

たけうちとおるの
「絶対得する ... 自動化のすすめ」
全てを惜しみなく伝授するセミナー

AppleScript

セミナーレジュメ

ほんとになんでも
できる

Adobe Indesign も

classic 環境の
スピードスター☆

QuarkXPress も健在

10 以上はチラシに
あなどれない機能
盛りだくさん

Adobe Illustrator も

Adobe Photoshop も
ちょっとだけ

Adobe Acrobat も

Finder も

Excel も FileMaker も
Mail も

全部 AppleScript で
コントロール

try and error で
おぼえられます。

本番のデモは動く
か？ どうぞ期待！

たいへんでした。
ほんとうに。。

全てを惜しみなく伝授するセミナー
なので内容みっちりのレジュメ作りました (笑)

DTP はまだまだこ
れからですよ。

目次

第1章 最低限のAppleScript1	スクリプトラベル..... 13
AppleScriptとは 1	masterページアイテムを上書きする 14
スクリプト編集プログラム..... 1	トンボ作成..... 14
変数 set 変数 to ○○..... 1	ライブラリの配置..... 14
list変数 (配列変数) set 変数 to {○○,○○,○○} as list..... 1	スニペットの配置と書き出し..... 14
if文 if 変数 is ○○ then ~ end if..... 1	画像のフィット..... 15
repeat文 (くりかえしの処理) repeat ~ end repeat 2	選択画像の回転..... 15
コマンド (命令)・ハンドラ (関数)..... 2	画像の配置..... 15
演算子1 2	選択ファイルを配置&保存..... 15
演算子2 2	EPSのページ配置 15
	PDFのページ配置 16
第2章 AppleScriptの構文2	名刺の10枚面付け 16
対話する display dialog 3	名刺の10枚面付け2 17
ファイル選択choose file 3	縦組にする..... 17
フォルダ選択 choose folder 3	特殊なノンブル..... 17
ファイル保存choose file name 3	段落1行に長体 17
ファイルの読み込み..... 3	改ページ文字を入れる..... 17
AppleScript's text item delimitersを使って検索置換 4	searchをつかった検索置換..... 18
エラー処理など..... 4	段落スタイル..... 18
ちょっとした便利スクリプト..... 4	文字スタイル..... 18
	文字スタイルのフォントを変更..... 18
第3章 AppleScriptの練習4	字取り変更..... 18
ASじゃんけんスクリプト 5	字形変更..... 18
	text frameに (Tag付き) テキストを読み込む 19
第4章 アプリケーションコントロール6	ルビ..... 19
階層構造 tell ~ end tell 6	テキストを表にペースト..... 19
プロパティを調べる・セットする get properties 6	選択された表を調べる..... 19
用語辞書の調べ方..... 6	表の罫線を変更する..... 19
コマンドを送る..... 7	表内のあふれた文字に長体..... 19
リファレンスの調べ方..... 7	疑似囲み罫..... 20
	表の行を追加する..... 20
第5章 Illustratorをコントロールする 8	値段表の値段を一律料金アップする..... 20
選択されたオブジェクトを調べる..... 8	PDFのブックマーク作成 20
複製・移動する..... 8	PDFハイパーリンク作成 20
オブジェクトを作成する..... 8	
変倍する..... 8	第7章 AcrobatとPhotoshopほか 21
document 8	PDFにしおりをつける 21
layer 9	PDFをバラす 21
path item 9	しおりのプロパティ変更..... 21
text frame 9	簡易デジタル検版..... 21
paragraph 段落 9	Excelサンプル。セル内の改行を にする 21
テキストボックスに1行になるよう長体 9	Finder拡張子表示 22
character 文字 10	Finderアクセス権変更 22
オーバーフローを調べる..... 10	mail新規メールに定型文挿入 22
文字を挿入する..... 10	classic環境のQuarkXPress 22
テキスト全部保存..... 10	
連続数字を等幅半角字形にする..... 11	第8章 自動組版の注意点 22
画像を配置する埋め込みとリンク..... 11	データベース設計..... 22
クリッピングマスクする..... 11	理論上できるのと動かしきるのは大違い..... 22
画像の回転..... 11	ヒューマンエラー..... 22
トンボ作成..... 11	事故..... 22
線幅を変える..... 12	
ほかできること、できそうにないこと..... 12	付録 デザイナーのためのFLASH講座初級 23
処理が止まってしまうとき..... 12	2つのマスターすべき事 (オブジェクトと時間) 23
	オブジェクトの基本..... 23
第6章 InDesignをコントロールする 12	時間の基本..... 23
選択されたオブジェクトを調べる..... 12	時間:練習 (拡大と回転) 23
Document 13	時間:練習 (パスにそって移動) 24
すべてのドキュメントをPDF書き出し 13	オブジェクト:練習 (FLASHに直接描く) 24
マスターページの横ガイドを消す..... 13	オブジェクト:練習 (小さなMOVIEを作る) 24
ガイドを作成する..... 13	オブジェクト:練習 (シーンに配置) 24
レイヤーを作成する..... 13	オブジェクト:練習 (タイミングをずらす) 24
線の作成..... 13	時間:練習 (ムービーが終わったらストップする) 24
ページ数を調べる..... 13	作成する (パブリッシュ) 24
スプレッド1は何ページあるか 13	ほかにも..... 24

