第一章 你的第一个AppleScript程序

你将学会：

如何使用Script Editor，输入代码以及如何运行你的程序。同时你还会学会如何通过display dialog命令来写一个简单的对话框。

使用Script Editor

OS X系统里都自带了AppleScript的编辑器，你可以通过右上角的Spotlight搜索AppleScript来打开它。

打开之后你将看到编辑器的界面（你可以很庆幸，虽然XCode不是中文的，但是AppleScript的脚本编辑器是中文的）。

好了，我们现在开始演示一下怎么使用这个家伙（另一个好消息，这玩意比C++简单多了）

你应该在编辑器窗口输入这个东西

```
100 * pi
```

你的编辑器看上去会是这个样子的。（记得把中文输入法关了，星号可不是×号）

编译和运行你的程序

在你输入完成之后，点击那个运行按钮。

然后你就会看到下方的结果框内出现了一行数字，这个就是运行的结果。

工作原理

当你打开AppleScript脚本编辑器的时候，实际上系统已经自动给你创建了一个叫做Untitled的窗口。在这个窗口里你可以输入你的AppleScript脚本。在刚才这个最简单的例子里，一个程序可以就是一个简单的计算式子（100×π）。

下面我再解释一下你的脚本程序运行的过程。从你点了运行开始，脚本编辑器开始分析你的代码确保他们都是合乎规范的，然后将他们编译成另外一种更适合执行的格式（类似于Java中间码的一种处于机器码和脚本代码之间的东西），之后通过特定的格式来执行你的代码，并且改变你的单词，操作符和表达式的颜色，最后执行你的程序并且在结果中显示你的程序的结果。

存储程序

既然你已经写完了你的第一个程序，你应该试试去存储他。文件->存储或者Command+S也可以。

需要注意的是，当你存储你的脚本文件的时候，编辑器会尝试在保存前先编译它，因为整个原因，你必须保证你的文件没有任何错误才能够存储。当你保存成功之后，下次再修改它你只需要在Finder里面找到并双击你存储的文件就行了。

扩展你的第一个程序

你并没有完成你的第一个程序，在这里，你会学习一下AppleScript的脚本编辑器是如何报告错误的，之后你会学习到如何在你的程序里添加描述以及如何使用命令建立一个对话框。

错误报告

首先，你得把你的脚本写错了才行，我们可以尝试一下，把pi写成π。点击运行，π被高亮了，而且弹出了错误内容。

而当你把pi改为p的时候，又会报另一个错误（考虑到可能有人没有编程基础，提前说一下变量的问题，本来不想说的）。

这里就出现了一个概念，变量，最简单的说法就是一个标记，它可以存储一些数据，而这个标记在没有确定存储什么之前，是不能使用的。（我还是不太说得明白……果然对于我来说，定义这个词就是最小的概念了……）

添加描述

这里你得注意以下，这个是描述，并不是注释。
添加描述是在代码编辑框的下方，点击描述后就可以写入了。
再次存储之后，这个描述就存储进你的脚本文件了。

显示一个对话框

如果你用过windows API 你会知道建立一个窗口需要多少代码（我一直记得大一时写的那个快10k行的那个API+DirectX的打砖块……），如果你用过Qt，你会知道相比windowsAPI这会简单不少（实际上是代码由IDE制作了，但是看上去还是比较优雅的），而在AppleScript中，只需要两个单词display dialog。

好了，让我们开始写我们的第二个程序

```
display dialog "AppleScript 中文教程 BY www.cselife.info"
```

当然，你可以试试替换一下文字，例如和刚才的第一个程序结合起来


```
display dialog pi
```

试试这个输出什么。
当你点击确定之后，你会看到在结果栏里返回了一些东西

{button returned:"好"}

如果你点击的是取消，那将会是

error "用户已取消。" number -128

从这里你就可以了解了，建立一个对话框不仅可以输出，还可以进行交互，因为系统可以返回来用户的选择结果。

理解对话框命令中的按钮参数

还记的上面那个例子么？我们可以使用buttons参数来去掉或增加一些按钮。

现在我们修改一下上面的代码

display dialog "AppleScript 中文教程 BY www.cselife.info" buttons {"Slain.DEV"}

这样就将对话框的按钮修改成了图中的样式

因为修改了默认的按钮，所以取消按钮就消失了，只剩下我们做的这个。

现在让我们分析一下我们添加的代码:

首先是关键字buttons 然后是一对大括号，{}在AppleScript中有特殊的意思，它代表着一种特殊意义，列表（list），里面用引号引起来的单词相当于这个list容器中的一个元素，当然你可以放进去n个元素，所以你可以试着修改一下代码，让他有更多的按钮。

display dialog "AppleScript 中文教程 BY www.cselife.info" buttons {"Slain.DEV", "www.cselife.info"}

运行结果是这样的。

然后我将稍微讲解一下list这个东西（呃，就先叫东西吧，我认为类似于C中数组，第六章时候再详解），你可以用它存储一些数据，比如说一组数或者一些单词，甚至混着存储也是可以的（如果你没有编程经验，或许你会把这当成理所当然的，但是对C程序员来说可能会觉得世道乱了.....），以下的代码都是正确的

```
{"www", "cselife", "info"}
{1, 2, 3, 4, 5}
{"cselife.info", "1", 123, "Slain.dev"}
```

设置默认按钮

你需要用到的是default button这个单词，下面这两行代码都将Slain.DEV设为默认按钮。

```
display dialog "AppleScript 中文教程 BY www.cselife.info" buttons
  {"Slain.DEV", "www.cselife.info"} default button 1
display dialog "AppleScript 中文教程 BY www.cselife.info" buttons
  {"Slain.DEV", "www.cselife.info"} default button "Slain.DEV"
```

上面的代码看上去好像是四行，其实是两行代码，脚本编辑器会自动转行，所以你不需要去手动输入回车来打断一行代码，除非是两条语句才可以。（如果你必须要手动断行一条语句，你需要输入option+return而不是return）

给你的对话框添加图标

```
display dialog "AppleScript 中文教程 BY www.cselife.info" buttons
  {"Slain.DEV", "www.cselife.info"} default button "Slain.DEV" with icon
  note
```

使用with icon，你可以添加图标，内建的有 note，caution和stop。

这是以上三种的图标样式。

添加注释

分为两种注释方式，单行注释和多行注释，单行注释由两个减号构成，而多行注释则是使用(**)框起来。好的，看一下例子。

```
--AppleScript 中文教程
(*
	website:http://www.cselife.info
	author:slain.dev
	Aug 2010
 *)
display dialog "AppleScript 中文教程 BY www.cselife.info" buttons
	{"Slain.DEV", "www.cselife.info"} default button "Slain.DEV" with icon
	note
```

