写在前面

关于AppleScript：AppleScript很简单——保证比VB还简单，但很实用！学起来很轻松！学会之后你会发现你节约了很多时间，摆脱了很多机械性的无聊琐事。

关于创作目的：国内目前有关AppleScript的资料还非常少，我希望以最简单最简洁的形式来介绍尽可能多的AppleScript知识。简明是我的最终目的！

关于截图：本教程中所有截图都是依据Mac OS X 10.6简体中文界面进行的，在10.5系统中通常不会有明显区别，如果存在区别会在出现图片的地方加以说明的。特别声明：教程中未注明出处或作者的图片均由我亲自制作。

关于制作：使用Pages排版，Photoshop制图。

关于教程：本教程基本均为原创，以本人经验和国外相关网站作为参考（如developer.apple.com和macscripter.com），主要的权威参考书目为苹果官方的《AppleScript Language Guide》2008年3月版（可在官方网站上免费下载）。本人才学粗浅，涉水AppleScript时日也不长，如有疏漏和错误，请不吝指正。电子邮件：nocturn21st@gmail.com，有任何问题也欢迎在Macidea.com上发帖讨论。

关于发布：本教程将陆续在MacIdea论坛上发布，欢迎转载。此外，欢迎大家用iWork.com在线阅读和批注。

其他：AppleScript中译名应当为“苹果脚本”，但为了和系统统一，本教程中将不予翻译。

我认真写好本套教程的动力来源你的支持！

本教程为MacIdea而作！正常情况下每周至少出一期，预计于一个月内全部完成。

iDoraemon Nathan
2009年8月

目录

第一章 AppleScript入门

本章将初步介绍AppleScript的概念和基础知识，之后将简单讲解Automator的使用。前三节是讲AppleScript的一些套话——比如吹捧它的功能，你可以略看，不过仔细阅读也不会有任何损失。从第四节开始我就要求你必须认真阅读了。

第一节 什么是AppleScript

AppleScript的概念可以大致可以用下面几个词来描述:

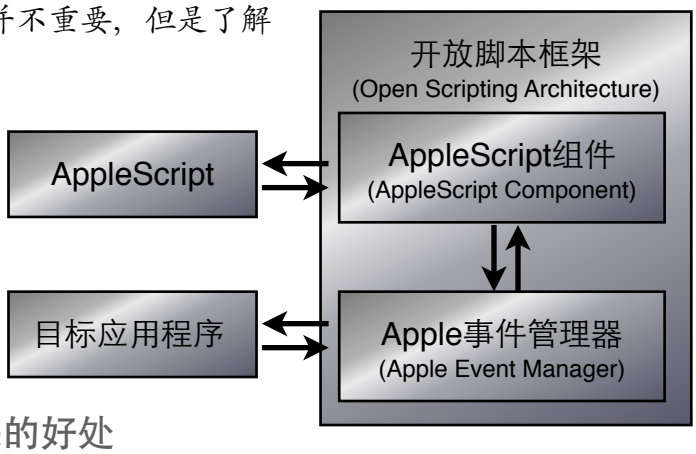
- 一种脚本语言
  和我们所知道的VBScript和JavaScript类似
- 内建于Mac OS
- 用来控制现有的应用程序
  请特别注意这一点！
- 使繁琐重复的机械操作自动化

第二节 AppleScript的工作机制

对于初学者初学者来说，工作机制并不重要，但是了解它终会有好处的。

如右图所示，AppleScript的工作机制中的四个部分均能实现双向交互。对于脚本编写者来说，只需要了解AppleScript和目标应用程序部分。

关于开放脚本框架(Open Scripting Architecture)我们暂时不需要了解。坦率地率，我认为不必要去了解。



第三节 AppleScript的用途和它带来的好处

AppleScript的用途举例:

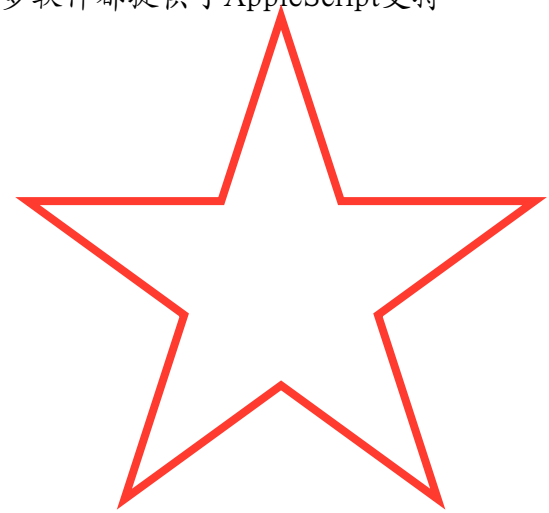
- 批量图片处理
- 网站日常维护
- 文件和文件夹维护
- 包括Adobe系列软件和Microsoft Office在内的很多软件都提供了AppleScript支持
- 还有很多很多。。。

AppleScript带来的好处:

- 高效率
- 低出错率
- 更高的统一性
- 更高的精确度
- 免去你的操心

什么时候用ApplesScript?

- 当需要做重复机械性的且耗时的工作时
- 当你需要在未来某个时刻还要做一样的事情时
- 当写一个脚本比实际上做那件事更快时



第四节 和AppleScript有关的程序和设置

如果你没有接触过AppleScript，请务必仔细阅读本节。

特别说明：AppleScript编辑器和设置工具在Mac OS 10.5 Leopard以及先前版本和10.6 Snow Leopard中有不同！请根据操作系统不同来调整。本节标题中括号内为Leopard和Tiger系统中的名称

AppleScript编辑器（脚本编辑器）

在10.5 Leopard和10.4 Tiger下：脚本编辑器和AppleScript实用工具位于“应用程序/AppleScript”文件夹中！

在10.6 Snow Leopard下：AppleScript编辑器位于“应用程序/实用工具”中；AppleScript实用工具已经不存在了，其功能合并入AppleScript编辑器，作为其偏好设置的一部分。

这个编辑器是我们用来编辑、调试乃至运行AppleScript脚本所必需的[1]。程序界面和其基本介绍如左下图。

在偏好设置（AppleScript实用工具）中打开“脚本菜单”

关于编辑器偏好设置（Leopard和Tiger中为单独的“AppleScript实用工具”），我只想提一样东西——“在菜单栏显示脚本菜单”（位于“通用”设置里）。
右侧的图片和下面几个问题是关于“脚本菜单的”介绍

- 脚本菜单是什么？
  就是预装的脚本——包括系统自带的和第三方提供的。它显示在菜单栏右侧（输入法的附近，如右图）。
- 脚本菜单用来做什么？
  快速打开已经编辑好的脚本。
- 如何添加自己的脚本到这个菜单？
  通过菜单第一项“打开Scripts文件夹”（本机的可被所有用户访问，用户的只能被当前用户访问到。），拷贝自己的脚本到这个文件。建议建立文件夹以保持整洁。

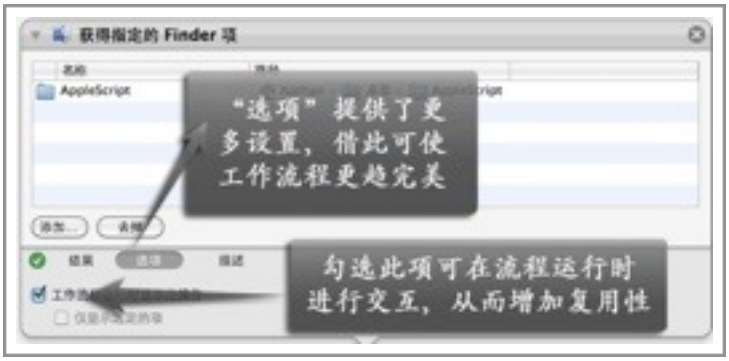
[1] 说是“必须”其实也未必，目前有第三方的AppleScript编辑开发软件，此外Xcode也提供了AppleScript的开发环境，同时Xcode也是所谓带有GUI的AppleScript Studio程序开发和编译所必需的。