第1章 最低限のAppleScript

AppleScriptの参考書籍は非常に少ないし手に入れにくいのですが、尊敬する掌田 津耶乃さんの「AppleScript Studioでゼンマイびゅんびゅん!!」や「AppleScript Programming for Mac OS X — Mac OS X v10.2対応」それとOS9だがこばやしゆたかさんとAppleScriptリファレンス制作委員会の「AppleScriptリファレンス」も非常に役に立ちます。
またWebでは「AppleScriptPARK」からのリンクで有用なサイトに行けるとおもいます。
この章では詳しい解説は良書をお願いして、本当に簡単に必要最低限のAppleScriptをお伝えします。

AppleScriptとは

AppleScriptとはMacintoshのマクロ言語です。AppleScriptを使うと普段手作業の仕事がプログラムから行う事ができます。例えば100個ファイルがあったとき、ファイル名を1～100までの連番にしようとする手作業でもできますがたいへんだし不毛です。AppleScriptを使うとこれらの作業が一気に行えますし正確です。さらにファイル数が1000でも10000でも問題ありません。

MacOSXでDTPの世界ではAdobeのアプリケーションがAppleScriptに対応しています。(対応しているということはAppleScriptでコントロールできるということです。)ほかに古くからQuarkXPressがAppleScriptに対応し我々スクリプターがさまざまなツールを作ってきました。さらにFileMakerやEXCELなども対応していますのでDB組版などにも利用できます。

AppleScriptによってInDesignやIllustratorの細かい作業を自動化することにより作業効率が大幅にアップします。さらに突き詰めていくと自動組版が可能になり。もっと突き詰めていくとWebブラウザからのリクエストをうけて組版を開始することも可能です。さまざまな可能性を秘めたAppleScriptですが最近では少し目立たない存在になっているような気がします。ソフト自体どんどん便利になってきてAppleScriptで自動化しなくても大きく効率化できるからか、それとも、いまだAppleScript未対応のIllustrator8で仕事をつづけているためかもしれません。

これを読んでAppleScript対応の魅力に気がついていただき、ぜひAppleScriptにチャレンジしてみてください。

スクリプト編集プログラム

MacOSXでアプリケーション:AppleScriptの中のスクリプトエディタを起動して下さい。起動したら下記のように入力します。

```
display dialog "こんにちは!"
```

ウィンドウ上部の実行ボタンをクリックしてください。"こんにちは!"とダイアログが現れたら大成功です。次はIllustratorを起動してください。ドキュメントを作成してテキストオブジェクトを作り、適当に文字を入力しておいてください。スクリプト編集プログラムに下記を入力して実行します。

```
tell application "Adobe Illustrator"
  tell document 1
    set contents of text frame 1 to "こんにちは!"
  end tell
end tell
```

Illustratorのテキストが"こんにちは!"になれば成功です。"Adobe Illustrator"を"Adobe InDesign CS2_1"に変えても動きます。

※実行ボタンをクリックするとInDesignはどこにあるか聞いてくるかもしれませんが。その場合アプリケーションフォルダの中から選択してください。

どうでしょう。簡単なスクリプトですが大きな可能性に気がつくと思います。このスクリプトを応用してやるとクライアントからもらったEXCELの原稿をつぎつぎにIllustratorに流し込む事ができます。簡単ではないですがAppleScriptが使えるようになれば現場で大きな効率化をはたすことができます。ぜひチャレンジしてみてください。