利用这个你可以给你的代码添加注释，使它更容易理解，更重要的是，让你在几个月甚至几年之后，忘干净了AppleScript的你也能大概看出来你究竟写了什么。

第二章 变量，类和表达式

你将学会：

AppleScript语言的基本概念，构成变量名、算术表达式的规则，以及使用AppleScript内建类的方法。一个与众不同的地方是AppleScripte内建日期类，可以让你很轻松地让你的程序在指定时间运行。

使用变量

变量允许你在程序运行期间利用一个比较方便的名字来存储数据。你可以使用st或者copy命令来给变量赋值，这里我只讲解set命令。

变量名和set命令

set命令的标准格式应该是这样的

```
set variable to value
```

如果你执行这个语句，那么将会把value存储到variable里面。下面我们举一个实例。

```
set variable to 1
```

这句话将1存储到了variable里面。（类似于其他编程语言的variable=1）

变量命名必须以一个字母（大小写都可以）或者一个下划线（_）开始，变量名可以使用任何字符，数字，或者下划线构成，你不可以使用一些有特殊意义的单词（就我们见过的来说，set、to都是不能使用的，因为它们本身在语法里就有特殊的意义了，这种东西在其他语言里叫做系统保留关键字），当然你也不能用应用程序字典里已经使用的变量名（关于这个，将在后期讲）。

AppleScript里面有各种各样的变量、类、命令，你或许不知道怎样才能判断你想用的单词是否是系统保留的，你可以在写下这个单词之后点击 编译 按钮，然后观察你刚写的那个单词的颜色，如果是蓝色，它是系统保留的;如果是绿色，你可以安全地使用它来作为变量名。

以下的几种是合法的变量名

```
last_name
lastName
_erroMessage
textFile101
```

以下几种是不合法的变量名

```
101textFile --不能以数字开头
set --不能使用命令名
last name --不能有空格
myAccount$ --不能有特殊符号
last-name --不能使用连字符
```

当你选择你的变量名的时候，记得别太懒了。选择那种能够反映你存储的数据的名字（例如使用ErroMsgStr而不是a），这样的话你或者其他人可以比较清楚地看出来你的代码到底在干什么（当然，注释也是很重要的一部分），所以如果你要统计文件数量的话，请这么写

```
set fileCount to 1
```

而不是

```
set i to 1
```

第二种写法，或者说变量名i更常见于编程语言的循环结构计数变量而不是一个存储数据的变量。

如果你学习过C++或者Java，你可能发现在这里变量不需要声明就可以使用，当你使用的时候只需要想出来变量名，然后给他赋一个值就可以了。同样的，这里的变量没有类型的限制，你可以用它来任意存储各种数据（惟一的代价是比那些语言慢，但是脚本语言的短小和方便可以在一定程度上弥补速度的问题）。

获取变量值和get命令

现在你已经在变量中存除了数据，你可以使用get命令来获取变量的值

```
set fileCount to 1
get fileCount
```

运行的结果是1

我们再写一个更复杂的例子

```
set a to 1
set b to 2
set sum to A + b
```

运行的结果是3，如果你不能理解，那么你可以把set去掉，然后把to都看成=就好了。说实话忍住不把to写成=需要很大的毅力阿。

你可能注意到了，上面的最后一行中有大写的A，不过不要紧，在AppleScript中，大小写是不敏感的，所以A和a是等价的，但是在你编译的时候，编辑器会自动将他们改成同样的样式，以第一个出现的样式为准（a第一次出现是第一行中的a，所以第三行的A被改为了a）。

使用特殊的变量名

还记得上文中几个不合法的变量名么？如果你念念不舍地想用这些名字的话，告诉你个好消息，AppleScript还是支持的，使用的方法就是在你的变量名之间用上竖线间隔符(|)，这样任何字符都可以使用了（中文也是可以的！）。

需要注意的是，如果你用ll将你的变量包起来，那么大小写不敏感还是适用的，例如abc和lAbcl，这两者是相同的（AppleScript 2.1.2是这样，如果是1.x版本，那么是大小写敏感的）。

特殊的result变量

首先，来看一个两行代码的脚本

```
set i to 1
set abc to "www.cselife.info"
result
```

显示的结果是 “www.cselife.info”，看上去貌似和不写result差不多，那是因为系统默认输出了result。

这里需要讲解一下原理可能才更容易理解一些。result是内建的一个变量，每一行语句的运行结果在下一行运行之前都存储在result中，你可以理解为result中始终存储着上一行语句的运行结果，这就不难理解最终输出result显示的内容了。

当然除了输出最终结果之外，result还可以当中间变量使用

```
set i to 1
set abc to result + 12
result
```

这段代码的运行结果是13，请试着分析一下。

更多的命令和语句

命令

到目前我们已经使用过的命令有

display dialog

set

get

回想一下，这些命令有的有可选的参数可以使用来控制命令，例如display dialog可以使用buttons参数来修改对话框的按钮。

语句

一条语句就是单独的一行代码，其中包含着命令以及表达式。这一行可能是物理意义上的许多行（参看第一章那几条很长的代码）。

复合语句

在第三章中你将学习到复合语句，这种语句可能由许多行组成，并且使用end作结尾，来看一个判断语句的例子（现在你不需要理解，到第三章会详细讲解的）。

```
if (count) is equal to 100 then
    display dialog "count equal 100"
end if
```

这种if语句包括着一个display dialog命令，并且由最后一行的end if来结束（你可以只在最后一行输入end，编辑器会在编译的时候自动按照类型帮你添加完整，在这里就是添加成end if）。

log命令和事件日志

在刚才的例子中你已经看到了result是多么有用的东西，但是你肯定不会满足于只是看到最后的结果，无论是测试还是查错，你都需要知道中间过程中一些变量的值，这时你就需要log命令来帮忙了。

看一下下面的一段代码。

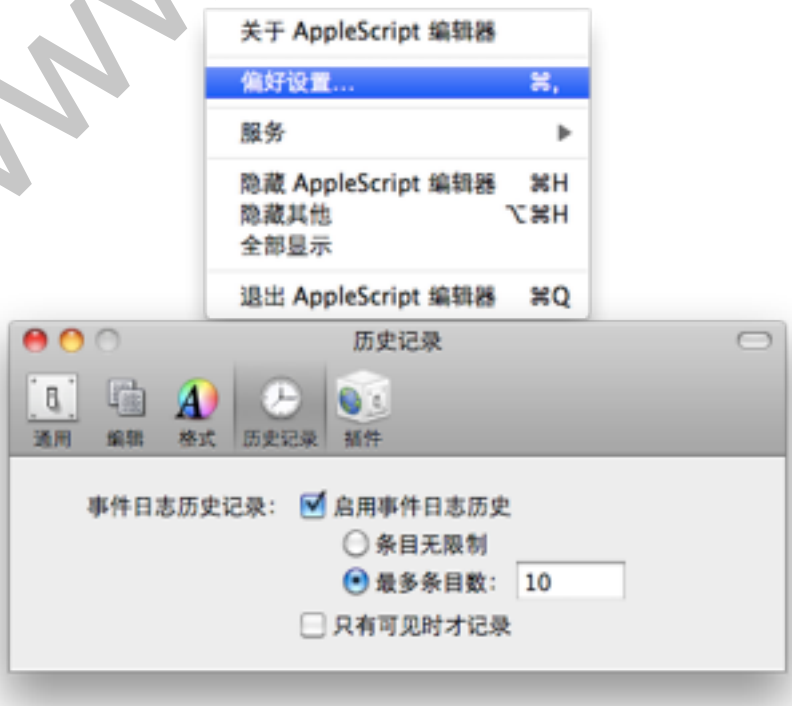
```
-- Illustrate the log command and the Event Log
log "Put this into the log."
log "And then put this in as well."
log 100 -- Log an integer
log 100 * pi -- And the result of an expression
set x to 100
log x -- log the value of a variable
```