第五节 Automator和AppleScript

Automator也是Mac OS自带的程序之一，他是一个“阉割版”AppleScript编辑工具，提供了直观的视图和简单的拖曳操作，但是功能上比AppleScript少很多（举例：Automator不支持循环）

下面图片中的流程可以实现对文件夹中所有图片进行批量格式转换。如果你有兴趣，请自己尝试下。（其实还是挺实用的噢）

优化工作流程：每个“指令”都有各自的选项，修改它们可以获得不同的结果，如右图



Automator虽然功能局限，但是并不代表它是鸡肋，它仍然具有很多的实用价值。本教程重点在于AppleScript的学习，故Automator将不作深入介绍。 Automator程序作为AppleScript的简化版，大家有空不妨多多动手尝试。你一定会觉得Mac还真的好用！

第二章 快速上手AppleScript编辑器[2]

本章将介绍AppleScript编辑器的一些“隐藏”功能，脚本的存储格式，以及应用程序对AppleScript脚本的支持情况，此外将通过一个实例来介绍并评价“录制脚本”功能。本章节对更好的理解AppleScript有一定的帮助。

第一节 挖掘实用的功能

在默认情况下，或者第一次使用AppleScript编辑器，你会发现这个编辑器和其他编程工具比起来非常不好用。但做一下几个简单的设置就可以让它迅速成为一个你的好助手。这里举两个例子，其他更多设置请读者按照自己喜好设置。

打开脚本助理

位于AppleScript编辑器的“偏好设置...”-“编辑”中。

此助理在默认情况下未打开，但是强烈强烈建议你勾选它！将会给你带来大大的方便。下面是主要的两个功能，第一个用处不大，但是第二个是非常有用的。

助理使用方法之一：输入代码时会以灰色字母或点来提示接下来应该输入的字母（有多个备选时以“..”表示，点的个数和字母数一致）

助理使用方法之二：只需输入代码的开头几个字母，按下F5键（根据你的设置，非常有可能是Fn+F5键），便会出现如右图所示的备选框。相信有编程经验的同学们不会陌生。

显示“tell application”弹出式菜单（仅限Snow Leopard）[3]

位于AppleScript编辑器的“偏好设置...”-“编辑”中。

为了直观说明这个特性，请对比下面两段代码（其实现的功能是一致的，都是在桌面上建立新文件夹），请留心图片上方的工具条。

未打开此功能时，需要输入三句：

打开此功能后，先在上方第二栏“tell current application”菜单中选择“Finder”，只需输入一句话：

第二节 脚本的存储格式

在“存储为...”对话框中，提供了四种备选方式来保存我们编写的脚本，即脚本、脚本包、应用程序和文本。这四种方式都比较常用。下面我将依次介绍。

---

[2] 在Mac OS 10.5以及10.4中，该名称为“脚本编辑器”。本书中所有“AppleScript编辑器”都是如此。请读者根据自己的操作系统进行修正。

[3] 此功能为“AppleScript编辑器” 2.3 (118)版本中提供的新特性，该版本随着Mac OS X 10.6提供，因此之前版本的用户将可能无法使用此功能。但是，旧系统用户下载2.3版的编辑器即可使用此功能。

脚本

这中保存方式直接将编辑的脚本保存为可运行（也许可以被编辑）的脚本，扩展名为.scpt。其不具有应用程序架构或者包结构。

在保存为脚本时，如勾选“仅运行”，将使得脚本不可被编辑，并且作为可执行文件打开，直接运行代码。

若未勾选“仅运行”，则其默认打开方式为“AppleScript编辑器”，代码可被编写并可更新该脚本文件。

脚本包

除了它具有包结构，扩展名为.scptd，其他和保存为“脚本”都一样。

所谓包结构，即在Finder中如右击（或者ctrl单击）该文件，会有“显示包内容”这个命令，其包中含有rtfd介绍文件、plist配置文件和sctp脚本。

此外，AppleScript编辑器窗口右上方的“包内容”按钮也将可用，并可修改其中内容。

应用程序

保存为扩展名为.app的应用程序，它将具有标准Cocoa程序的架构。它包内容含量比脚本包更多，进一步包含了图标，包简介(PkgInfo)，Unix可执行文件等等。