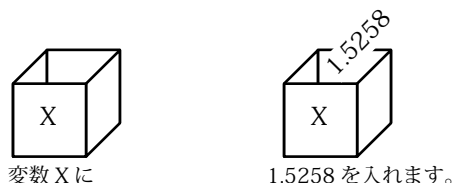
変数 set 変数 to ○○

変数は入れ物です。名前のついた入れ物で、なんでも入ります。取り出すときは入れ物の名前を指定してやると中身を見ることができます。変数には好きな名前をつけることができます。

```
set myStr to "こんにちは。"-変数myStrに"こんにちは。"を入れる
display dialog myStr-変数myStrをダイアログに表示
もちろん変数ですから数字も入れることができます。
set Y to 1-変数Yに1を入れる
set X to 5-変数Xに5を入れる
display dialog Y+X-変数Y+Xの結果をダイアログに表示
```

※--はコメントで、プログラムでは--以降の文字は無視されます。

変数の種類と変数の型変換



boolean--真偽値
integer--整数
real--実数
string--文字列
list--リスト
booleanにはtrue(真)かfalse(偽)が入ります。integerは整数です。小数点を含む実数を整数に変換するときに使います。

```
set X to 1.5258-変数Xに1.5258を入れる
set X to X as integer-変数Xを型変換する
display dialog X-変数Xをダイアログに表示
=>2
```

変数1.5258を整数に型変換したため結果は2になりました。

```
set myNum to 1
set myStr to myNum as string
```

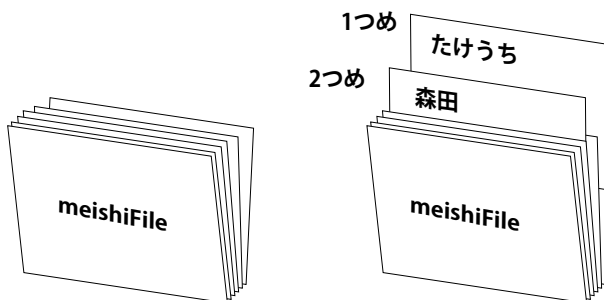
上の例は変数myNumを文字列に変換してmyStrに入れます。人間の見た目は整数の1も文字列の"1"もあまり変わりませんが、AppleScriptからDTPアプリケーションをコントロールするときなど良く使われます。

list変数 (配列変数) set 変数 to {○○,○○,○○} as list

変数は何でも入る名前のついた入れ物と書きましたがlistはその入れ物の中にさらに仕切りがあり番号がついているイメージです。例えば名刺ファイルがあったとします。その中に名刺を入れます。1から順に名刺を入れていきます。1番目は"たけうち"さん、2番目は"森田"さん、3番目は"おおもり"さん、4番目は"ばば"さん、5番目は"なかむら"さんといった感じです。あとで取り出すときに、名刺ファイルの5番目とか1番目から順番に全部の名刺とかいうように取り出します。これをAppleScriptで書くと

```
set meishiFile to {"たけうち","森田","おおもり","ばば","なかむら"} as list
display dialog item 5 of meishiFile & "さん"
=>"なかむらさん"
```

というようになります。このlist変数がわかりにくいのですが例えばIllustratorで選択したアイテムを調べると選択アイテムは1つとは限らないので複数のlistで値が帰ってきます。そこから選択アイテムの1つめといったように取り出します。とにかく非常によく出てくるしlistを使わないとAppleScriptは書けないので、がんばって理解してください。



変数meishiFileに1つめは「たけうち」2つめは「森田」と入れる。取り出すときはmeishiFileの1番目というように取り出す。

if文 if 変数 is ○○ then ~ end if

「もしも～なら」という処理です。必ずend ifがいります。

```
set AAA to "あいうえお"
if AAA is "あいうえお" then --もしAAAが"あいうえお"なら
    display dialog "あいうえおです。"
end if
```

「もし～ではなかったら」の場合は下記のように書きます。

```
set AAA to "あいうえお"
if AAA is not "あいうえお" then --もしAAAが"あいうえお"ではなかったら
    display dialog "あいうえおではありません。"
end if
```

「もしも～なら」と「そうではなくて～なら」と「それ以外なら」というように条件によって振り分ける場合は下のように書きます。

```
set AAA to "あいうえお"
if AAA is in "あいう" then --もしAAAが"あいうえお"に含まれるなら
    display dialog "あいうえおです。"
else if AAA is "かきくけこ" then --もしAAAが"かきくけこ"なら
    display dialog "かきくけこです。"
else--それ以外なら
    display dialog "それ以外です。"
end if
```