看一下这个的输出。

在事件日志中里面出现了很多(**)括起来的文字，这就是我们使用log输出的。在这个例子中我们尝试了输出字符串，数字以及变量的值，你还可以尝试一下输出result，看看是什么结果。

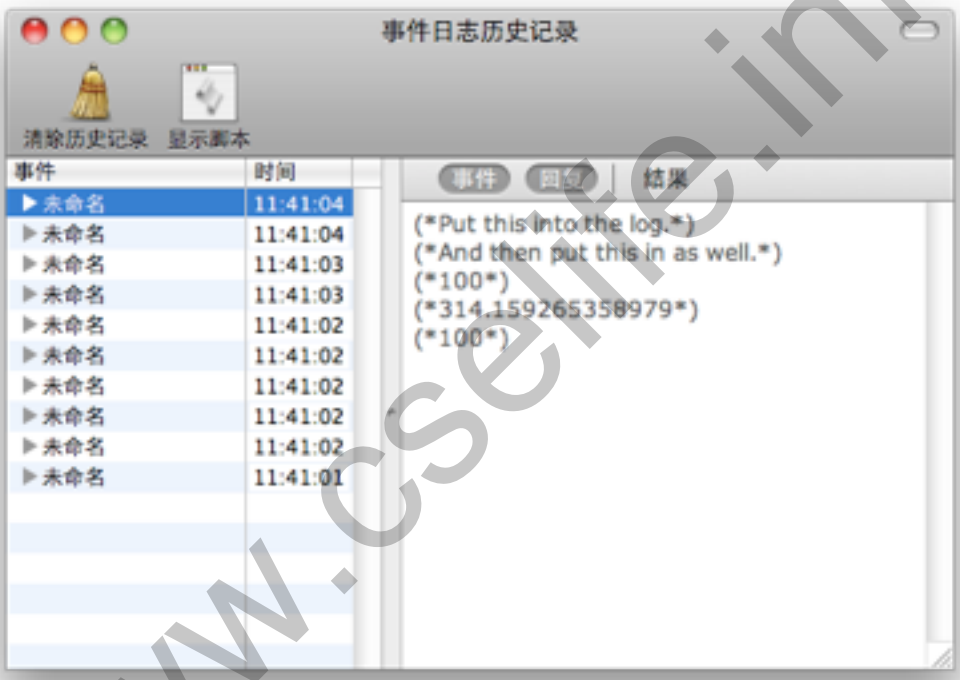
使用历史记录

除了在程序中显示事件日志之外，你还可以将他们存储到历史记录里面去，要打开历史记录工具，你需要点击AppleScript编辑器->偏好设置



然后单击 历史记录，勾选启用事件日志历史，去掉只有可见时才记录。
然后选择窗口->事件日志系统记录

你就可以看到刚才我们输出的日志了。



基本数据类型

AppleScript拥有一些内建的支持数据类型，这些可以让你很容易地处理数字、真假（布尔值）、字符串和日期。很多这些类（Class）将会在后续的章节中仔细描述，List和Records将会在第六章详细描述。

这里需要注意的是，还记的我们怎么使用变量么？不用申请，随用随赋值（这点大多数脚本语言都是这样的），当你给一个变量赋值一个证书或者字符串的时候，它都会自动转换成这种类的。

字符串类

在第一章，我们已经知道了怎么是字符串，使用双引号来引起来对吧。

```
"www.cselife.info 软件人生"
```

然后我们也学习了怎么把字符串赋值给变量。

```
set myString to "www.cselife.info 软件人生"
```

这句代码将字符串保存在了myString里面，这样你就可以使用另一种方式来使用这个字符串。

```
set myString to "www.cselife.info 软件人生"
display dialog myString
```

这样子就相当于你在display dialog后面输入了"www.cselife.info 软件人生"。

整数类

一个整数就是没有小数的数，如果你需要统计一下某个文件夹里面文件的数量，希你可以使用一个整数。

这个整数前面可以包括一个减号（-）当然也可以有一个加号（+），但是加号会在你的程序编译的时候自动被删除掉（你连留着看看的权利都木了）。

整数的范围为±(2^{29}-1）也就是±536870911（看看就得了，一般用户不需要考虑溢出的，况且……脚本语言也需要考虑基础数据类型的溢出么？）。

小数和双浮点小数

当你使用某些软件的时候，你会看到小数或者双浮点小数的类型，技术上说，AppleScript使用32位来存储整数值，一个小数是以16位的形式存储的，然而一个64位双浮点数是使用64位存储的。AppleScript将小数当作整数来处理，将双浮点数当作实数。

实数类

所有的实数都是以小数的形式存储的，即使它是一个0，那也是一堆的零。

如果你写下一个数，前面有着一个符号（不写或者减号），然后后面有好多零或者好多零之后跟着几个数字（无论怎样，实数至少得有一个0-9之内的数字出现，无论是在小数点之前还是之后）。

还有一个好消息，你可以使用科学记数法，譬如说你可以写1.23e3，在编译时候会自动转换成1230，而如果你写0.001，编译器会自动给你转换成1E-3，也就是说实数小数部分长于三位（0.001）的数字，AppleScript编译器会默认使用科学记数法（或许解释的不太清楚，大家可以尝试一下，不过这些不会影响到编程，输入的时候选择你舒服的方式就可以了，毕竟编辑器会自动修改的）。

