（Mail）就是采用这种导航结构。



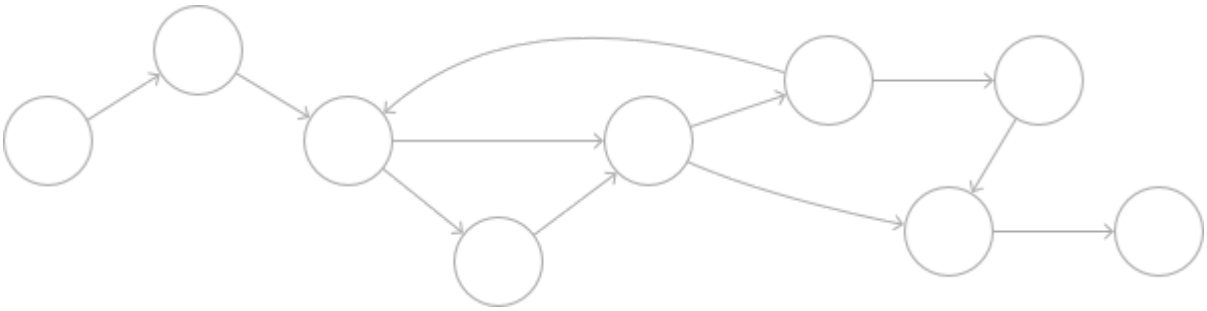
扁平导航：

在不同的内容类别间切换。原生应用音乐（Music）和App Store就是采用这种导航结构。



内容驱动或是体验驱动式导航：

在内容中自由地转换，或是内容定义导航。游戏、阅读以及其它沉浸式应用一般都采用这种导航结构。



有的应用结合了多种导航形式。比如，采用了扁平导航的应用也可能在每个类别之内使用层级导航。

总是提供清晰的路径

用户应该一直知道他在应用的什么位置以及如何去往下一个目标位置。除了要有清楚的导航形式，还应该确保对象间的路径是合理的、符合预期的并且容追溯的。一般来说，为用户提供到达某一屏的唯一路径。如果他们需要在非常多的情景下看到某一屏幕的内容，那么考虑采用操作列表、警告框、浮出层或是模态视图的形式展示这些内容。了解更多内容，请参阅Action Sheets,Alerts,Popovers, 和Modality。

设计一个能够快速简单地访问内容的信息结构

合理地组织你的信息结构，保证它只用最少次数的点击、横扫和屏幕间跳转就能访问相应的内容。

使用触摸手势来制造流畅感

让用户能轻松地在界面内跳转，而感受不到阻力。比如，你可以让用户在屏幕边界右滑，而返回到上一屏。

使用标准的导航组件

可能时，使用标准的导航控件比如页面控件、标签栏、分段控件、表格视图、集合视图和拆分视图。用户已经熟悉了这些控件，他们很自然地就知道如何玩转你的应用。

使用导航栏访问分层内容

导航栏内的标题栏能够说明当前的层级位置，使用返回按钮能够轻易地回到上一个位置。了解更多指导，请参阅Navigation Bars。

使用标签栏来展示内容或功能相似的类别

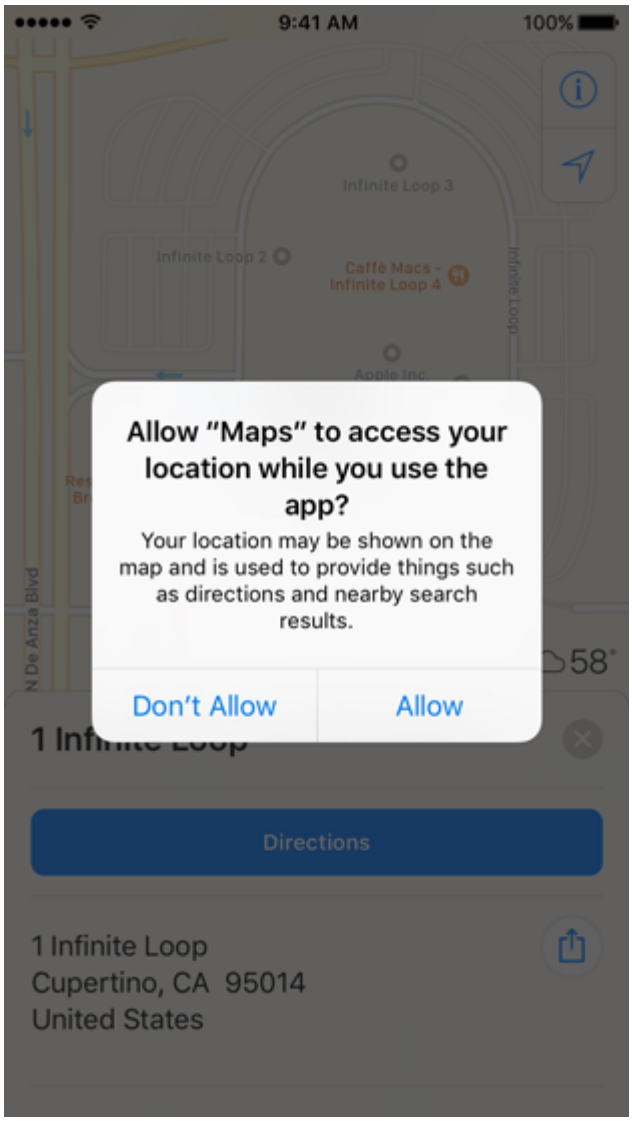
标签栏让用户能够快速简单地在类别中切换自如，而不受当前位置的限制。了解更多指导，请参阅Tab Bars。

使用多页面展示同类型的内容时请使用页面控件

页面控件能够清楚地表示总页数，以及当前页的位置。天气（Weather）应用就使用了页面控件来表示不同地理位置的天气页面。了解更多指导，请参阅Page Controls。

> TIP
>
> 分段控件和工具栏不具备导航功能。使用分段控件能够组织信息放入不同的类别。使用工具栏为当前内容提供交互控件。了解这些元素的更多信息，请参阅Segmented Controls和Toolbars。

2.13 请求许可（Requesting Permission）



用户必须对应用予以授权，应用才能获取用户的个人信息，比如当前位置、日历、联系人信息、提醒事项以及照片。虽然用户在使用获得这些信息的应用时会感到方便，但是他们还是希望能够控制自己的私人数据。比如，用户希望为他们的照片自动标上当前的地理位置，或是寻找附近的朋友，但是他们又同时希望能有关闭这些功能的选项。

只在应用真的需要时才向用户请求获得个人数据

用户会质疑个人信息的请求是很自然的，尤其是他们发现当前的请求没有明显的必要时。确保允许请求只在用户真的在使用某些需要个人数据的功能时才出现。比如，一个应用只有在激活一个位置跟踪的功能时才请求获得当前的位置。

当需求不明显时向用户解释为什么你的应用需要这些信息

你可以在系统提供的允许请求警告框上添加自定义的文本。使用明确且有礼貌的文本，这样用户就不会感到有压力。使用简短文本，并且使用句子。没有必要包含你的应用名字。系统已经替你在警告框上说明了应用的名字。

在应用一启动时就请求允许那些对运行你的应用至关重要的信息

如果用户明确地知道你的应用只有获得这些个人信息才能运行，那么他们就不会反感。

不必要时不要请求位置信息

在获得位置信息之前，检查系统以查看位置服务是否已经被打开。使用这个知识，可以延迟提醒，直到使用需要该信息的功能时才进行提醒，甚至可能完全避免提醒。

学习如何实现定位功能，请参阅Location and Maps Programming Guide。

2.14 设置（Settings）

有一部分的应用可能需要一开始就让用户决定设置或布局选项，但是大部分应用避免或是延迟这么做。成功的应用能够一开始就让用户很好地使用，并且同时提供了一个便捷的途径去调整体验。当你的应用被设计成满足大部分用户的需求，你就可以减少他们对对设置的需要。