「もしも～でさらに～なら」と「もしも～または～なら」というように2つの条件によって振り分ける場合は下のように書きます。

```
set AAA to 10
if 0 < AAA and AAA < 20 then--andの入ったif文
    display dialog "変数AAAは0より大きく20より小さい"
end if
if 0 > AAA or AAA > 20 then--orの入ったif文
    display dialog "変数AAAは0より小さいまたは20より大きい"
end if
```

以上のようにif文はかなり簡単ですがif文そのものよりもどこでif文を使うかという考え方のほうが理解しにくいかもしれません。

repeat文（くりかえしの処理） repeat ～ end repeat

くりかえし1

1から10まで順に繰り返す。InDesignなら1ページから10ページまで処理をするという時にrepeat文を使います。くりかえしでもっともよく使う処理が下です。下記は変数CCCに1～10までの値を順番に入れながら実行します。必ずend repeatがいります。

```
repeat with CCC from 1 To 10
    display dialog CCC
endrepeat
=>結果は1から10までの文字がダイアログで発生する。
```

くりかえし2

無限に繰り返す処理を入れある条件が成立したら処理を終わります。

```
set DDD to 1--変数DDDに1を入れる。
repeat--無限に繰り返す。
    if DDD > 50 then--もし変数DDDが50以上なら
        exit repeat--繰り返しを抜ける
    end if
    set DDD to DDD + 1--変数DDDに1をプラスする。
end repeat
```

くりかえし3

配列変数のなかを1つずつ変数に入れます。

```
set meishiFile to {"たけうち", "森田", "おおもり", "ばば", "なかむら"} as list
repeat with myName in meishiFile
    --↑配列変数meishiFileの中を1つずつ順に変数myNameに入れる
    display dialog myName & "さん"
end repeat
=>「たけうちさん」「森田さん」「おおもりさん」。。。と順に表示される
```

コマンド（命令）・ハンドラ（関数）

コマンド（命令）

メッセージを表示する。下のような命令をコマンドといいます。

```
display dialog AAA
```

上のように命令があってその後にはパラメータが入ります。ほかにもいろいろなパラメータやオプションパラメータなどたくさんの文字が続くことも多いです。さらに値を返すコマンドもあります。AdobeやQuarkXPress共通でなにかを作りたいときにmakeというコマンドを作ります。

```
make text frame
```

というように書くのですが

```
set myObj to make text frame
```

とすると作られたオブジェクトが変数myObjに入ります。

ハンドラ（関数）

下記のような値を返すカッコの着いた命令をハンドラ（関数）と言います。Round()は数字をまるめる関数です。関数には()←カッコがついていてカッコの中には値を入れます。

```
set BBB to Round(10.5)
```

自分でハンドラ（関数）を作る事も出来ます。

```
my Disp("あいうえお")
```

--自分で作った関数Dispを呼び出す。myが必要なので注意

```
on Disp(AAA)
```

--自分が作った関数Dispカッコの中の変数に渡された値が入る。

```
    display dialog AAA & "です。"
```

```
end Disp
```

演算子1

足し算、引き算、かけ算、割り算。基本はこれだけです。あとできれば割ったあまりと割ったあまりを除いた整数部分が取り出せると便利です。

```
set X to A+1--変数Xに変数A+1を入れる
set X to X-1--変数Xに変数A-1を入れる
set X to N*5--変数Xに変数N×5を入れる
set X to N/2--変数Xに変数N÷2を入れる
set X to 5 mod 2--=>1になる。割ったあまり。
set X to 5 div 2--=>5になる。割ってあまりを除いた整数部分
```

割ったあまりをだすmodを使って偶数か奇数かを調べることが出来ます。

```
if N mod 2 = 1 then--もし変数Nを2で割ったあまりが1なら
```

```
    display dialog "変数Nは奇数です。"
```

```
else--そうでないなら
```

```
    display dialog "変数Nは偶数です。"
```

```
end if
```

これが最低限ですが、これだけでScriptを書くには十分です。

演算子2

文字を連結するときには&を使います。

```
set AAA to 10
set myStr to "変数AAAの内容は" & AAA & "です。"
=>"変数AAAの内容は10です。"
```