布尔类

布尔（Boolean）的值只能是两种，真或者假（true & false）。当你需要在你的程序里表示打开/关闭，真/假的时候，你需要使用这种数据类型。

当你在AppleScript里面比较两个数字的时候，编辑器会返回给你一个为真或者假的布尔值。

下面我们再来看一个例子。

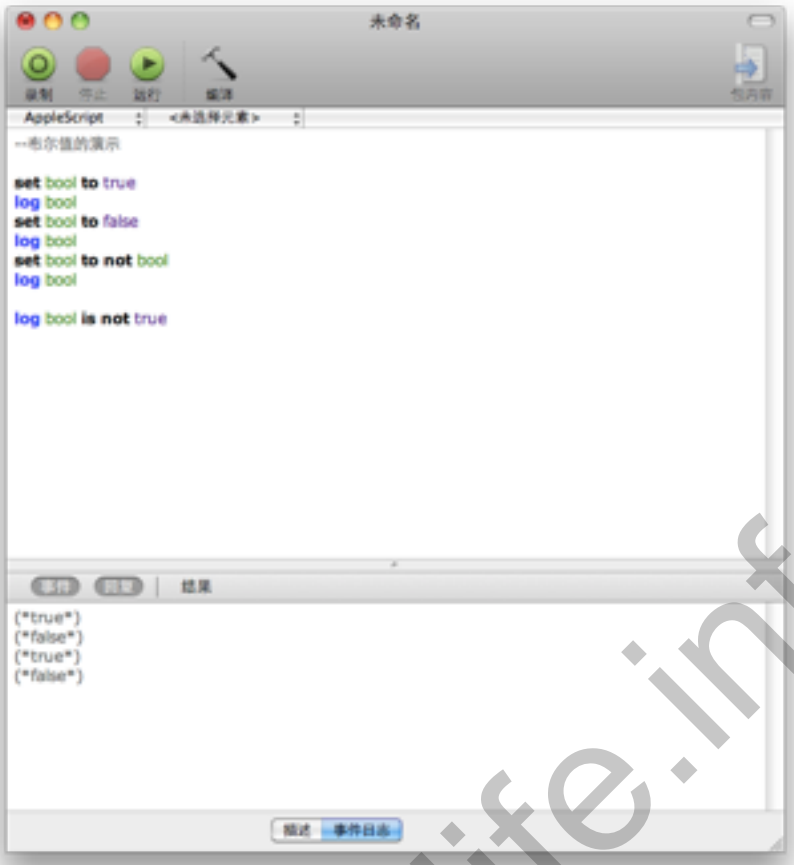
```
--布尔值的演示

set bool to true
log bool
set bool to false
log bool
set bool to not bool
log bool
log bool is not true
```

我们再看一下它的事件log。

```
true
false
true
false
```

下面我们来解释一下这个log结果。



首先我们给bool赋值true，所以log bool为true。然后我们给bool赋值false，所以log bool为false。再后来我们，恩，你看英语就可以看出来，我们使bool不等于bool（这里的bool和左侧的bool并不等价，这里的bool相当于提取出来的在这句脚本运行之前bool变量里面存储的值），所以bool也就从false变成了true，再然后我们写了一个判断句子，bool不等于true，这句话是false的，所以输出false。

class命令

上面讲了那么多类型，但是你会发现我们一个都没有主动用（我们给变量赋值，然后AppleScript自动给它选择类型），那么这些类型有什么用呢？现在我们可以用class命令来获取一个变量是什么类型的。

class命令在Debug（除错或者叫调试）程序时候是很有用的，你可以检查一个变量是否存储了正确的变量类型。

你可以这样使用class命令。

```
set a to "www.cselife.info"
set b to 25

log class of a
log class in b
```

看到了，你可以使用class of或者in来获取一个变量的类型，随你喜欢呵。

基础数据类型转换

AppleScript允许你随便将你的变量转换类型，因此，你可能需要随时知道你在一个变量里面究竟存了什么数据，是整数、实数、字符串还是一个时间。不过当你使用某个函数或者表达式的时候，它或许就已经自动给你把类型转换了。

例如我们写过的display dialog命令，如果我们这么使用

```
display dialog 100
```

实际上经过了这样的几个步骤，首先编译器将100转化为字符串，然后再运行

```
display dialog "100"
```

但是，这只是某些情况，更多的情况下，你需要自己进行类型转换，手动地进行上文的转换类型过程。

使用as操作符来转换数据类型

你可以使用as操作符来转换数据类型，as操作符的格式是这样的。

变量值 as 类型

在AppleScript术语中，将一个变量转换成另一种类型被称作coerced（如果你看apple reference或者外国的script comment，可能会用到）。

使用程序来测试新学到的操作命令永远是一个好主意，所以请你试试下面的代码吧。

```
set r to 123.45556
```


```
set int to 333
set str to "www.cselife.info"

log class of r
log class of int
```

```
set r to r as integer

log class of r
```

输出结果是real、integer、integer，我们可以看到，我们将real转换为integer，然后保存到r中，最后的log我们可以看出，r变成了integer类型。

再来看一段代码

```
set int to 123
display dialog (int as string)
```

这样我们就手动进行了AppleScript内部进行的类型转换，你会发现和前面的代码效果是一样的。

数字类的转换

你可以尝试一下把刚才代码里面的str转换为real或者integer，你会获得一个错误，你可以试试别的，我这里提供一种可以将字符串转换为数字的例子代码。

```
set str to "0.123"
log str as real

set str to " 0.1e3 "
log str as real
```

看出来一些了没？如果你的字符串里面是纯数字，并且整个数字之间没有空格之类的分割（你可以看到第二个例子里面我在整个数字的前后都放了空格，但是这是无所谓的，不过不能放在数字中间）。

除了字符串之外，数字和布尔也是可以互相转化的。true相当于1，而false相当于0（这里和C不同，AppleScript不可以把所有整数都转化为true，只有1才能转换为true）。

round命令

round命令是很有用的，使用它我们可以将实数通过四舍五入的方法转化为整数。

下面是round的几个参数。

Round参数	意义
rounding up	小数部分全部入
rounding down	小数部分全舍
rounding toward zero	向趋向零的方向入、舍
rounding to nearest	四舍五入，.5变成最近的偶数
rounding as taught in school	四舍五入

如果你没有使用参数的话，rounding to nearest 是默认参数。

如果你觉得看起来很绝望，那么看一下下面的代码可能你就明白了。

```
round 1.5                                  --2
round 1.25 rounding up                     --2
round 1.25 rounding down                   --1
round 1.25 rounding toward zero            --1
round 1.5 rounding to nearest              --2
round 1.5 rounding as taught in school     --2
```

注释中的是转换之后的整数。

数学符号和表达式

AppleScript支持一些基本的数学运算符，例如加减乘除（+ - * /），同样的还有一些比较特殊的运算符，例如说div，mod和^运算符。

div是整除运算符，例如说7除以3，大家回想一下小学数学，就会知道结果是2余1，7 div 3结果就是2。而mod就是大家熟悉的取余运算符，7 mod 3的结果就是1。^也是比较熟悉的一个运算符，2^3就是2的三次方。