推断你可以从系统中得到什么

如果你需要关于用户、设备或是环境的信息，那么尽可能地向系统请求而不是直接询问用户。比如，如果你想要知道用户的邮编来提供本地的选项时，可以向用户请求获取他们的当前位置。

在你的应用中对配置选项的优先排序深思熟虑

应用的主屏是一个放置关键或是常用选项的绝佳位置。次屏则适合放置只偶尔才更改的选项。

把不经常更改的配置选项放到系统设置里

系统的设置（Settings）应用是更改系统配置的核心地带，但是用户必须离开的应用才能到达那里。因此在你的应用中直接调节设置更加方便。如果你的应用必须提供很少改动的设置选项，请参阅Preferences and Settings Programming Guide中的Implementing an iOS Settings Bundle 部分。

适当时提供去设置的快捷路径

如果你的应用包含引导用户去设置的文本，比如“去设置>我的应用>隐私>定位服务”，请提供一个能够自动打开该界面的按钮。了解如果实现这个行为，请参阅UIApplication Class Reference中的Settings Launch URL部分。

2.15 用辞（Terminology）

每一个在应用中的文字都是与用户对话的一部分。利用好这个对方让用户在你的应用中感到自在舒适。

使用熟悉易懂的单词和短语

科技可以让人感到害怕。避免使用用户可能不理解的或是技术术语。根据你对用户的了解来决定哪些单词和短语是合适的。总的来说，能够吸引每个人的应用是不应包含深奥的技术语言的。这类语言比较适合针对高端或是技术用户的应用。

保持界面文本的清晰和简洁

用户能够快速且轻易地理解短而直接的文本，他们不喜欢在完成任务时被强迫去阅读很长的文本。找到最重要的信息，简洁地陈述它，然后突出地展示它，这样用户就不需要为了知道他们在找什么或是下一步该做什么而阅读太多信息。

避免使用让人听起来很傲慢的语言

避免使用“我们”，“我们的”和“我的”（比如“我们的教程”和“我的锻炼”）等字段。他们有时候被理解为无礼或是傲慢的。

尽量使用日常且友好的语气

一个日常亲近的风格就类似你在和别人吃午饭时聊天的语气。偶尔使用简写，并使用“你”和“你的”来直接与用户对话。

请谨慎使用幽默

记得用户可能会多次阅读你界面上的文字，而那些第一次看起来很俏皮的文字可能在多看几次之后会显得恼人。同样记住在一种文化中的幽默方式可能并不适用于其它文化。

使用相关且一致的语言和图像

确保引导在当前环境中总是合适的。如果某人在使用iPad，那么久不要给他展示与iPhone相关的文字和图片。根据平台选择使用相符的语言。你在触摸屏上点击、滑动、横扫、捏合或者拖曳对象。你按压物理按钮，或者按压对3D触摸作出反应的对象。你旋转和摇晃设备。

提供精确的日期

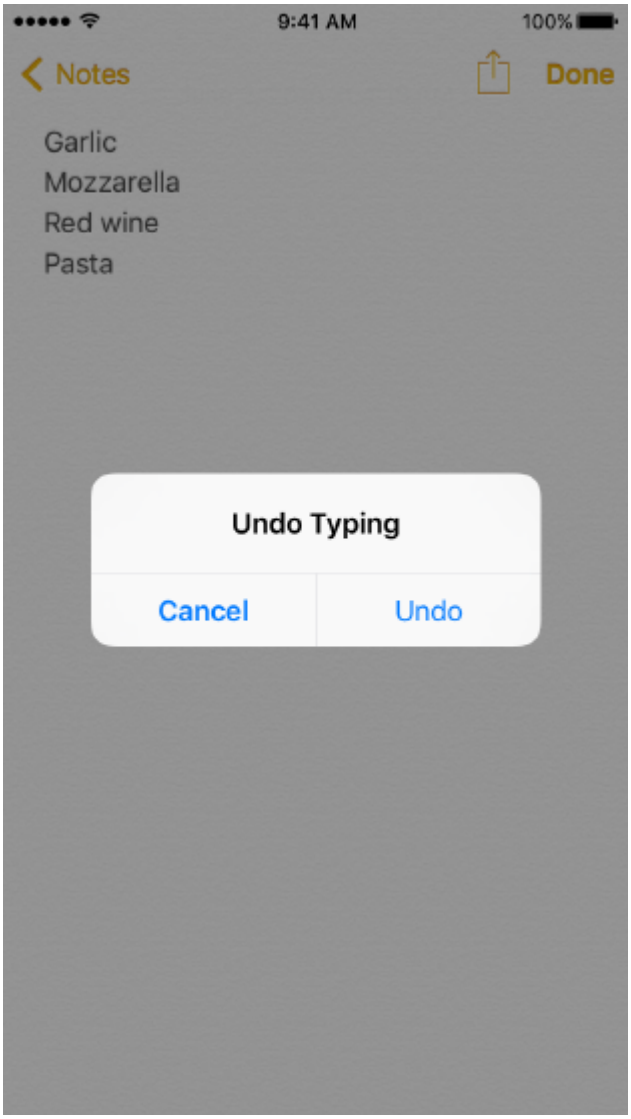
使用今天、明天这类友好的词语是合理的，但是如果你没有详细说明当前的位置，那么这些词语就会令人困惑或是显得不够精确。请考虑一个在午夜12点前发生的事件。在某个时区，这个事件可能发生在今天。但是在另一个时区，同样的事件可能在昨天就已经发生了。总而言之，日期应该体现出正在查看事件的用户所在的时区。然而，在某些情况下，比如一个跟踪航班状态的应用内，明确地显示起飞地区的日期和时区才更加清楚。

恰当地指出可交互的元素

用户应该瞥一眼就能知道这个元素是什么用的。当给按钮或是其它可交互元素标记时，使用操作动词，比如连接、发送和添加。

2.16 撤销和重做（Undo and Redo）

很多的应用都允许用户通过摇晃设备来撤销或是重做某个操作，比如打字或是删除。当该撤销和重做通过摇晃被触发时，会出现一个提示框，询问用户是要撤销（重做）操作还是什么都不执行。



简明扼要地描述将要被撤销或是重做的操作

撤销和重做的提示框标题会自动地包含“撤销”或是“重做”这样的前缀（以及后面的空格）。你需要在前缀后面提供额外的一两个词语用来形容什么会被撤销或是重做。比如，你可以创建一个提示框标题叫做“撤销命名”或者“重做地址更改”。

如果你已经把摇晃手势用来撤销和重做，那么就不要把它用于其它操作

即使你能通过编程赋予摇晃手势不同的意义，但同时你也冒着很大的风险使用户困惑，并让你的应用变得不可预知。

节制地使用撤销和重做按钮

如果在应用中为执行相同任务提供多种途径便会让人困惑。如果你的应用真的需要专门的撤销和重做按钮，那么请使用系统提供的标准按钮并且把它们放在一个符合预期的位置，比如导航栏。

只在当前情境中执行撤销和重做操作

撤销和重做必须对当前的（而非之前的）情境有明确直接的影响。

3. 特性（Features）

- 3.1 多任务处理（Multitasking）
- 3.2 通知（Notification）

- 3.3 打印（Printing）
- 3.4 快速查看（Quick Look）
- 3.5 Siri

4. 视觉设计（**Visual Design**）

- 4.1 动画（Animation）
- 4.2 品牌化（Branding）
- 4.3 颜色（Color）
- 4.4 布局（Layout）
- 4.5 字体（Typography）

3.1 多任务处理（**Multitasking**）

多任务处理让你能够通过iOS设备上的多任务处理界面或是在iPad上使用四指手势，随时且快速地从一个应用切换至另一个。在iPad，多任务处理也能够让你在Slide Over,分屏视图（Split View）或者画中画（Picture in Picture）模式下同时使用两个app。从屏幕右侧横扫就能进入Slide Over模式，它能让你在不离开当前app的情况下暂时性地使用第二个app ，比如在你使用Safari浏览器的时候快速地查看你的邮箱。分屏视图让你同时地使用两个并排的app，而画中画让你在一个app工作时也能观看视频。

