AppleScript for Programmers

Christopher Nebel
AppleScript Technical Lead

Overview

- The Basics
- Syntax Overview
  - What's (mostly) the same as other languages
- What's Different
  - Why it's valuable
  - How to keep it from biting you

The Basics

The Basics

- Bytecode interpreted
- Garbage collected
- Dynamic typing
- Static scoping
- Variable declarations not required

The Basics

- Line break is a statement terminator
  - Use “¬” to continue across lines
- Blocks end with **end** *block*
  - E.g., **if … end if**
  - Just type “end”, AppleScript will fill in the rest.
- Case-conforming
  - Identifiers forced to match first occurrence

The Basics

- “--” comments to end of line
- “(* *)” comments a block
- **the result** is the result of the last statement
- Color words: **the**, **named**
  - **the item named “Bob” ⇔ item “Bob”**

Values

- integer: 2^{29}… 2^{-29}
- real: IEEE double precision
- boolean: true or false
- string: Macintosh styled text
- list: **{1, 3.14, “foo”}**
- record: **{name: “Bob”, rank: 42}**
- object specifier: **window 1 of app “Finder”**
- Unicode text: UTF-16 text, not styled

References

- Not exactly a pointer
- Defers evaluation of the referenced expression

set x to 19
set q to a reference to x
set the contents of q to 23
get x → 23

set x to {1, 2, 3}
get last item of x → 3
set q to a reference to the last item of x
set end of x to 4 -- x is now {1, 2, 3, 4}
contents of q → 4

Operators

- Resembles Pascal more than C
  - **and**, **or**, **not** and **=**, not **&&**, **||**, **!**, and **==**
- With some additions:
  - **&** to concatenate lists and strings
  - string and list containment
    - **starts with**, **ends with**, **contains**
  - comparisons may be spelled out
    - **≤** ⇔ **is less than or equal to**

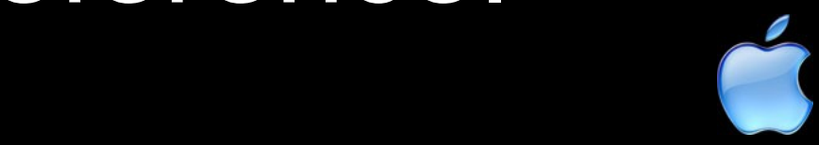
Variables

- **set x to 19**
- **[get] x**
- **copy 19 to x**
  - **set** shares complex values
  - **copy** clones them
- Parallel assignment

```
set {a, b, c} to {1, 2, 3}
set {a, b} to {b, a}
```

Subroutines

- (**on** | **to**) *name parameters*
  …
  end *name*
- Parameters may be positional
  on f(a, b, c)
- Parameters may be named
  to f from a through b
- Scalar parameters are passed by value, complex ones by reference.



If

- Simple

  if *condition* **then** *statement*

- Block

```
if condition then
    ...
[else if condition then
    ... ]
[else
    ... ]
end if
```

Repeat

- **repeat** (forever)
- **repeat** *n* **times**
- **repeat with** *i* **from** *x* **to** *y* [**by** *step*]
- **repeat with** *i* **in** *list*
- **repeat while** *condition*
- **repeat until** *condition*
- **exit repeat**

Try

- Handle exceptions (or just ignore them)

```
try
    ...
    [on error [message] [number number]
        ... ]
end try
```

- Throw exceptions

```
error [message] [number number]
```

Considering and Ignoring

- Alter AppleScript's normal behavior
  - **considering** *attribute* [**but ignoring** *attribute*]
  - **ignoring** *attribute* [**but considering** *attribute*]
- String comparisons
  - **case, diacriticals, expansion, hyphens, punctuation, white space**
- Event sending
  - **application responses**

Tell

- Tell specifies a default subject

```
tell window 1 of application “Finder” to close
```

- Tell subject completes partial specifiers

```
tell application “Finder”
  tell window 1 -- of application “Finder”
    close -- window 1 of application “Finder”
  end tell
end tell
```