有一点需要注意的是，mod运算符支持小数取余，也就是100.3 mod 0.15。

关于运算优先级，请查阅初中四则运算课本。

Date类

AppleScript的Date类是于其它类都很不同的，你可以使用变量来存储日期，计算两个日期之间过了多少天，在某个时间段内建立了多少文件，这个日期所在的星期等等。

date类的基本格式是这样的（AppleScript 2.1.2版本使用中文，1.x版本支持英文）：

AppleScript2.x版本

2.x版本的格式是这个样子的，你可以尝试输入这行代码来显示当前时间。

```
current date
```

显示结果是

```
date "2010年8月24日星期二 下午06:30:05"
```

这个就是date的基本格式，年月日（星期可以省略，自动补全） 十二小时制时间。

如果你只写到年月日，则自动为上午12：00：00。

AppleScript1.x版本

Weekday,Month day, year hh:mm:ss am/pm

参数	内容
Weekday	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
Month	January,February,March,April,May,June,July,August,September,October,November,December
day	1->31
year	年份
hh:mm:ss	小时：分钟：秒
am/pm	12小时制时写AMIPM，24小时制则忽略

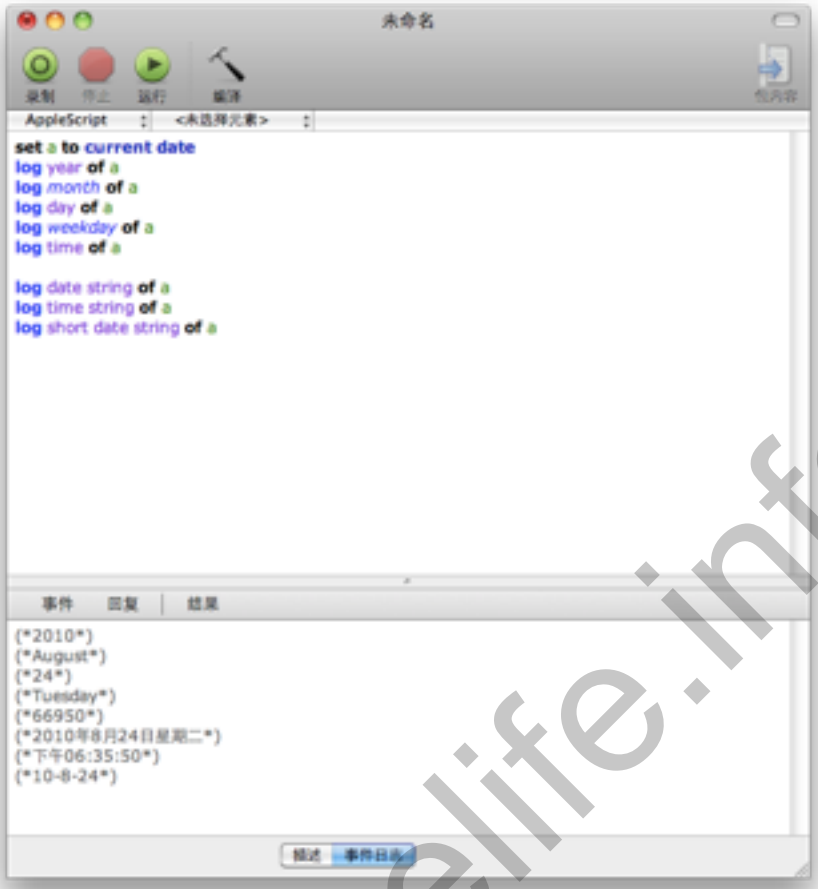
获取date类的信息

直观的话 我们来看一下代码

```
set a to current date
log year of a
log month of a
log day of a
log weekday of a
log time of a

log date string of a
log time string of a
log short date string of a
```

看一下输出




```
(*2010*)
(*August*)
(*24*)
(*Tuesday*)
(*66950*)
(*2010年8月24日星期二*)
(*下午06:35:50*)
(*10-8-24*)
```

这个代码没有改变，1.x和2.x还是一样的。最后三个获取的是字符串。

有必要解释一下的是，time返回的是从00：00：00开始的秒数，所以如果你需要使用hh:mm:ss格式时间的话，还是需要计算一下才可以的。

计算方法我写了这个，你可以使用一下变量存储上步结果，可以减少一下计算量提高速度的。

```
log (time of (current date)) div hours
log ((time of (current date)) mod hours) div minutes
log ((time of (current date)) mod hours) mod minutes
```

如果你写好了可以发email到页面右下角的地址，我会添加到书中的。

修改date类信息

除了获取信息，我们还可以修改，看一下下面的代码。

```
set a to current date
log a
--开始修改date类

set year of a to 2012
```

```
set month of a to 12
set day of a to 24
set time of a to 3 * hours + 12 * minutes + 0
log a
```

看一下log你就明白我们是怎么修改的了。(hours和minutes是内建的，直接使用就好了)。

```
(*date 2010年8月24日星期二 下午07:02:42*)
(*date 2012年12月24日星期一 上午03:12:00*)
```

很简单吧，这玩意是很easy的。

对date类进行计算操作

还是同样的，代码说话，这个部分里面没什么概念性的东西，大多是操作方法，所以仔细研究代码就可以了。

```
set a to current date
set b to date "2000年9月9日星期六 下午09:00:00"

log a
log b

set betweenDate to a - b

log betweenDate
log betweenDate div hours
log betweenDate div days
log betweenDate div weeks
log betweenDate div (days * 365)
```

让我们和结果对比一下

```
(*date 2010年8月24日星期二 下午07:15:40*)
(*date 2000年9月9日星期六 下午09:00:00*)
(*314144140*)
(*87262*)
(*3635*)
(*519*)
(*9*)
```

我们可以看到，两个日期相减的结果是两个时间点相差的秒数，通过整除算法我们就可以清楚地得到我们想要的数据了。

除此之外，我们还可以做加法，方法是给一个date类加上相应的秒数，这个大家可以实践一下，然后email给我，我可以把你的代码添加进来呵呵～

在date类的最后，让我们来写一个小东西，计算距离2012年12月21日晚上6点还有多久。

```
set curDate to current date
set doomDate to date "2012年12月21日星期五 下午06:00:00"

set btwDate to doomDate - curDate

set y to btwDate div (days * 365)
set d to (btwDate - y * days * 365) div days
set h to (btwDate - y * days * 365 - d * days) div hours
set m to (btwDate - y * days * 365 - d * days - h * hours) div minutes
set s to (btwDate - y * days * 365 - d * days - h * hours - m * minutes)
```

```
display dialog "距离末日还有" & y & "年" & d & "日" & h & ":" & m & ":" & s buttons {"www.cselife.info"}
```

第三章 做出决定

你将学会：

使用if表达式来控制程序，从用户获取输入。

if语句

AppleScript的if语句看上去是比较像BASIC的（因为使用if then - end if结构）。

if 布尔语句 then

语句

end if

如果布尔语句是true，那么将会运行结构内的语句，如果为false，那么将跳过if end if之间的语句。

```
set x to -5
if x is less than 0 then
    set x to -x
end if
```