设计一个能在多任务处理环境下从容工作的app取决于你的app是否能够和谐地于其它应用在设备上共处。也就是说，你的app不应该使用过多的CPU、存储空间、屏幕空间或是其它系统资源。它应该合理应对来自其它app的突发干扰和声音，快速流畅地隐去后台或是从后台中被呼出，并且即使在后台也能可靠地运行。

准备好应对中断，并且准备好随时恢复。你的app在任何时候都可能被中断。当中断发生时，你的app应该快速精准地保存当前的状态，这样用户返回app时，就能完美无缝地从他们上次离开的地方继续使用。了解更多实施细则，请参阅 App Programming Guide for iOS 中的 Preserving Your App's Visual Appearance Across Launches 部分。

确保你的界面能够与双倍高度的状态栏（double-high status bar）共处。一些比如进行中的通话、录音和共享功能会在屏幕顶部展示额外的一个状态栏。在对此无准备的app中，这个增加的高度会遮挡其它的界面元素或是把它们挤下去，从而导致布局问题。在你的app中测试这些情况以保证你的界面能够应对自如并且仍然看起来很棒。

暂停需要时时关注或是用户参与的活动。如果你的app是一个游戏或是观看媒体文件，请保证用户在切换至其它app时也不会错过任何内容。当他们切换回来时，让他们从上一次离开的地方继续就好像他们从未离开过。

恰当地处理来自应用外的声音。有时候，你的app的音频可能会被来自其它app或是系统的声音打断。比如，来电铃声或是被Siri打开的音乐播放列表会中断你的app的音频。当这种情况发生时，你的app应该以符合用户预期的方式处理。对于重要的音频干扰，比如播放音乐、广播或是有声读物，你的app应该停止播放它自己的音频。对于短暂的干扰，比如GPS导航通知，你的app应该暂时地降低它的音频音量或是先暂停音频然后在干扰结束时继续播放。了解更多指导，请参阅 Audio。

在后台完成用户发起的任务。当用户开启了一个任务，即使离开了app他们也希望任务能够被完成。如果你的app正在执行一个不需要额外用户输入的任务，请在app回到前台前在后台完成它。

节制地使用通知。无论你的app在前台、后台或是完全没有被打开，它都能在特定的时间给用户推送通知。使用通知来传达重要讯息是可行的，但是避免让用户被过多通知烦扰。比如，当你的app在后台时，不要每完成一个任务就给用户展示一个通知。相反的，让用户通过返回你的app来查看任务的完成情况。了解更多指导，请参阅 Notifications。

了解关于iPad的实施细则，请参阅 Adopting Multitasking Enhancements on iPad。

3.2 通知（**Notification**）

无论设备被锁屏还是在使用中，app都能随时利用通知来提供及时和重要的信息。比如，通知可能会在以下几种情况出现：当新消息到来时、一个事件将要发生时、有新数据可获取时或是某些状态发生改变时。用户在锁屏上、在屏幕顶部（使用设备时），以及通知中心（通过从屏幕顶部边缘下滑呼出）看到通知栏。每个通知都包含应用名称、一个app图标以及一条消息。通知的到来也可能伴随声音提示，以及app图标上小红点角标的出现和更新。

每个app的通知行为都可以在设置里面被单独管理。只要是支持通知功能的app，你有可以完全地打开或关闭这个功能。你同样可以设置通知是否在通知中心和锁屏上可见，是否在app图标上出现角标，以及选择以下一种通知样式：

横幅：当设备在使用时在屏幕上方出现几秒，然后消失。

提醒框：当设备在使用时在屏幕上方出现，直到被手动关闭。

在未锁屏的设备上通过点击通知、或是在锁屏时右滑，来结束通知、把它从通知中心移除并打开发送通知的应用展示相关的内容。比如，在未锁屏的设备点击一条新的邮件通知，就会打开邮箱并且显示新的信息。

在一个未锁屏的设备，上滑通知或让它消失能够关闭通知，也可能将它从通知中心移除。

使用3D Touch 在一个通知上按压，或时在未锁屏时在通知上下滑，就能打开拓展的详情视图。这个视图支持自定义并且包含最多四个操作按钮。比如，一个待办事项app可以推送一个含有详情视图的任务通知，上面有可以推迟任务和标记为已完成的按钮。一个日历事件的通知提供了“小睡”操作来推迟事件的闹铃。

> **TIP：**
>
> 通知可以是本地或是远程的。本地通知由同一个设备发出和接收。一个待办事项应用就使用本地通知来提醒用户一个将要到来的会议或是到期日。远程通知，也叫做推送通知，来自一个服务器。一个多玩家游戏就使用远程通知让每个玩家知道什么时候轮到他们了。
>
> 用户必须明确通过选择来接收来自每个app的通知——他们在第一次使用app的时候都被要求这么做。如果有人选择不接收通知，他们同样也能通过访问设置来选择接收。

提供有用的通知。用户打开通知是为了快速获得最新消息，所以你的重点是提供有价值的信息。使用完整的句子，句首字母大写，合适的标点符号，并且不要截短你的信息——必要时系统会自动处理。避免在通知中引导用户打开你的app，进入指定页面然后点击指定按钮来完成一些任务，因为当通知被关闭时用户很难记住它们。

就算用户没有作出回应，也不要为同一件事情发送多个通知。用户只有在方便时才会理会通知。如果你为同一件事发送了多个通知，并且填满了通知中心，那么用户就很可能关闭来自你app的通知。

不要包含你的app名字和图标。系统会自动地在每条通知的顶部显示这些信息。

角标是用来补充说明通知，而不能表示重要的信息。记住app的角标可以被关闭。如果你的app依赖于通过角标来传达重要信息，就等于你在冒着用户会错过这些信息的风险。

保持角标实时更新。当收到对应的消息时立即更新你的app角标数字。你也不想让用户猜测是否收到了新消息，他们只有在看到确切提示之后才会进入你的app查看。请注意将角标上的数字清零意味着同时在消息中心移除所有相关的通知栏。

提供声音以辅助你的通知提醒。当用户没有盯着屏幕时，声音是一个引起他们注意的绝佳方式。一个待办事项app，在需执行重要任务时可能就会播放一个提示音。你的app可以使用自定义声音或是系统的提示音来达到效果。如果你使用自定义的声音，请确保它是简短、与众不同并且制作精良的。请参阅Local and Remote Notification Programming Guide中的Preparing Custom Alert Sounds部分。请记住用户可以随时地关闭通知提示音。他们也可以开启伴随着声音的振动——这只能被手动开启，而不能通过你的app程序来关闭。

考虑提供一个详情视图。一个通知的详情视图提供了关于该消息的更多信息，并且允许他们在不离开当前环境的情况下去执行快速的操作。这个视图应该包含有用、易识别的信息，让人感觉就是一个你的app自带的插件。它可以包含图片、视频以及其它内容，它还能在显示时动态更新。比如，一个拼车app就能够在该视图展示一个地图，并标出正在朝着你的位置行驶的汽车位置。

提供直观、有用的操作。一个通知的详情视图能最多包含四个操作按钮。这些按钮应该用来执行常用、省时的任务，而不用通过打开你的app。使用简短、首字母大小写的名称，明确地描述操作的结果。一个通知的详情视图还能在屏幕上呼出一个键盘用来收集执行操作需要的信息。比如，一个通讯app可以允许用户直接在新消息通知上回复。

避免展示破坏性的操作。要在通知详情视图里展示破坏性操作之前请仔细考量。如果你必须展示它们，确保用户拥有足够的上下文信息，以避免出现意外后果。破坏性的操作应该以红字呈现。