比較演算子などif文の中に書き込む内容です。

```
set AAA to 10
if AAA > 5 then
    display dialog "変数AAAは5より小さい"
end if
if AAA is 10 then
    display dialog "変数AAAは10"
end if
if AAA is not 5 then
    display dialog "変数AAAは5以外"
end if
if AAA is in {9, 10, 11} then
    display dialog "変数AAAは9,10,11に含まれる"
end if
```

第2章

AppleScriptの構文

AppleScriptのユーザーインターフェイスは簡単なダイアログを出してなにかを入力させるか選ばせるかする程度です。しかしこれだけでも様々なことが出来ますし、複雑なインターフェイスはXcodeで作ることが出来ます。興味のある方はインストラCDからインストールできますし参考書籍が多数あります。

対話する display dialog

確認

display dialogの出し方と何を入力したか調べます。

```
display dialog "こんにちは" --キャンセルをクリックすると処理終了
```



display dialog は変数を単に表示するだけで今入っている変数の値を確認するデバッグ用としてつかえる。

入力

```
set myAnswer to display dialog "入力してください" default answer "ここに入力"  
set inputStr to text returned of myAnswer  
display dialog "入力したのは「" & inputStr & "」ですね"
```



入力された文字は変数myAnswerの中に入っている。text returnedで取り出す事が出来る。

ボタン

```
set myAnswer to display dialog "食べたいものは" buttons {"くり", "こめ", "つけもの"}  
set selectedBtn to button returned of myAnswer  
display dialog "クリックしたのは「" & selectedBtn & "」ですね"
```



押されたボタンも変数myAnswerに入っている。こちらはbutton returnedで取り出す。

入力とボタン

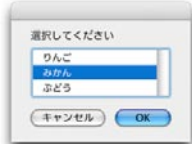
```
set myAnswer to display dialog "ページ数" default answer "1" buttons {"縦組", "横組"}  
set inputStr to text returned of myAnswer  
set selectedBtn to button returned of myAnswer  
display dialog "ページ数が" & inputStr & "ページで" & selectedBtn & "ですね"
```



入力した文字と押されたボタン両方を調べる事も出来る。

選択

```
set mySelect to choose from list {"りんご", "みかん", "ぶどう"} with prompt --  
"選択してください"--は実際には改行を入れない  
display dialog item 1 of mySelect--listで返ってくるのでitem 1 で取り出す。
```



こちらはリストから選択する。さらに下記はリストから複数選択可能にしたもの。どちらも値はリストで返ってくるので要注意。

```
set mySelectList to choose from list {"りんご", "みかん", "ぶどう"} with prompt --  
"選択してください" with multiple selections allowed --複数選択可  
repeat with mySelect in mySelectList --listの中のitem分繰り返す  
display dialog mySelect  
end repeat
```

ファイル選択choose file

ファイルを選択します。choose fileで選ぶとalias型の値で返ってくるのでas stringで型変換するとテキスト形式のフルパスが返ってきます。その後なんらかの処理を (InDesignに配置するとかテキストを読み込むとか) すればOKです。

```
set this_item to choose file with prompt "配置ファイルを選択"  
set Fpath to this_item as string--ファイルのフルパスをテキスト形式で取り出す  
display dialog Fpath--表示する。
```

下記ではファイルタイプが"EPSF"か"EPSP"かまたは種類が"EPS ファイル"のファイルだけ処理関数に渡すといったちょっと気の利いた事をしていきます。下記では処理関数setEPS2を呼び出しています。

```
set this_item to choose file with prompt "配置ファイルを選択"  
set Fpath to this_item as string--ファイルのフルパスをテキスト形式で取り出す  
set Finfo to info for this_item--ファイルインフォメーションを取り出す  
set FType to file type of Finfo  
--ファイルインフォメーションからファイルタイプを取り出す  
set Fkind to kind of Finfo--ファイルインフォメーションから種類を取り出す  
if FType is "EPSF" or FType is "EPSP" or Fkind is "EPS ファイル" then  
my setEPS2(this_item, Fpath)--処理をする。  
end if
```

フォルダ選択 choose folder

choose folderでフォルダ選択ダイアログを出せます。フォルダはalias型の値で返ってきます。フォルダ内のファイルを全て処理するには下記のようなrepeatを使います。list folder フォルダでフォルダ内のファイルがリスト形式で返ってきますが、それらはファイル名のみです。そのためファイルの位置はフォルダのフルパス+ファイル名で調べます。

```
set myFol to choose folder with prompt "フォルダを選択"  
set myFol to myFol as string--フォルダのフルパス  
repeat with myFile in list folder myFol without invisibles--フォルダ内リピート  
set this_item to (myFol & myFile) as alias--フォルダとファイル名を合体  
set Fpath to this_item as string--フルパスを取り出す  
display dialog Fpath  
end repeat
```