这段的运行结果是5，如果x是大于0的，那么将会跳过set x to -x 语句，你可以log一下x，观察一下不同的结果。

单行的if语句

单行的结构是

if 布尔语句 then 语句

上面的代码修改成单行格式的话，应该是

```
if x is less than 0 then set x to -x
```

AppleScript关系操作符

还记的我们刚才怎么比较数字么？less than，我想对于学过其他编程语言的人来说，<=是更方便更简单的东西，好消息是，我们也可以在AppleScript中使用这些关系操作符。

文字	操作符	示例
is less than is not greater than or equal to comes before	<	x is less than 0 x < 0 “a” comes before “b”
is less than or equal to is not greater than does not come after	≤	x is less than or equal to 0 x ≤0
is equal to is	=	x equal to 0 x = 0
is not equal to is not	≠	x is not equal to 0 x≠0
is greater than is not less than or equal to comes after	>	x is greater than 0 x > 0

文字	操作符	示例
is greater or equal to is not less than does not come before	≥	x is greater than or equal to 0 x ≥ 0

上面的符号不好输入么？你可以点住option键，然后点><=，就可以分别得到≥≤≠了，输入还是很简单的～

说完了关系运算符，让我们来实际使用一下。

数字测试

```
log "数字测试"
set n to 100
log n < 200
log n is less than or equal to 99
log n mod 2 = 0
log n ≠ 100
log n > 99
log n ≥ 0
```

输出结果是

```
(*数字测试*)
(*true*)
(*false*)
(*true*)
(*false*)
(*true*)
(*true*)
```

需要记得的是，在这里 = 号只可以用于构造布尔表达式，不可以用于赋值。

字符串测试

```
log "字符串测试"
log "阿" comes before "阿门"
log "阿门" comes before "中国"
log "阿" comes before "cse"
log "100" > "alpha"
log "abc" = "Abc"
```

输出结果是

```
(*字符串测试*)
(*true*)
(*true*)
(*false*)
(*false*)
(*true*)
```

AppleScript的字符串比较方式是根据“查字典”方式的，简单的说，如果你查字典的时候这个字符串页数比另一个字符串在前，那么他就是小的（也就是comes before），而英文，中文，数字的排列顺序是（英文<数字<中文）。并且没有大小写敏感（最后一个的结果是true）。

下面的例子可能会让你比较迷惑，是关于字符串里的数字和真正的数字进行关系操作。

```
log "字符串测试2"
log 100 < 50
```

```
log "100" < "50"
log "100" < 50
log 100 < "50"
```

运行结果

```
(*字符串测试2*)
(*false*)
(*true*)
(*true*)
(*false*)
```

看上去比较迷惑是吧，我来解释一下。当你比较两个不同类型的数据的时候，AppleScript默认将第二个数据转换成第一个的数据格式，所以第三个比较会是true，因为1比5靠前;而最后一个是false，因为数字100大于数字50。

关于数字与字符串比较自动转换格式的一个特例是等号比较，如果你你的表达式是100=”100”,那么这个的结果是false。

日期测试

比较两个日期是很有用的，举个例子，你可能希望每个月初都进行一次系统备份，为了达到这个目的，你需要让你的程序知道在哪一天运行。你同样需要让他知道需要备份哪些文件，例如在十四天内创建的所有文件。

下面是一个日期比较的例子，我们使用current date获取当前系统时间，更加详细的请参看上一章中的相关内容。

```
log (month of (current date)) > August
log (month of (current date)) comes after December
```

输出的结果是（为了显示清楚，我勾选了事件和回复两个栏）：

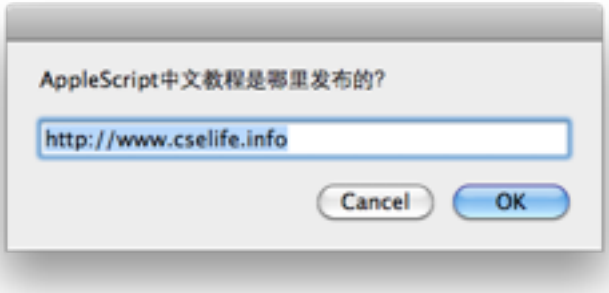
```
tell current application
    current date
        --> date "2010年9月1日星期三 下午02:47:19"
    (*true*)
    current date
        --> date "2010年9月1日星期三 下午02:47:19"
    (*false*)
end tell
```

从用户处获取数据

终于写到这里了，这是一个很重要的起点。在我看来，一个没有交互的程序是无法使用的——至少对那些不懂代码的人来说是正确的。现在我们终于迈出了交互第一步，让我们看看我们能从用户那里获得什么吧。

在第一章我们就学习了display dialog，现在我们继续使用display dialog，不过不但是显示数据，而是获取数据。如果你需要一次输入很多东西或者类似于软件使用偏好那种样子的东西的话，你需要再继续看下去，我们会在以后讲到AppleScript Studio的。

我们使用display dialog的default answer参数来让你的用户输入一段信息，让我们来看一个例子。



```
display dialog "AppleScript中文教程是哪里发布的？" default answer "http://www.cselife.info"
```

很熟悉的对话框不是么？不过多了一个输入而已。

在运行结果中我们看到了返回的文字。

```
{text returned:"http://www.cselife.info", button returned:"OK"}
```

如果你不希望使用默认内容的话，只需要将default answer后面的双引号中的东西晴空就好了（记得要保留那两个双引号，否则会报错）。

提取用户输入

只是让输入内容显示在结果中不是我们想要的，我们还需要能够将它存储在变量里，到目前为止我们获得的returned有两个，一个text一个button，我们可以通过以下的代码来获取它们。

```
text returned of result
button returned of result
```

有一点需要注意，在一个dialog下面只能用这两者之中的一个命令，否则的话是会报错的。另外的就是，如果你需要获取dialog返回的数据，必须在display dialog 的下一局立刻set一个变量to它，否则的话你是无法得到数据的，这也就解释了上边为什么只能使用一个命令的原因。

如果你需要同事获取text和button怎么办呢？text和button都是在result里面的对吧？如果只能存储一个的话，那我们存储result，不就变相把text和button都存储了么？

```
display dialog "AppleScript中文教程是哪里发布的？" default answer "http://www.cselife.info"
set dialogResult to result --存储result的结果
set dialogText to text returned of dialogResult
set dialogButton to button returned of dialogResult
```

这样你就将返回的数据存储到变量里了。

从字符串中提取数字

如果你需要用户输入一个数字，譬如说班级人数什么得，你可以将default answer设为一个数字来提醒对方（譬如说0）。当用户输入后，我们先提取text，然后进行类型转换（详细请看第二章）。

```
display dialog "你多大了～" default answer "0"
set age to text returned of result as number
```