了解更多实现细节，请参阅Local and Remote Notification Programming Guide。

3.3 打印（Printing）

你的app能够利用系统自带的AirPrint技术来使用兼容的打印机实现图片、PDF以及其它内容的无线打印。当用户在有AirPrint功能的应用内浏览可打印的内容时，他们一般通过在导航栏或是工具栏点击一个操作按钮，然后再点击打印按钮来打开打印视图。 这个视图提供了一个可用打印机的列表以及一些自定义选项，比如打印的份数、页面范围，并且提供了一个开始打印的的按钮。

让打印选项易于发现。如果你的app有一个工具栏或是导航栏，请使用系统提供的操作按钮来打印。用户对这个按钮更加熟悉，并且在其它应用中也是用它来打印。如果你的app没有工具栏或是导航栏，那么设计一个自定义的打印按钮来代替。

只在可以打印的情况下才允许打印。如果在你的屏幕上没有任何内容或是没有可用的打印机，那么不要在用户点击“操作”按钮后显示打印按钮。如果你的app使用自定义的打印按钮，在无法打印时让其不可点击或是隐藏。

提供有价值的打印选项。想想用户在打印来你的内容时会想要指定哪些选项。考虑可以选择页面范围和打印份数的选项。启用附加的选项，比如双面打印，如果这样有意义并且打印机也支持的话。

了解更多实现细节，请参阅 Drawing and Printing Guide for iOS 和 UIPrintInteractionController Class Reference。

3.4 快速查看（Quick Look）

在你的app中，快速查看让用户能够预览Keynote,、Numbers、Pages、PDF文档、图片以及其它类型的文件（即使你的应用并不支持这些文件格式）。邮件（Mail）使用Quick Look来查看附件。在下载附件之后，Mail在邮件信息内显示附件的图标和文件名。点击图标就能预览附件。

在当前环境下合理地展现预览视图。在iPhone上，如果你的app有导航栏，让预览视图下移留出位置给导航栏，就和你的app其它层级的视图一样。在iPad或是没有导航栏的app内，在一个全屏的有导航栏的模态视图中打开预览视图。通过以上两种方法，导航栏就能提供退出Quick Look的按钮，以及预览特有的一些按钮，比如分享和标记。如果你的app包含一个工具栏，那么预览特有的按钮就会在工具栏出现而不是导航栏。

了解更多实现细节，请参阅 Document Interaction Programming Topics for iOS 和 Quick Look Framework Reference for iOS。

3.5 Siri

你的app可以与Siri联动来执行一些任务以应对来自用户的语音命令和问题。

音频和视频通话应用：拨打电话和查找通话记录

消息应用：发送消息和阅读收到的消息

提供支付服务的应用：发送和请求支付

管理图片的应用：查找和显示图片

提供交通服务的应用：预定行程和提供行程状态信息

提供健身活动的应用：开始、暂停、恢复、结束和取消锻炼

与CarPlay联动的车载软件：更改车上的音频源、温控系统、除霜设置、座椅温度和无线电台。

Siri负责语言处理和语义分析来把语音请求转换成你的app能够处理的操作指令。你的app应该负责定义它所支持的功能、验证收到的信息、为Siri提供它需展示的信息以及采取操作。

在验证信息的时候，如果某些信息丢失或是不明确，你的app可以指示Siri来展示选项，向用户请求确认，或是请求更多信息。按照Siri的逻辑，某些任务比如发送信息和支付，在app执行任务之前首先需要用户确认。

来自你的app的回应信息会由Siri说出来并且呈现在Siri界面。合适的话，你的app可以提供自定义的内容让Siri来展现。比如一个健身app，可能会提供自定义的锻炼信息。

力求一个无需触屏或注视屏幕的声控体验。用户在使用Siri时不会经常盯着屏幕看。他们可能会通过耳机、汽车或是穿过房间来与Siri互动。尽可能地，让用户在无需解锁屏幕的情况下也能完成任务。

快速应答并且减少交互操作。用户使用Siri是为了方便，所以不要让他们等待回应。你的app应该在收到请求之后尽快地验证信息以及采取操作。当需要说明和更多信息时，呈现高效且集中的选项以降低需要额外提示的可能性。

将用户直接带到指定内容。从Sir转换你的app，应该直接去往用户期望的目的地。不要显示中间画面或是信息，阻碍转换过程或是拖慢用户。

保证是相关的、精确的以及合适的。你的app的回应必须和当前的请求相关而且必须精确地反映用户期望。永远不要包含可能会被认为是冒犯或是侮辱人格的内容。

将最安全、价格最低的选项设为默认值。一个应答无论如何都不应该骗人或歪曲信息，尤其是当它会带来经济上的影响时。对于一次涉及不同价位的购买，不要默认选择最贵的。当用户决定付钱时，不要在不告知用户的情况下收取额外的费用。

提升自定义词库的准确性。通过定义那些用户可能会在请求时用到的特殊术语，比如联系人、照片标签、相册名字、路线选择和运动名称，你的app能够帮助Siri去了解更多与执行你应用操作有关的内容。这些术语必须在你的app中是非一般的、特殊的，并且用户可能会在发起请求时真正用到。这些你提供的词汇不能包含其它应用的名字、与其它app明显相关的术语、不合适的语言或是被系统占用的短语，比如“Hey Siri”。记住任何你定义的术语都会被Siri用来解决用户请求，但是并不保证一定能够被识别。

提供请求例句。为Siri提供例句，当用户点击Siri界面的帮助按钮时这些例句就会被展现在指南里。使用这些例句引导用户如何以最简单高效的方式通过Siri来使用你的app。

确保你的自定义界面与Siri很好的融合。你可以使用app特有的颜色、象征性图像或是其它设计元素来表达品牌风格，但是任何自定义界面元素必须让人感觉它们仍然适合于Siri界面。

不要在自定义界面包含你的app名字和图标。系统会自动展示这些信息。

不要打广告。属于你应用的Siri体验永远都不能包含广告、营销或是app内购买的推销。

不要试图模仿或是操控Siri。你的应用永远都不能模仿Siri，也不能试图复制由Siri提供的功能，或是提供一个来自于Apple公司的应答。

了解更多实现细节，请参阅 SiriKit Programming Guide。

4.1 动画（Animation）

贯穿于iOS系统的优美、精细的动画在用户和屏幕屏幕内容之间建立了一种视觉上的联系。当动画被合理利用时，它能够表达状态、提供反馈、加强直接操纵感，并且视觉化呈现用户的操作结果。

明智且审慎地使用动画和动效。不要为了使用动画而使用动画。过度或是无理由的动画会让用户感到不连贯或是错乱，尤其是在那些不能提供沉浸式体验的app中。iOS经常使用动效，比如在主屏和其它地方使用了视差效果，来建立用户对深度的认知。这些效果有助于增强理解和提升愉悦感，但是滥用它们就会让一个app变得令人困惑并且难以控制。如果你想使用动效，一定要进行用户测试以保证它们真的能完成使命。

使用连贯的动画。一个熟悉并流畅的体验能一直让用户参与其中。用户已经习惯了贯穿于iOS系统的精细动画，比如平稳的过渡、横竖屏之间的流畅转换和基于物理现实的滚动。 除非你在创造一个沉浸式体验，比如游戏，不然自定义动画都应该和系统自动地动画相符。

力求真实性和可信性。用户可以接受艺术创造，但是当动效没有意义或是违背了物理定律时，用户就会感到混乱。打个比方，如果用户通过在屏幕顶部下滑呼出一个视图，那么他们应该也能通过上滑将该视图关闭。

4.2 品牌化（Branding）

成功的品牌化不仅是单纯地在应用中添加品牌元素。优秀的app通过优雅别致的文字、颜色和图片来营造独特的品牌辨识度。提供足够多的品牌元素让用户感觉是处在你的app中，但要因为给予太多而变成干扰。