保存为此格式时，有三个选项：

- 仅运行：使应用程序包中的脚本不可编辑。
- 启动屏幕：使程序运行开始前显示一个对话框（包含description.rtfd的内容）。
- 保持打开：针对那些拖曳应用程序，使它始终处于可用状态。

文本

保存为扩展名为.applescript的纯文本文件。

第三节 支持AppleScript的应用程序

在第一章第一节中就已经提到了AppleScript是用来控制现有应用程序的，因此有必要了解哪些应用程序提供了AppleScript支持。

其实提供AppleScript支持的应用程序非常多，包括Finder、Adobe Photoshop、Microsoft Office[4]等等，这里不一一列举。查看程序是否支持AppleScript控制，请查阅对应程序的帮助文档。请注意，本教程将着重介绍Mac OS自带的程序（如Finder）。

应用程序对AppleScript的支持类型

即使支持AppleScript，不同的应用程序对它的支持方式不同，有下面三种类型：

- 可编程：通过输入脚本来控制应用程序
- 可录制：通过AppleScript编辑器的录制功能（将在下一节介绍）
- 可嵌入：应用程序支持AppleScript脚本交互式运行，应用程序在其菜单中有专门的脚本菜单（或者子菜单）。如Photoshop CS4中的“文件(File)-脚本(Scripts)”菜单

一个应用程序可以支持多种类型，并且很多程序都同时支持三种类型。

应用程序脚本的特点

- 应用程序通常都提供了自己的脚本控制指南
- 部分应用程序提供了示例脚本
- 不同应用程序的脚本编写难度不同
- 不同应用程序的脚本可进行的操作不同

---

[4] 此处应用程序均指Mac版本的对应程序。Adobe®，Microsoft®是各自公司的注册商标。

第四节 AppleScript的录制功能

“录制”允许用户通过最简单的方式来“输入”代码——记录你的每一步操作。Automator程序同样也支持“录制”。只需按下AppleScript编辑器左上角（Automator右上角）（如右图）的按钮就行了。

这个功能听起来很诱人，因为它简单易用并且可以帮助学习脚本语言。不过事实上它还是有很大的局限性的，主要体现在三方面：一是无法实现循环；二是将错误操作也记录了下来；三是代码质量低下，可读性差。

应用实例1：建立100个子文件夹

功能说明：这段脚本用于在桌面下建立一个名为“Test”的文件夹，并在其下建立100个子文件夹。

目的：现在你不需要理解这段代码！举这个例子的目的在于让你明白“录制”功能的局限性。你可是试试看录制出这段脚本（试想下，你需要确确实实地建立起100个文件夹）

以下为标准的代码

----------代码----------[5]

```
tell application "Finder"
	make new folder at desktop with properties {name:"Test"}
	repeat with a from 1 to 100
		make new folder at folder "Test" of desktop with properties {name:a as string}
	end repeat
end tell
```

以下为录制的代码（部分节选）

----------代码----------

```
tell application "Finder"
	activate
	make new folder at folder "Desktop" of folder "Nathan" of folder "Users" of startup disk
		with properties {name:"未命名文件夹"}
	set name of folder "未命名文件夹" of folder "Desktop" of folder "Nathan" of folder "Users"
		of startup disk to "Test"
	make new folder at folder "Test" of folder "Desktop" of folder "Nathan" of folder "Users"
		of startup disk with properties {name:"未命名文件夹"}
	set name of folder "未命名文件夹" of folder "Test" of folder "Desktop" of folder "Nathan"
		of folder "Users" of startup disk to "1"
	make new folder at folder "Test" of folder "Desktop" of folder "Nathan" of folder "Users"
		of startup disk with properties {name:"未命名文件夹"}
	set name of folder "未命名文件夹" of folder "Test" of folder "Desktop" of folder "Nathan"
		of folder "Users" of startup disk to "2"
	...

	--还有196句代码
	...
end tell
```