这样之后age中存储的就是实数了（如果你想要整数，可以改number为integer）。

但是用户总不是那么可爱的，20%的代码实现功能，而80%的代码则需要去处理各种交互中的问题（具体的原话我不太记得了，不过印象最深的就是在ubuntu论坛上看贴出来的一小段gnome的代码，很干净很整齐地大篇目去处理用户的各种不规范输入）。在AppleScript中，你可以使用try语句来处理不规范的用户输入，过一会儿我们会讲到这个的。

if-else 语句

假设你写了一个单选菜单，上面有三个选项，按照我们到目前为止学过的东西，你需要写三个 if-end if模块来处理它，现在你应该尝试使用if else语句来使代码更加容易理解（如果是多选菜单，你使用3个if-end if分别处理将会比把所有排列组合都弄出来好的多）。

```
display dialog "你多大了～" default answer "0"
set age to text returned of result as number
if age ≤ 18 then
	display dialog "你好，小童鞋～" buttons {"ok"}
else
	display dialog "你好，老同志～" buttons {"ok"}
end if
```

好吧，这段代码和单选菜单没什么关系，我只是想把这章后面的代码统一一下而已。

if-else语句嵌套

延续上面那个三个选项的单选菜单的问题，如果使用if-else语句，那么最多只能处理两个对吧？所以我们需要用到了语句嵌套。

再看看我们上面给的代码，如果你输入小于等于0的数字，它依旧会说你好，小童鞋，完全没有注意到你并没有出生并且是以灵魂出窍状态按的鼠标，所以我们需要修改一下代码，使用语句嵌套来完善一下。

```
display dialog "你多大了～" default answer "0"
set age to text returned of result as number
if age ≤ 0 then
	display dialog "呃……你先回去等出生了在来好不好？"
else
	if age ≤ 18 then
		display dialog "你好，小童鞋～" buttons {"ok"}
	else
		display dialog "你好，老同志～" buttons {"ok"}
	end if
end if
```

这样子就完善了，不用担心缩进问题，编译的时候会自动帮助你排好版的（怨念为什么其他IDE木有这个功能，或许是因为每个人的习惯都不一样吧……怎么弄都容易被骂）。

if-else if 语句

和上面差不多，不多说了，直接看代码，我建议这么写，因为比上面的更加直观一点，当然还是要看具体情况的。

```
display dialog "你多大了～" default answer "0"
set age to text returned of result as number
if age ≤ 0 then
	display dialog "呃……你先回去等出生了在来好不好？"
else if age ≤ 18 then
	display dialog "你好，小童鞋～" buttons {"ok"}
else
	display dialog "你好，老同志～" buttons {"ok"}
end if
```

另外推荐这种写法还有一个原因，你只需要写一个end if。

复合关系运算

上面讲过的关系运算都是比较简单的，下面我们讲一下复合的关系运算，也就是有了与或非的关系运算。

在AppleScript中，与为and，或为or，非为not。

表达式	结果
A and B	当A和B均为真时为真
A or B	当A或B有一个为真即为真
not A	当A为真时表达式为假

需要记住的一点是，这三个的优先级是从右向左的，并且not优先级大于and和or，所以记得加上小括号来确保表达式正确。

A or B and C 等价于 A or (B and C)

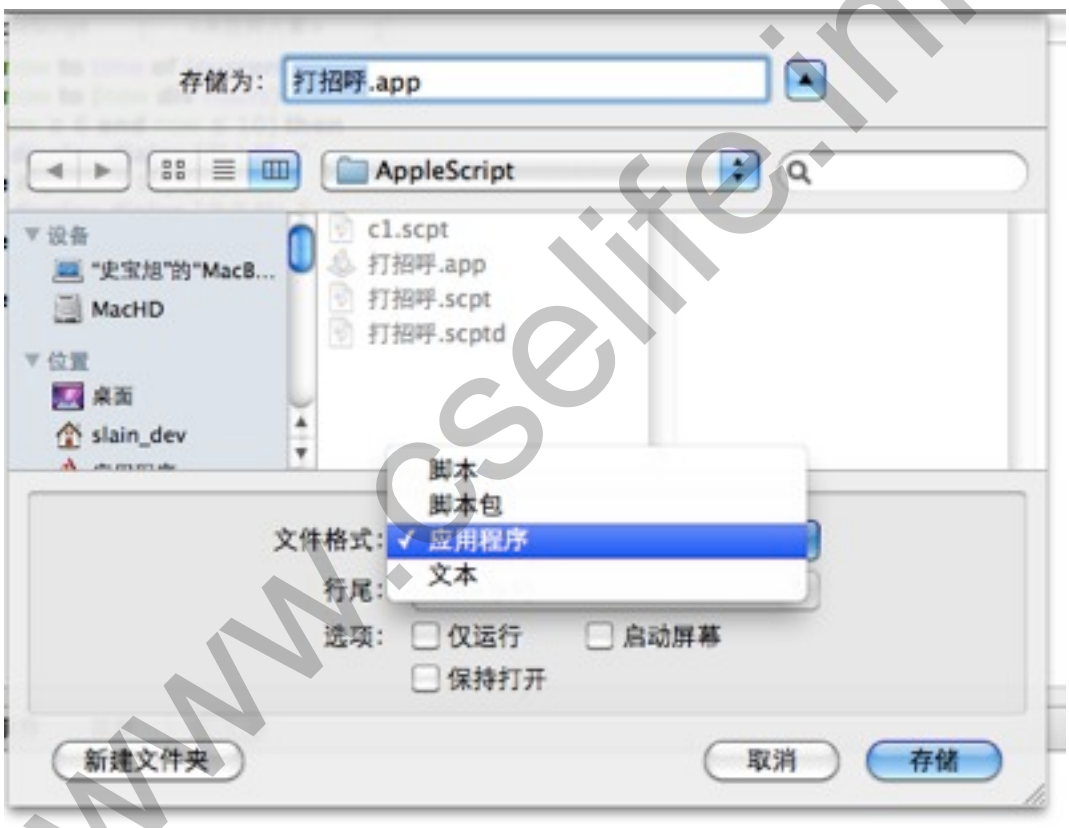
A and not B and C 等价于 A and ((not B) and C)

创建你的开机问候程序

下面我们将制作一个根据时间弹出问候语并且在开机时运行的程序。

```
set now to time of (current date)
set now to (now div hours)
if (now ≥ 6 and now ≤ 10) then
	display dialog "早上好～"
else if (now < 16) then
	display dialog "中午好！"
else if (now ≤ 22) then
	display dialog "晚上好～"
else
	display dialog "快去睡觉吧～"
end if
```