- **it** is the current subject

Objects and Scripts

- Objects have properties and elements
  - property ⇒ exactly one
    - name of a document
  - element ⇒ zero or more
    - documents of an application

Objects and Scripts

- Make your own objects using script objects

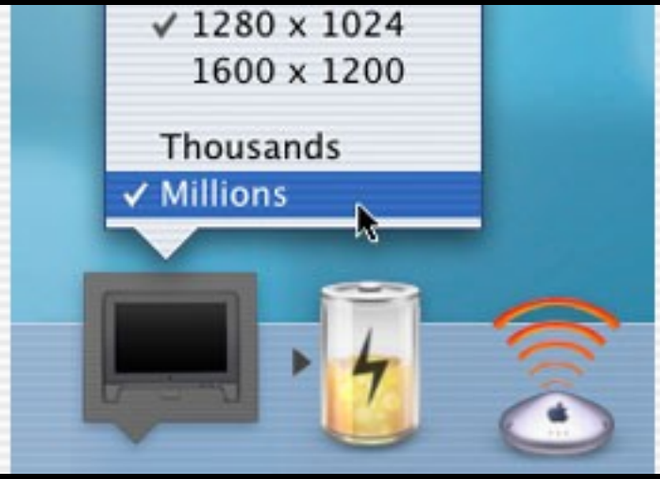
```
script thingie
  property color: “blue”
  on frotz…
end script
```

- Script objects can inherit from other scripts through enclosure or the **parent** property
- Properties and handlers, but not elements
- Can’t make new ones at run time

Object specifiers

- *piece* of *parent* -or- *parent*'s *piece*
  - **name of document 1 ⇔ document 1's name**
- property: **name of x**
- by index: **document 1, first document**
- by name: **document "MacHack"**
- range: **items 1 through 3**
- all: **every item**
- filter: **every shape whose color is blue**

What’s Different



Philosophy

- English-like
- Dynamic vocabulary
- Script the model, not the interface

English-like

- Easy to read
- Often easy to write

- Not actually English!
  - Read the dictionary first.

Dynamic vocabulary

- Components define task-specific terms for their objects, so they match the user's task

- Same word can mean different things to different objects
  - Be aware of who you're telling to do what

Script the Model

- Why *not* the inteface?
  - Easy to record, but difficult to read and write
  - Often more complex
  - Interface changes often; model doesn’t
- Think in model terms
  - Don’t activate applications unless you want to
  - Don’t select things, just change them
  - Use variables, not the clipboard

Every and Whose

- **every** and **whose** let you eliminate loops!

```
tell folder x of application “Finder”
    repeat with i from 1 to count items
        if name of item i ends with “.tmp” then
            delete item I
        end if
    end repeat
end tell


tell application “Finder” to delete every item ¬
    of folder x whose name ends with “.tmp”
```

Every and whose

- **every** can even replace recursion:

```
set label index of every item of entire contents ¬
    of folder x to 0
```

- Caveats
  - Not all applications support them
  - Some do it incorrectly
  - Built-in types support **every** but not **whose**

The Rest of the Story

Expansion Options

- Scriptable applications
  - Lots of excellent commercial apps
  - Lots of excellent free ones, too!
    - See www.apple.com/applescript for a list
- Scripting additions (*aka* osaxen)
  - Add new global verbs, but not objects
  - See www.osax.com

Things I Skipped

- Syntax
  - **with timeout**, **with transaction**, **using terms from**, remote **tell**, raw data, raw events
- Techniques
  - text manipulation
  - file I/O
  - user interaction
  - and many, many more…

Other Resources

- www.apple.com/applescript
- AppleScript Language Guide
  - Available as HTML or PDF from Apple
  - Available as book from fatbrain.com
- AppleScript Users mailing list
  - www.lists.apple.com/applescript-users

Who to Contact

Jason Yeo
Mac OS Technology Manager
Apple Worldwide Developer Relations
jason@apple.com

Q&A