融入精妙的、不唐突的品牌元素。用户使用你的应用是获得娱乐、得到信息或是完成任务，而不是为了观看一个广告。要想达到最好的体验，请巧妙地将品牌融于应用设计中。让app图标的颜色贯穿于界面设计是一个在你的app中提供专属环境的好办法。

不要让品牌化阻碍了优秀的应用设计。首先，让你的app像是一个iOS app。保证它是直观的、易于导航的、易用的并且以内容为中心的。当你的app在其它平台也适用，不要为了保持品牌的一致性而牺牲了设计的质量。

内容比品牌化更重要。在屏幕顶部一直放置一个除了展示品牌元素以外没有任何用途的头栏，就意味着牺牲了用来浏览内容的空间。取而代之的，考虑采用低侵入性的方式来实现品牌化，比如使用自定义的配色方案和字体，或是巧妙地自定义背景。

抵制住想要在应用中到处展示logo的诱惑。避免在app中到处展示logo，除非它是品牌化中是必不可少的一部分。这点在导航栏中尤其重要，因为提供一个标题比logo更加有用。

遵循Apple的商标准则。Apple的商标不能在你的应用名字或是图像中出现。请参阅Apple Trademark List 和 Guidelines for Using Apple Trademarks。

> **TIP**
>
> App Store 为突出你的品牌提供了更多的机会。了解相关指导，请参阅 App Store Marketing Guidelines。

4.3 颜色（Color）

在iOS，颜色能够暗示可交互性、增加活力以及提供视觉的连续性。在挑选app色调的颜色时，请参考系统的色彩方案，以保证这些颜色无论是单独还是组合、在浅色背景还是深色背景上都看起来很棒。

在app内使用互补的颜色。你的app内的颜色应该和谐共处，不会互相冲突和干扰。如果你的app风格的基础色调是柔和的，那么使用一系列与之协调的柔和色调。

考虑在app中统一使用一种关键色来暗示交互性。在Note中，可交互的元素是黄色的。在Calendar中，可交互的元素是红色的。如果你定义了一种关键色用于传递可交互性，那么你要保证其它颜色不会与之冲突。

一般来说，选择与你的app logo相符的颜色数量有限的色板。巧妙地使用颜色是一个传达品牌的好办法。

避免给可交互和不可交互的元素使用相同的颜色。如果可交互和不可交互的元素是同一种颜色， 用户就很难知道到底哪里是可点击的。考虑半透明对颜色的影响。颜色在半透明元素之下和之上（比如工具栏）都会看起来很不一样。

在多种光线条件下测试你的app的颜色方案。光线会在市内和室外、房间氛围、不同的时间、气候等条件下明显地变化。你的app在现实世界中使用时看到的颜色不会一直和你在电脑上看到的颜色相同。你应该在不同的光线条件下预览你的应用来观察颜色的真实表现，比如在晴朗的户外。必要时，应当调整颜色以求在大多数的使用场景下提供最好的视觉体验。

考虑True Tone显示屏对颜色的影响。True Tone显示屏利用了环境光传感器来自动调整显示屏的白点，以适应当前环境下的光线情况。专注于阅读、照片、视频和游戏的应用可以通过确定一种白点纠正模式来强化或弱化True Tone的效果。了解更多实现细节，请参阅 Information Property List Key Reference。

关注色盲用户以及不同文化对颜色的认知差异。不同的用户看见的颜色是不一样的。比如，很多色盲用户很难分辨红色和绿色（以及任何灰色），或是蓝色和橘色。避免把这些颜色组合作为区分两种状态或值的唯一方式。比如，用红色方块和绿色圆形来表示下线和上线状态，而不是用红色和绿色的圆形。有些图形编辑软件含有能够帮助你证明你是否是色盲的工具。同样地，考虑你对颜色的app在其它国家和文化中会被如何看待。比如，在某些文化里，红色用来表示危险；但在另一些文化里，红色又有着积极的含义。请确保在你的app中的颜色传达了合适的讯息。

使用足够的颜色对比度。在app中，过低的对比度会让内容难以阅读。比如，图标和文本可能会和背景相融合。在线的颜色对比度计算器能够帮助你精确的分析应用中的颜色对比度，以确保它符合最佳标准。请确保你的app对比度至少达到4.5:1,但是7:1更好，因为它符合更加严格的辅助功能标准。

4.4 布局（Layout）

用户总是希望能够在他们所有的设备以及任何一种模式下使用他们最喜欢的app。在iOS，界面元素和布局能够被配置以在不同的设备中、在iPad中多任务操作时、分屏模式时以及屏幕旋转时，自动改变形状和大小。因此，提前计划并且设计一个在任何环境下都能提供非凡体验的app是十分重要的。

环境变化时保持当前内容的焦点不变。内容是你的最高使命。让焦点随着环境变化而改变是令人迷惑又沮丧的，它会让用户感觉当前的app失控了。

确保最重要的内容在默认大小下清晰可读。除非用户选择调整大小，否则不应该让用户横向滑动才能阅读重要的文字信息，或是放大才能看清重要的图片。

在app内保持整体一致的视觉外观。一般来说，具有相似功能的元素应该看起来相似。

利用视觉权重和平衡来表示重要性。大的对象能够抓住人的眼球，显得比小的更加重要。大的对象也更易于点击，当app在容易分散注意力的环境中（比如厨房和健身房）被使用时这点尤其重要。一般来说，把首要的对象放在屏幕的上半部分并且放在偏左的位置——处于从左往右的阅读环境时。

利用对齐来方便浏览，并且表达结构和层级。对齐让app看起来整齐有次序，当页面滑动时有助用户聚焦，更容易找到信息。缩进（indentation）和对齐还可以表明多组内容之间的关系。

避免无缘由的布局变动。即使用户旋转了设备，也不代表整体的布局需要变换。比如，如果你的app在竖屏模式展示了一网格的图片，那么在横屏模式你没必要依次展示同样的图片。相反地，你只需要简单地调整网格的尺寸就行了。尽量在任何环境下都能维持相当的体验。

可能时，同时支持竖屏和横屏模式。用户更喜欢在两种模式下都能使用app，所以最好能够满足他们的期望。

如果你的app只支持一种模式，那么请支持该模式的两种变量。如果你的app只能在一种模式下运行，那么确保它能够支持该模式的两种方向变化是十分重要的。比如，如果你的app只在横屏模式运行，那么无论Home键在左边还是右边，应用都该能正常使用。如果设备被旋转180度，那么你的app内容也该同时旋转180度。反之，当用户拿错设备方向时，你的app没有自动旋转，那么他们就会很自然地知道应该旋转设备。你无需告诉他们该如何纠正。

根据当前使用内容来定制应用对旋转的反应。比如，一个需要用户旋转设备来控制角色移动的游戏，就不会在游戏中根据设备的旋转来改变模式。但是，它可以根据当前设备的旋转方向来展示菜单和引导步骤。

为可交互元素提供足够的空间。尽量让所有控件都有不小于44pt x 44pt的点击区域。

准备好应对文本大小的改变。当用户在设置里选择了不同的文本大小，他们总是希望大部分的app都能合理适配。为了适应某些文本大小的改变，你可能需要调整布局。了解更多关于应用内文本使用的信息，请参阅Typography。

了解更多适应性的实现细节，请参阅 Auto Layout Guide。

4.5 字体（Typography）

iOS的系统字体是San Francisco。该字体有两个变种：SF UI Text （用于19p及以下大小的文本）和SF UI Display（用于20p及以上大小的文本）。当你在标签和其它界面元素应用了系统字体时，iOS系统会根据字号自动选择最合适的字体样式。它还会根据需要自动改变字体，以满足辅助性功能的设置。请前往 Resources 下载San Francisco字体。