对比之下，我们很容易就发现录制出来的代码很糟糕。

[5] 根据本章第一节的介绍，可以通过“tell application”菜单来进一步精简代码。但是本教程考虑到与10.5和10.4系统的统一性，使用了最为标准的代码。

第三章 AppleScrip语言初步

本章内容是AppleScript的基础，内容比较多而且都非常重要，请读者仔细阅读。友情提示：看完后自己操作是最好的学习和复习的方法。

第一节 对象、属性和命令

AppleScript是一种面向对象（Object-Oriented，简称OO）的脚本语言，和现在主流的面向对象程序语言一样，它拥有三个重要的OO术语：对象（Object）、属性（Property）和命令（Command）[6]。

考虑到新手的学习，这里将通过一个实际生活例子——汽车来简单介绍下：

对象：又称“类”（Class），在生活中即一个具体的物体——汽车本身，当然也可以是一个部件，如轮子、车门、引擎——这三者作为汽车的一个组成部分，在OO概念里被称为子类。请记住，对象（类）是有可以拥有层次关系的。

属性：是对象的特性。如汽车的颜色、车身长度、重量都是对象汽车的属性。一个对象如果没有具体的属性，那么它将是抽象的或者是笼统的。

命令：又称“方法”（Method）。顾名思义，命令就是指令，它告诉脚本/程序应该去做什么。在非面向对象（面向过程）的语言中，命令或者方法可被成为“函数”。

第二节 标识符和关键字

首先值得说明的是：AppleScript的采用Unicode文字编码，并且不区分大小写。

标识符（Identifier）

标识符就是对象、属性、常量、变量等等的名称。就像人的姓名一样。

AppleScript规定：标识符必须以英语字母开头，可使用26个英语字母、10个阿拉伯数字以及下划线（_）。

特殊规则：如果标识符以“|”开头并结尾，则标识符可以使用任何Unicode字符，但是标识符名称本身是不包括“|”。

例：　abcd，ABC_91，a0abc，|a&b*c|，|中文名称|　　都是合法的标识符

　　91abc，_abc，中文名称，a&b*c　　不合法的标识符，请读者思考原因

建议：无论你是新手还是编程老手，强烈建议使用有意义的标识符，既方便自己也方便别人阅读。同时，避免使用符合“|”这一特殊规则的标识符。

关键字（Keyword）

关键字就是AppleScript保留的标识符，这些词通常拥有特殊含义（如被用于AppleScript语法），它们不能被用户定义为自己的标识符。

下表列出了常见关键字：请注意部分关键字由两个词组成。

about	back	considering	end	fourth
above	before	contain	equal	from
after	beginning	contains	equals	front
against	behind	contains	error	get
and	below	continue	every	given
apart from	beneath	copy	exit	global
around	beside	div	false	if
as	between	does	fifth	ignoring
aside from	but	eighth	first	in
at	by	else	for	instead of

[6] 关于面向对象的三个重要术语，英语术语及其翻译和其他程序语言可能存在不同。请注意，这里采用了苹果官方《AppleScript Language Guide》（可在苹果开发者网站developer.apple.com上找到）中的说法。

into	not	reference	tell	transaction
is	of	repeat	tenth	true
it	on	return	that	try
its	onto	returning	the	until
last	or	script	then	where
local	out of	second	third	while
me	over	set	through	whose
middle	prop	seventh	thru	with
mod	property	since	timeout	without
my	put	sixth	times	
ninth	ref	some	to	

第三节 数据类型

数据类型（Data Type或者Value Class）可以由AppleScript语言本身定义，也可以是由应用程序定义的。在AppleScript常见的有以下几种:

Boolean（布尔型）

仅仅包含两个值: True和False

Number（数字型）、Integer（整型）和Real（实型）

如: 1, 2, 1.0, 1.1, 3.14, -1.56

Number类可进一步分为Integer（整数型）和Real（实数型）。

Text（文本型）和 String（字符串型）

如: "This is a text"。 请注意引号为英文引号，以后都是这样。

在目前的AppleScript中，Text和String两个类型是一致的[7]。

Date（日期型）

如: date "2009年8月30日星期日 下午12:31:34"。

此格式的具体形式由“系统偏好设置-语言与文本”的相关设置决定。

Constant（常量型）

如: yes, no, ask

这些常量可以是已经被AppleScript预定义的，也可以是用户定义的不可变变量。这种类型的数据一经确定不可更改。此外可以认为所有关键字都是常量型的数据

List（列表型）

如: {1,2,3}, {{1,2},{a,b,c},}, {1,1.9, "text"}

列表型数据由{}包裹，一个列表中可以再包含列表，形成多维列表，列表里的具体数据可以是同类型的。

Record（记录型）

如: {firstName:"iDoraemon", lastName:"Nathan"}

记录就是带有名称的列表。记录中的每一项都有名称（标识符）。我们可以认为List是每个数据都是匿名的Record。Record也可以进一步包含另一个Record。

---

[7] String类和未提及的Unicode Text类是在AppleScript 2.0时代的东西，现在Text类已经完全代替了它们，但是Apple®为了方便起见，保留了这两个类。根据《AppleScript Language Guide》2008年3月版，“ text, string, and Unicode text will all compare as equal”，三者完全一致。

此例中，包含两个Text型数据 "iDoraemon"和 "Nathan"，它们的标识符分别是firstName和lastName。通过of关键字可以得到想要的数据:

----------代码----------

```
firstName of {firstName:"iDoraemon", lastName:"Nathan"}
```

上面这行代码返回 "iDoraemon"。

确定数据类型-Class of

要确定一个数据到底是什么类型的，使用Class of语句

----------代码----------

```
class of "string"
```

此代码得到结果text。

第四节 强制数据类型转换

有时默认的数据类型并不是我们想要的，比如"1"的数据类型为Text，而我们想要的是Integer；另一个例子是命令要求的参数类型与实际类型不一致时，我们就需要强制类型转换（Coercion）。

as关键字

用as关键字即可实现强制类型转换。用法：数据 as 类型

请注意，强制类型转换有时会丢失精度。

----------代码----------

```
--文本类型转数字类型
"1.99" as real          --得到Real类型的1.99，而原来的数据是Text（因为带有引号）。
"1.99" as integer       --得到Integer类型的2，精度丢失！
"1" as real             --得到Real类型的1.0，自动提升精度！

--转换成List类型
"text" as list          --得到{"text"}
1.99 as list            --得到{1.99}
{a:1, b:2} as list      --得到{1, 2}，精度丢失（标识符丢失）！

--错误举例
"text2" as number       --错误！Text中包含了无法转换成数字的字符。
1 as record             --错误！无法获得标识符。
{1, 2} as record        --错误！无法获得标识符。
```

第五节 运算符[8]

数学运算符（常见七种）

运算符	含义	举例	运算符	含义	举例
+	数学+	1+1 返回2	^	指数运算a^b即a^b	3^2 返回9.0
-	数学-	2-1 返回1	div	整除，舍弃余数	5 div 2 返回2
*	数学 ×	2*3 返回6	mod	除法取余数	5 mod 2 返回1
/或 ÷	数学 ÷	3/2返回1.5			

[8] 本节中特殊符号通过按住Option键加符号来输入。如÷用option和/组合键输入，≥用option和>输入。

请注意：/和^两个运算结果在任何情况下均为Real类型的数值。div和mod运算符在任何情况下结果都为Integer。其他运算符根据输入的数值，结果可以为Real也可为Integer。

这里给编程达人一个提醒（请初学者忽略），AppleScript对除法的运算过程和代数学一样，不会出现其他编程语言（如C和Java）中3/2=1的情况[9]，AppleScript更贴近实际生活，请不要和其他编程语言混淆。

比较运算符

下表中所有运算符均返回Boolean（布尔型）数值。同一个格子的运算符中是同一功能的不同表达方式。从表格中不难看出，AppleScript在设计的时候的确很用心，语言很接近英语。