这里代码有点小技巧，后两个我都是直接写了一半范围，我想应该不用解释了吧。



将代码保存成应用程序到某个目录内之后，你将会看到一个这样的图标在你的文件夹里。

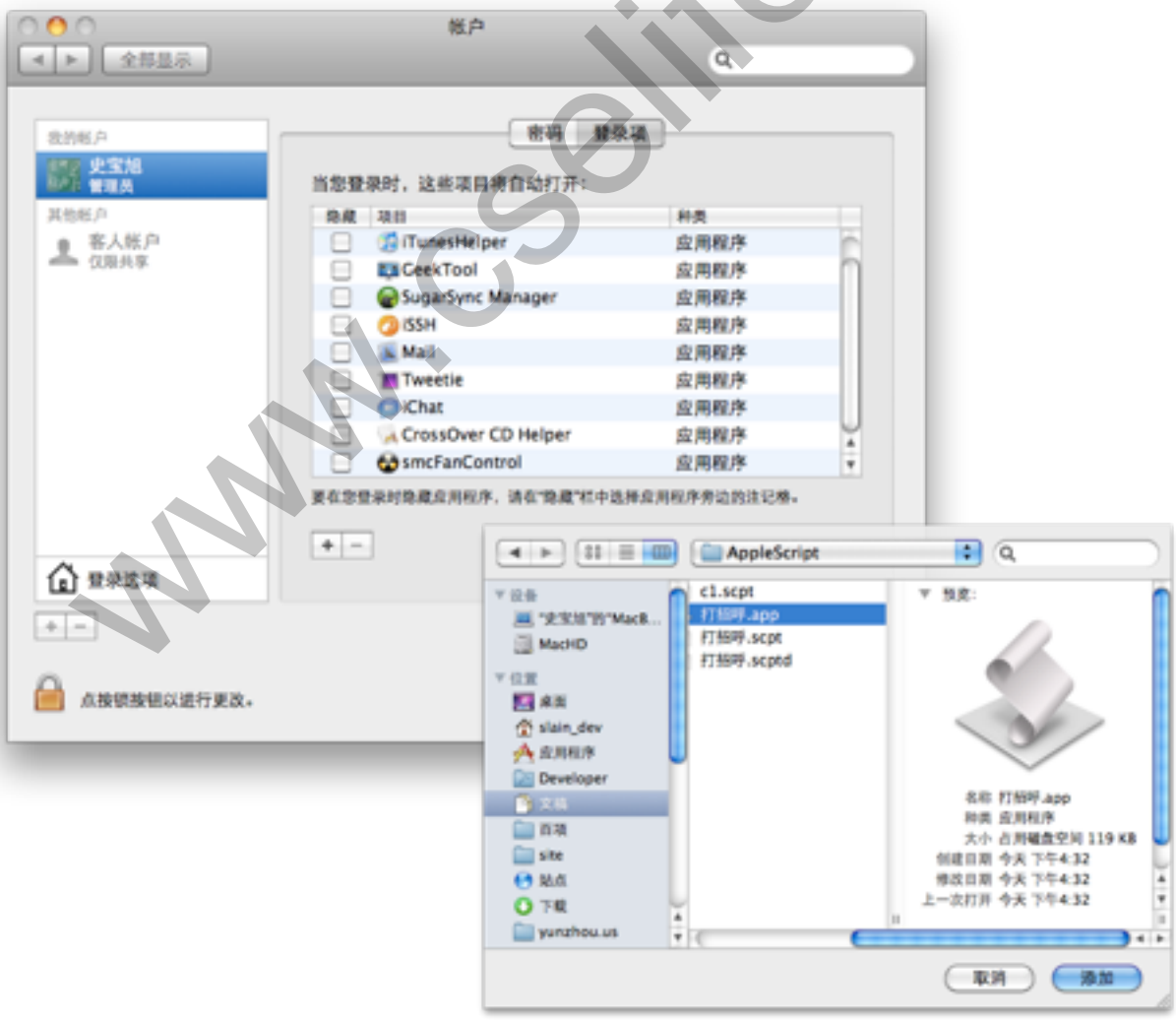
双击这个文件就自动运行了，你再也不需要点击运行或者编译了（如果你打开这个app包的话，可以在Scripts里找到你的代码文件，我不知道有没有加密的方法来输出app，如果可以，希望能够email给我添加上来）。

除此之外我们还可以让它变得更加有意思一点，譬如说，让OS X告诉你现在几点了（悲剧的是这个命令只能念英文）。

```
set now to time of (current date)
set now to (now div hours)
say "It's " & now & " o'clock"
if (now ≥ 6 and now ≤ 10) then
	display dialog "早上好～"
else if (now < 16) then
	display dialog "中午好！"
else if (now ≤ 22) then
	display dialog "晚上好～"
else
	display dialog "快去睡觉吧～"
end if
```

还有就是可以把这个东西设为开机启动。

打开你的系统偏好设置->帐户->登录项，在这里你可以点击+号，选择你的程序将它添加到启动项里。



各位用mac的同学，记得写个小东西放她（他）电脑里，等待生日那天说生日快乐吧～

使用try语句来捕捉错误

这章的最后，我们来讲这个东西，上面我们曾经提到过，如果将英文字符转换为数字将会出现错误，而这个错误将会导致我们的脚本结束运行，为了防止这个问题，我们引入try语句。

```
display dialog "你多大了～" default answer "0"
try
	set age to text returned of result as number
end try
if age ≤ 0 then
	display dialog "呃……你先回去等出生了在来好不好？"
else if age ≤ 18 then
	display dialog "你好，小童鞋～" buttons {"ok"}
else
	display dialog "你好，老同志～" buttons {"ok"}
end if
```

还是上面的那段代码，但是加上了try语句，你可以尝试一下输入一个英文字符，仍旧返回错误，可是却不是无法转换了。

这说明了try语句的作用，当其中的代码出现错误的时候将跳过并继续执行下面的代码，这也就是出现age没有定义的原因。

```
set age to missing value
display dialog "你多大了～" default answer "0"
try
	set age to text returned of result as number
end try
if age ≠ missing value then
	if age ≤ 0 then
		display dialog "呃……你先回去等出生了在来好不好？"
	else if age ≤ 18 then
		display dialog "你好，小童鞋～" buttons {"ok"}
	else
		display dialog "你好，老同志～" buttons {"ok"}
	end if
else
	display dialog "你输入的不是数字！"
end if
```

这段代码中我们在使用前给age赋予了missing value这个常量，你也可以使用其他的非数字值来代替它，只需要在下面判断的时候相应修改一下就行了。

第四章 程序循环

你将学会：

AppleScript的repeat表达式，它是重复执行一系列命令的基础，你会看到如何去写一个从1到10的猜数游戏。

第五章 字符串控制

你将学会：

如何控制字符串，从字符串中获取每个字符或者单词，获取字符串长度，比较字符串。