作为iOS的系统字体，San Francisco含有拉丁、希腊和西里尔字母，以及多种用于其它文字系统的字体。

强调重要信息。使用字体粗细、大小和颜色来强调app中最重要的信息。

如果可能的话，使用单种字体。混合使用多种字体会让你的app看起来零散和草率。考虑使用一种字体和几种样式和大小。

可能时使用内置的文本样式。内置的文本样式让你能在视觉上清晰地表达内容，同时保持最佳的可读性。这些样式建立在系统字体的基础上并能让你得益于关键的排版特性，比如动态字体，能够根据每种字号自动调整字距和行间距。iOS含有以下文本样式：

确保自定义字体清晰可辨。iOS支持自定义字体，但往往很难阅读。除非你的app必须使用自定义字体，比如出于品牌目的或是为了营造沉浸式的游戏体验，否则的话请坚决使用系统字体。如果你使用了自定义字体，请确保它是可读的。

让自定义字体实现辅助功能。系统字体能够自动对辅助性功能做出反应，比如当需要加粗文本时。使用自定义字体的app，应该通过检查辅助功能是否启用或是向系统注册以收到设定更改的通知，来执行同样的行为。请参阅 Accessibility。

动态字体大小

SF UI字体的两个变种经过精心的设计，无论尺寸大小都十分清晰可读。动态字体通过让阅读者选择喜欢字体大小，从而提供了额外的灵活度。

- 最小号
- 小号
- 中号
- 大号（默认）
- 加大号
- 超大号
- 最大号

根据内容的优先级来应对字体大小的改变。不是所有的内容都一样重要。当用户选择了一个更大的尺寸，他们只想让自己关心的内容更易于阅读，而不是希望屏幕上的每一个文字都变得很大。

字体使用和字距

在界面原型中使用正确的字体变种。当字号不大于19点时使用SF UI Text。使用SF UI Display来展示20号及更大的文字。根据以下规则适当调整字距：

5. 图像（Graphics）**

- 5.1 应用图标（App Icon）
- 5.2 自定义图标（Custom Icons）
- 5.3 图片大小和分辨率（Image Size and Resolution）
- 5.4 启动画面（Launch Screen）
- 5.5 系统图标（System Icons）

6. UI 栏

- 6.1 导航栏（Navigation Bars）
- 6.2 搜索栏（Search Bars）
- 6.3 状态栏（Status Bars）
- 6.4 标签栏（Tab Bars）
- 6.5 工具栏（Toolbars）

5.1 应用图标（App Icon）

每个app都需要一个精美、令人印象深刻的图标，能在苹果商店和主屏幕夺人眼球。轻瞥图标的瞬间，是你的第一个机会来传达你的app的目的。你的图标也会在系统中经常出现，比如在设置里和在搜索结果里。

拥抱简洁。寻找个单一的元素能够表现你的app的精髓，然后通过一个简单但是独特的形状来表现这个元素。谨慎地添加细节部分。如果一个图标的内容或是形状过于复杂，那么细节就很难辨认了，尤其是在更小的尺寸时。

提供一个单独的焦点。为图标设计一个单独的、集中的点，使它能快速吸引注意力并且明确地代表你的app。

设计一个易于辨识的图标。用户不应通过分析图标才能弄清楚它代表什么。比如，邮件app的图标使用了一个信封，因为它普遍与邮件联系在一起。花点时间去设计一个好看迷人且精炼的图标，艺术性地传达你的app的目的。

让背景简单并且避免使用透明度。确保你的图标是不透明的，并且不要让背景变得杂乱。使用一个简单的背景，这样它就不会过度影响周围的其它图标。你没有必要将整个图标填满内容。

只有当logo全部或部分由文字组成时，才在图标使用文字。在主屏幕时，一个app的名称会在图标之下显示。不要包含没有意义的文字重复说明名称或是告诉用户该如何使用你的app，比如“Watch”或“Play”。如果你的设计包含了一些文字，那么请强调文字与你的app提供的实际内容相关。

不要包含照片、屏幕截图或是界面元素。影像细节在很小的尺寸会难于辨认。屏幕截图对于一个app图标来说太复杂了，也一般不利于传达app的目的。在图标中的界面元素会令人误解并且困惑。

不要复用Apple硬件产品的图形。Apple产品受版权保护，不能在你的图标或是图片中被二次创作。一般来说，避免复用设备的图形，因为硬件设计频繁地更新换代，这会导致你的图标看起来易于过时。

不要在界面里到处放置app图标。在app里发现一个图标用于多种目的会让人困惑。反之，考虑使用图标的色彩方案。请参阅 Color。

在不同的壁纸下测试你的图标。你不能预期用户会为他们的主屏幕选择什么样的壁纸，所以不要只是在一种深色和一种浅色的背景上测试你的图标。而是观察它在不同的照片上如何表现。在有动态背景的真实设备上试用它，因为背景会随着设备移动而改变视角。

保证图标的四角是方的。系统会自动覆盖一个遮罩层让图标变成圆角。

App 图标大小

每一个app都必须提供一大一小两个app图标。

小图标会出现在主屏幕，并且当你的app被安装后会被系统使用。

为不同的设备提供不同尺寸的小图标。确保你的app图标在所有支持的设备上都看起来很棒。

大图标会被用在苹果商店。

让小图标与大图标类似。尽管大图标有着与小图标不同的用途，但它终究是你的app图标。大图标一般都和小的看起来差不多，但是可以稍微丰富一些，更有细节，因为不会有任何视觉效果叠加在它上面。

Spotlight和设置图标

每个app都应提供一个小图标，在Spotlight搜索，如果关键词与app名称相符，iOS会展示该图标。同时，需要设置的app同样应该提供一个小图标用于在系统内置的设置app中展示。两个图标都应该清晰标识你的app——更理想地，它们应该与app图标相符。如果你不能提供这些图标，iOS就会压缩你的app主图标展示在上述场合中。

不要对用于设置的图标叠加或是描边。iOS会自动为所有图标加上1个像素的描边，以确保它们很好地呈现在设置白色的背景上。

TIP：

如果你的app能创建自定义文档，你无需额外设计文档图标，因为iOS会利用你的app图标自动创建文档图标。

5.2 自定义图标（Custom Icons）

如果你的app含有不能用系统图标表示的任务或模式，又或是系统图标与你的APP风格不符，你可以设计你自己的图标。自定义图标通常被叫做模板，它不含有色彩信息并且通过mask来创建你在导航栏、标签栏、工具栏或是主屏幕快速操作视图看到的图标样式。

创作简单、辨识度高的设计。太多的细节会让图标看起来粗糙且不具可读性。为一个大多数用户都能正确理解并且不会反感的设计而努力。

设计一个纯色并带有透明度的、无锯齿、无阴影的图标。iOS会去除所有的色彩信息，所以没必要使用多于一种的填充颜色。允许用透明度来定义图标的形状。

使你的自定义图标与系统图标有所区分。你的图标不能轻易地与某个系统图标混淆。如果你想让你的图标看起来与iOS图标家族相似，请使用非常细的描边去绘制它。1pt的描边适用于大多数图标（在@2x分辨率下使用2px）。

保持图标之间一致连贯。无论你只使用自定义图标或是混合使用自定义图标和系统图标，在app中的所有图标都应该在大小、细节程度、透视和描边粗细上保持一致。

提供两种自定义标签栏图标的版本。为未选中和选中态提供两个图标。选中态的图标经常是未选中态图标的填充版本，但是某些设计会使用不同的方法。比如，苹果的原生app经常会将内部细节反过来填充，增粗或减细描边，以及把图标放在譬如圆形的形状内。

不要在自定义标签栏图标内包含文本。如果你需要展示文本，请在标签下面直接加上标题并且适当调节位置。

不要复用Apple硬件产品的图形。Apple产品受版权保护，不能在你的图标或是图片中被二次创作。一般来说，避免复用设备的图形，因为硬件设计频繁地更新换代，这会导致你的图标看起来易于过时。