运算符	含义	运算符	含义
= equals is equal [is] equal to	比较两端是否相同 支持的数据类型非常多。	≠ is not isn’t isn’t equal [to] is not equal [to] doesn’t equal does not equal	比较两段是否不同 支持的数据类型非常多。
> [is] greater than comes after is not less than or equal [to] isn’t less than or equal [to]	左边是否数值大于/文本长于/时间晚于右边 支持类型: integer, real, date, text	⩾ >= [is] greater than or equal is not less than isn’t less than does not come before doesn’t come before	左边是否数值大于/文本长于/时间晚于或等于右边 支持类型: integer, real, date, text
< [is] less than comes before is not greater than or equal [to] isn’t greater than or equal [to]	左边是否数值小于/文本短于/时间早于右边 支持类型: integer, real, date, text	⩽ <= [is] less than or equal is not greater than isn’t greater than does not come after doesn’t come after	左边是否数值小于/文本短于/时间早于或者等于右边 支持类型: integer, real, date, text
start[s] with begin[s] with	前者是否以后者为开始 支持text和list	end[s] with	前者是否以后者为结束 支持text和list

[9] 在Java和C中，两个整型变量相除得到的还是整数，这就产生了3/2=1的情况。而在AppleScript中，即使是整型变量相除，仍会得到小数。如：(3 as integer) / (2 as integer)结果仍然是1.5。

运算符	含义	运算符	含义
contain[s]	前者中是否包含后者这一元素 支持text，list和record类型	does not contain doesn’ t contain	前者中是否不含后者这一元素 支持text，list和record类型
is in is contained by	后者中是否包含前者这一元素 支持text，list和record类型	is not in is not contained by isn’ t contained by	后者中是否不含前者这一元素 支持text，list和record类型

比较运算符表达方式非常多样，读者应该根据自己的实际情况选择一种或几种。记得要亲自试一试！

逻辑运算符

and（逻辑与），or（逻辑或）和not（逻辑非）。
三个对参与运算的数据要求均为Boolean型，not为单目运算符。

&运算符

简单来说这个是合并运算符。支持任何类型数据。
记住“&”运算符的三个规则:

- “&”左边的数据类型为Text（文本型）时，结果为Text；存在报错可能
- “&”左边的数据类型为Record（记录型）时，结果为Record；存在报错可能
- “&”左边的数据类型为其他时，结果为List类型

----------代码----------

```
"Text" & 1                  --结果：“Text1” (Text类型)
1 & "Text"                  --结果：{1, "Text"} （List类型）

{name:"a"} & "b"            --结果：{name:"a", «class ktxt»:"b"} (Record类型，第二个元素匿名)
3 & {name:"a"}              --结果：{3, "a"} (List类型，且丢失标识符)

{1, 2, 3} & {4, 5, 6}       --结果：{1,2,3,4,5,6} （List类型-一维的）
{1, 2, 3} & 4 & 5 & 6       --结果：同上
{a:1, b:2} & {c:3}          --结果：{a:1, b:2, c:3} （Record类型-一维的）

--错误举例
"Text" & {name:"a"}         --错误！无法将Record类型数据转为文本
{name:"a"} & 3              --错误！无法将Integer转换为Record
```

还是那句话，请亲手试试看“&”，这个运算符非常重要。

第六节 提取对象中的元素

提取字符串中的字母或单词

从简单的开始：提取字符串中全部字母（单个Unicode字符），结果为List类型

----------代码----------

```
every character of "一个字符串"      --结果：{"一", "个", "字", "符", "串"}每个字符
characters of "A String"            --结果：{"A", " ", "S", "t", "r", "i", "n", "g"}每个字符
```

请注意：every character of 和 characters of 两个命令的功能完全一致。

类似地，提取字符串中全部单词[10]，结果为List类型

```
----------------代码----------------
words of "A string"                 --结果：{"A", "string"}
every word of "我的名字叫张三"    --结果：{"我", "的", "名字", "叫", "张三"}
------------------------------------
```