自定义图标尺寸

5.3 图片大小和分辨率（Image Size and Resolution）

iOS通过坐标系在屏幕上放置内容。该坐标系以点为测量单位，这些点映射到显示器中以像素表示。在一个标准分辨率的屏幕中，1点（pt）（1英尺的72/1）等于一个像素(px)。高分辨率屏幕有更高的像素密度。因为在相同的物理空间内有了更多数量的总像素，所以平均每点有了更多的像素。因此，高分辨率的显示屏需要像素更多的图片。

你必须为你的app支持的设备提供所有的高分辨率图像。依据设备的不同，将每张图像的固有像素乘以相应的比例系数，就可以得到该设备所适配的分辨率。一张标准分辨率的图像对应的比例系数为1.0，并被称为@1x图片。高分辨率的图像对应的比例系数为2.0或是3.0，并被称为@2x 和@3x图片。

打个比方，假设你有一张标准分辨率（@1x）的图片，它的分辨率为100px*100px*。那么，这张图片的@2x版本就是*200px*200px，@3x版本就是300px*300px。

为不同设备准备图片时请参照以下比例系数。

5.4 启动画面（Launch Screen）

启动画面出现在app刚开始启动时候。随后，启动画面会很快被app的首屏代替，让人感觉你的app是快速响应的。启动画面不是一个炫技的时机，它只是为了增强用户对你的app能够快速启动并且立即被使用的感受。每个app都应该提供一个启动画面。

∆ 启动画面

∆ 首屏

因为设备屏幕大小不同，启动画面的大小也有所差异。你可以以Xcode故事板或是一组静态(static)图片的形式为你的app所支持的设备提供启动画面。因为Xcode故事板灵活性高且易于适配，所以推荐采用该形式。你可以使用一个单独的故事板来管理你的所有启动画面。了解关于可适配界面的开发细节，请参阅 Auto Layout Guide。

设计一个与你的app首屏几乎相似的启动画面。如果你的启动画面包含了与首屏看起来不同的元素，那么用户会在启动画面过渡至app首屏时经历一次不愉快的跳转体验。

避免在启动画面包含文本。因为启动画面是静态的(static)，任何展示的本文都不能被定位。

淡化启动。用户通常会在不同的app之间频繁切换，所以请设计一个启动画面，能够让app的启动体验不易被人察觉。

不要打广告。启动画面不是一个宣传品牌的时机。不要设计一个类似开屏广告或是介绍窗口的登入体验。不要在启动画面包涵logo或是其它品牌元素，除非它们是你的app首屏的静态（static）元素。

静态启动画面图片

最好使用Xcode故事板制作启动画面，但必要时你也可以提供一组静态图片。根据设备创建不同尺寸的静态图片，并不要忘记包涵状态栏区域。

5.5 系统图标（System Icons）

iOS提供了大量表示常用的任务和内容类型的小图标，用来在导航栏、标签栏、工具栏和主屏幕快速操作中使用。因为用户对这些图标很熟悉，所以最好尽可能地使用这些内置的图标。

导航栏和工具栏图标

TIP：你可以在导航栏和工具栏使用文本代替图标来表示某项。比如，日历app在工具栏使用“今天”、“日历”和“收件箱”三个字符。你还可以使用固定空间（fixed space）元素为导航栏和标签栏的图标提供间距。

∆ 了解开发细节，请参阅UIBarButtonItem。

标签栏图标

△ 了解开发细节，请参阅UITabBarItem。

快速操作图标

△ 了解开发细节，请参阅 UIApplicationShortcutIcon。

6.1 导航栏（Navigation Bars）

导航栏出现在app屏幕的顶部，状态栏之下，它能实现在一系列有层级的app页面间的导航。当进入一个新页面时，导航栏的左侧会出现一个返回按钮，并且一般会标有上一个页面的标题。有时，导航栏右侧还有含有类似编辑或完成按钮的控件，用来管理当前视图的内容。在分屏视图内，导航栏可能只会出现在分屏视图的一个单独窗格。导航栏是半透明的，也可能会有一个背景颜色，并且在适当时可以被隐藏，比如当前屏幕有键盘时、施加了某手势时或是某个视图在调整大小时。

TIP：当不需要导航或是你需要多个控件来管理当前内容时，请使用工具栏。请参阅 Toolbars。

考虑在导航栏显示当前视图的名称。在大多数情况下，名称提供了环境，因为它让用户知道他们在看什么。但是，如果给导航栏命名看起来是多余的，你可以让名称栏空着。比如，备忘录(Notes)不会在当前笔记的导航栏上放名称，因为内容的第一行已经提供了所需的环境线索。

考虑在app最高层级的导航栏放置一个分段控件。如果这么做帮助你扁平信息层级，让用户更容易找到他们想要的内容，你会深受其益。如果你在导航栏使用了分段控件，请确保为返回按钮选择了正确的标题。了解更多指导，请参阅 Segmented Controls。

不要包涵多段的面包屑路径。返回按钮只能执行一个单独的操作，即返回到上一屏。如果你认为用户可能会因为忘记到达当前屏的完整路径而迷路，请考虑扁平你的app层级。

避免用过多控件填充导航栏。一般来说，导航栏最多只能包含当前视图的标题、返回按钮以及一个管理当前视图内容的控件。如果你在导航栏使用了分段控件，除此之外，该栏就不应再包含标题或其它控件。

给文本按钮留出足够的空间。如果你的导航栏含有多个文本按钮，文本可能会让多个按钮看起来像是同时运行的，导致按钮间难以区分。在按钮之间插入固定空间项使它们隔开。了解开发细节，请参阅 UIBarButtonItem Class Reference中的UIBarButtonSystemItemFixedSpace constant value。

考虑在显示全屏内容时暂时隐藏导航栏。当你想要关注内容时，导航栏会令人分心。暂时隐藏该栏以提供一个更加沉浸式的体验。地图app在浏览一个全屏地图时会隐藏导航栏及其它界面元素。如果你要隐藏导航栏，允许用户通过简单的手势复原导航栏，比如点击。

使用标准的返回按钮。用户已经知道，标准的返回按钮会让他们在信息层级中按原路径返回。但是，如果你使用了自定义的返回按钮，请确保它们看起来像是返回按钮，有直观的表现，和界面的其它部分保持一致，并且在app内统一使用该自定义按钮。如果你用自定义图片替换了系统提供的返回按钮，请同时提供一个自定义遮罩图片(custom mask image)。iOS使用这个遮罩来实现按钮标题在转场时的过渡动画。

了解开发细节，请参阅 UINavigationBar Class Reference。

6.2 搜索栏（Search Bars）

用户通过搜索栏在大量的信息中查找。搜索栏有两种样式：显眼（prominent）（默认）和极简（minimal）。“通讯录”使用了显眼搜索栏，含有引人注目的浅色背景。“照片”使用了极简样式，更好地融入了周边界面。搜索栏默认是半透明的，但也可以被设置成不透明的。有需要时，搜索栏也可以自动遮盖住导航栏。

△ 显眼

△ 极简

搜索栏含有一个单独的搜索框，该搜索框可以包含占位文本、清除按钮、书签按钮和结果列表按钮。除了搜索框之外，搜索栏还会含有一个退出当前搜索的取消按钮。

∆ 占位文本

∆ 清除按钮

∆ 取消按钮

∆ 书签按钮

∆ 结果列表按钮

让用户通过搜索栏而不是文本框去搜索。文本框不具备用户期待的标准搜索栏所拥有的外观特征。

包含清除和取消按钮。大部分的搜索栏都含有一个清除按钮用来清空输入栏的内容，以及一个取消按钮来快速退出搜索。

选择合适的搜索栏样式，使其能够反映出搜索功能在你的app中的价值。如果在你的app中搜索是个关键功能，使用默认的、显眼搜索栏样式。如果搜索功能使用频率不高，则使用极简样式。