其中：every word of 和 words of 功能也完全一致

下面是进阶一点的：提取指定未知的几个字符或单词，结果为List类型；提取特定位置的一个字符或单词，结果为Text。

```
----------------代码----------------
characters 3 through 5 of "A String"        --结果：{"S", "t", "r"}              （几个字符-List）
words 3 through 5 of "我的名字叫张三"       --结果：{"名字", "叫", "张三"}       （几个单词-List）

word 2 of "This is a text"                  --结果："is"                         （一个单词-Text）
character 3 of "This is a text"             --结果："i"                          （一个字符-Text）

ninth character of "This is a text"         --结果："a"，说明AppleScript还能认识序数词！
------------------------------------
```

特别注意：

- 第一个字符的位置是1，而不是0！
- “through”可以缩写为“thru”
- 不推荐用序数词方法获得一个单词或字符
- 不推荐对于中文引用“word”相关语句

提取Finder文件列表

用提取字符串中字母的思想，现在来提取Finder文件列表。直接看代码：

```
----------------代码----------------
tell application "Finder"
    every file of desktop       --获得桌面上所有文件（List类型），其内容很详细
    files of desktop            --同上

    every folder of desktop     --获得桌面上所有文件夹（List类型），其内容很详细
    folders of desktop          --同上

    name of every file of desktop    --结果：获得桌面上所有文件名称（List类型）
end tell
------------------------------------
```

说明：2-5行代码返回的List中每一项都非常详细，形如“document file "AppleScript.pages" of desktop of application "Finder"”；相反倒数第二行代码仅仅包含Text类型的名称列表。

继续前进！现在提取符合指定条件的Finder文件列表

```
----------------代码----------------
tell application "Finder"
    every file of desktop whose name begins with "a"    --仅获得文件名以a开头的文件列表
    every file of desktop where its name contains "a"    --仅获得文件名包含a的文件列表
end tell
------------------------------------
```

[10] 对中文竟然也有效，它的断词依据系统词典（个人猜测可能是输入法的词库），看起来断词非常符合中文语言规范。不过建议不要依赖这个功能。

这段代码中，使用到了上一节中介绍的比较运算符，代码中的“begins with” “contains”可以用其他比较运算符代替。

新知识：whose和where its：两者功能基本一致，用于限定条件

第七节 添加注释和括号

所谓注释就是写在代码里却不会被执行的文字，通常用于辅助说明代码功能。学过编程的人都听说过一句号“没有注释的代码永远不会是优秀的代码”。对于注释的重要性想必大家都能理解。

行尾注释（End-of -line comment）

以“--”（两个连字符，不含引号）开头[11]；--之后的内容全部为注释，而之前的内容仍然为会被执行的代码，仅对当前一行有效！在前面的代码示例中，我就采用了这种注释方法。

块注释（Block comment）

以“(*”开头，并以“*)”结尾（括号加上星号，均不含引号）；包含在里面的所有文字均视为注释，可以跨行，也可以在一行中间位置。

给代码添加括号

这里只是给大家一个提醒，由于代码执行的优先级问题，可能导致代码未按照预期执行，那么此时加上括号来强制控制运算优先级将很必要。更重要的是，对于复杂的代码的，不管默认优先级如何，总加上括号来明确是个好习惯！

第八节 代码缩写

为了节省时间，将冗长的关键字缩写为几个字母也是非常有用的。此处举几例:

- application 简写为 app（编译时会补全）
- end tell/repeat/try 简写为 end（编译时会补全）
- through 简写为 thru
- if语句 可省略 then（将在下一章节介绍）

[11] 行尾注释还有另一种方法：以“#”开始（井号，不含引号），这种注释在2.0版AppleScript才出现，目的是为了Unix的一些目的，比较复杂，在此不赘述同时也不推荐此方法。

第四章预告 寻找支持AppleScript的应用程序和帮助

本章将介绍如何使用AppleScript字典（Dictionary）来找到支持AppleScript的应用程序和获得参考文件（Documentation）。

此处只是预告可能的内容，制作时将根据我的理解进行调整以达到最佳效果。

----------代码----------

----------代码----------

''''

" "