必要时，在搜索栏提供线索和背景。搜索框可以包含占位文本来提示可搜索的类型，比如“搜索服装、鞋子和饰品”或只是简单的“搜索”二字。也可以在搜索栏正上方展示一行简明扼要的带有适当标点的文字，用来引导用户。比如股票（Stocks），就在搜索框上方展示了一行文本告知用户他们可以输入公司名称或股票符号。

考虑在搜索栏下方提供快捷键之类的内容。利用搜索栏下方的区域帮助用户更快地获取内容。比如浏览器（Safari），在你点击搜索框的时候会立即显示你的书签，无需输入任何关键词的情况下即可进入选择的对象。股票app在你对搜索栏输入的时候，会一边在下方展示相关的结果列表，你可以在列表中点击选择而不用完整输入字符。

必要时，为搜索栏添加书签和结果列表按钮。利用书签按钮让用户能够快速获得他们可能需要再次查找的信息，比如保存的、上一个或是最近的搜索记录。使用结果列表按钮来暗示搜索结果的存在，并在用户点击按钮的时候显示这些结果。但你无法同时展示上述两个图标。

了解开发细节，请参阅 UISearchController 和 UISearchBar。

范围栏（**Scope Bars**）

范围栏可以附加于搜索栏，让用户定义搜索的范围。范围栏采用和搜索栏一样的外观。

专注于优化搜索结果而不是一味添加范围栏。当用户想在定义明确的类别中搜索时，范围栏会有很大的帮助。但是，更好的做法是优化搜索结果，这样用户就无需通过范围栏进行筛选了。

了解开发细节，请参阅 UISearchBar。

6.3 状态栏（**Status Bars**）

状态栏在屏幕的顶端出现，显示与设备当前状态相关的有用信息，比如时间、运营商、网络状态以及电池容量。状态栏上真正显示的信息根据不同的系统设置有所变化。

使用系统提供的状态栏。用户希望状态栏在系统内部保持一致。不要用自定义的状态替换掉它。

∆ 明

Δ 暗

根据你的app设计选择协调的状态栏颜色。状态栏的文本和指标的视觉样式非明即暗，在你的app中，可以统一使用一种配色，或是根据不同的屏幕选择单独的配色。暗色系的状态栏在浅色的界面上效果好，浅色系的状态栏在深色系的界面上效果好。

遮盖状态栏下方的内容。状态栏的背景默认是透明的，这样会显示出状态栏下方的内容。既要保证状态栏的可读性，又不能让人误解下方的内容是可交互的，通常通过以下几种技巧来实现：

- 在你的app中使用导航栏，它会自动显示状态栏背景以保证状态栏下方不会出现任何内容。
- 在状态栏下方放置一张自定义图片，比如渐变或纯色背景。
- 对状态栏下方的内容进行模糊处理。

Δ 半透明

全屏展示媒体文件时考虑暂时地隐藏状态栏。当用户想要集中注意力在媒体上时，状态栏会令他们分心。暂时地隐藏状态栏元素能够提供一个更加沉浸式的体验。比如照片app，在用户全屏浏览照片时时会隐藏多余的界面元素。

避免永久地隐藏状态栏。在没有状态栏时，用户需要退出你的app去查看时间或是检查他们是否连接至Wi-Fi。允许用户可以通过简单、易于发现的手势来重新唤醒被隐藏的状态栏。在照片app中浏览全屏照片时，用户只需在屏幕上轻点即可呼出状态栏。

在状态栏显示耗时较长的网络活动状态。当你的app在使用网络，尤其是耗时较长的操作时，显示网络活动状态栏指示器，这样用户就知道活动正在进行中。请参阅Network Activity Indicators。

了解开发细节，请参阅UIApplication中的UIStatusBarStyle constant 和UIViewController中的preferredStatusBarStyle property。

6.4 标签栏（Tab Bars）

标签栏在app屏幕底部出现，提供了在app不同部分间快速切换的途径。标签栏是半透明的，也可能会有纯色背景，在横竖屏都保持一致的高度，并且在出现键盘时会被隐藏。一个标签栏可以包含无数个标签，但其所能容纳的可见的标签数量根据设备大小和横竖屏模式有所变化。受水平空间的限制，当某些标签无法被显示时，最后一个可见的标签会变成“更多”， 并通过该入口前往其余标签列表的另一屏。

一般来说，利用标签栏组织应用程序级别的信息。标签栏是扁平化信息层级的好办法，并且一次性提供了前往多个平级信息类别或模式的途径。

不要在某个标签的功能不可用时去掉该标签或是使其失效。如果标签时而可用时而不可用，你的app界面会变得不稳定和难以捉摸。确保所有的标签都是有效可点击的，并且向用户解释当前标签内容不可用的原因。比如，当本iOS设备没有歌曲时，音乐app的“我的音乐”标签页就对如何下载歌曲做出了说明。

标签栏只能作为导航。标签栏按钮不应该执行其它操作。如果你需要在当前视图提供作用于元素的控件，你可以使用工具栏。请参阅Toolbars。

避免太多的标签。每增加一个标签就减小了选择单个标签的可触区域，并且增加了app的复杂性，让用户更难找到所需的信息。尽管“更多”标签可以展示额外的标签项，但这需要额外的点击步骤，并且对标签栏的有限空间没有很好的利用。太少的标签也是个问题，它会让你的界面感觉被分离。一般来说，在iPhone上使用3至5个标签，在iPad上则可稍微多几个。

使角标（badge）低调地传达信息。你可以在标签上展示角标——一个带有白色数字或感叹号的红色椭圆——来暗示该标签视图或模式含有新信息。

总是在与标签栏相连接的视图切换内容。为了让你的界面符合用户预期，选择一个标签后应该直接作用于与标签栏相连接的视图，而不是屏幕其它范围的视图。比如，在分屏视图（split view）的左侧选择了一个标签，是不会导致右半部分突然变化的。在弹窗（popover）选择一个标签页不会导致下方的视图发生改变。

了解开发细节，请参阅 UITabBar。

TIP：

理解标签栏和工具栏之间的不同十分重要，因为这两种栏都是出现在app屏幕的底部。标签栏让用户在app的不同部分间快速切换，比如时钟app中的“闹钟”、“秒表”、“计时器”。工具栏包含了执行当前视图相关操作的按钮，比如创建项、删除项、添加注释或是拍照。请参阅Toolbars。标签栏和工具栏决不会在同一个视图内同时出现。

6.5 工具栏（Toolbars）

工具栏在app屏幕底部出现，包含了执行当前视图或包含内容相关操作的按钮。工具栏是半透明的，也可能会有纯色背景，并且通常在用户不太需要它们时被隐藏。比如，在浏览器(Safari)中，当你滚动页面表明你再阅读时，工具栏就藏起来了。当你在屏幕底部点击时，工具栏又会再次出现。当前屏幕有键盘时，工具栏也会被隐藏。

提供相关的工具栏按钮。工具栏必须包含在当前环境下有意义的常用操作命令。

考虑图标或文字按钮是否适合你的app。当你需要多于三个工具栏按钮时，图标可以实现。当你只有三个或是更少的按钮时，文字有时候看起来更加清楚。比如，在日历app中，文本就被当作按钮使用，因为图标可能会令人迷惑。因为使用了文本，“Inbox”按钮还能显示所有的日历和事件邀请数量。

给予文本按钮足够的空间。如果你的工具栏含有多个按钮，文本按钮就会挤在一起，导致按钮间难以区分。在按钮之间插入固定间距使其分离。了解开发细节，请参阅UIBarButtonItem中的UIBarButtonSystemItemFixedSpaceconstant value。

避免在工具栏使用分段控件。分段控件让用户切换内容，而工具栏更针对于当前屏幕。如果你需要提供切换内容的方式，请考虑使用标签栏替代。请参阅Tab Bars。

了解开发细节，请参阅 UIToolbar。