下記はフォルダ内のフォルダも含めて繰り返します。

```
set myFol to choose folder with prompt "フォルダを選択"  
my folderLoop(myFol)--オリジナル関数folderLoopを呼び出す
```

```
on folderLoop(myFol)  
set myFol to myFol as string  
repeat with myFile in list folder myFol without invisibles  
set this_item to (myFol & myFile) as alias  
set myInfo to info for this_item  
if folder of myInfo is true then  
my folderLoop(this_item)--自分自身を呼び出す  
else  
set Fpath to this_item as string  
display dialog Fpath  
end if  
end repeat  
end folderLoop
```

ファイル保存choose file name

ファイル保存ダイアログを出します。alias型の値で返ってきます。このダイアログを使うのはテキスト整形したものを保存する時です。

```
set newFile to choose file name  
display dialog newFile as string  
作成したファイルに書き込むには下記を使います。  
set newFile to choose file name
```

```
open for access newFile with write permission--書き込可能でファイルを開く  
write "こんにちは" to newFile--書き込む  
close access newFile--ファイルを閉じる
```

ところがこのあとで「おはよう」と書き込むと「おはよう」と前の文字のあふれた部分が残ってしまいます。うまい解決策は思いつかなかったのですが、下記ではファイルが存在するか調べて存在するならばゴミ箱に捨てる処理をさせて逃げています。

```
set newFile to choose file name  
  
tell application "Finder"--ファイルが存在するかどうかは"Finder"を使う  
if exists newFile then--もし存在するならば  
delete newFile--ゴミ箱に捨てる  
end if  
end tell
```

```
open for access newFile with write permission  
write "おはよう" to newFile  
close access newFile
```

ファイルの読み込み

下記はファイルを選択し読み込んだ内容を変数myStrに入れ表示します。開いたファイルは必ず閉じるようにしてください。

```
set myFile to choose file with prompt "ファイルを選択"  
open for access myFile--ファイルを開く  
set myStr to read myFile--読み込み結果を変数myStrに入れる
```

```
close access myFile--ファイルを閉じる
display dialog myStr
```

上とほとんど同じなのですが as list using delimiter {return} を付けると text ではなく改行区切りの list で読み込みます。1行づつ処理したい場合は便利です。

```
set myFile to choose file with prompt "ファイルを選択"
open for access myFile
set myList to read myFile as list using delimiter {return}
close access myFile
display dialog item 1 of myList
```

AppleScript's text item delimiters を使って検索置換

AppleScript's text item delimiters は AppleScript 内部の区切り文字の設定です。例えば区切り文字を "," にして "あいう,かきく,さしす" という文字列をリストに型変換すると {"あいう","かきく","さしす"} というリストになります。コマンド区切りの文字を読み込みリストにするのに大変便利です。

区切り文字を "%" に変更してリストを文字列に型変換すると区切り文字をつかってリストがつながります。先ほどのリストは "あいう%かきく%さしす" になります。これを利用してやれば検索置換が出来ます。

```
set myStr to "あいう,かきく,さしす"
set OriginalDelimiters to AppleScript's text item delimiters
--元々の区切り文字を保存しておく。決まり事として必ず使う
set findStr to "," --検索文字 set repStr to "%" --置換文字
set AppleScript's text item delimiters to {findStr}--区切り文字を","にする
set myStr to text items of myStr--","区切りでリストにする
set AppleScript's text item delimiters to {repStr}--区切り文字を"% "にする
set myStr to myStr as string--"%区切りでテキストにする
set AppleScript's text item delimiters to OriginalDelimiters
--もともとの区切り文字に戻しておく。これも決まり事。
display dialog myStr
=>あいう%かきく%さしす
```

検索置換のハンドラ (関数) replaceAll です。これをスクリプトの一部に書いておいて set myStr to my replaceAll("元の文字", "検索文字", "置換文字") とすれば検索置換できます。大量に検索置換すると (もしくは特定の文字があると) 文字化けする時があります。

```
set myStr to "あいう,かきく,さしす"
set myStr to my replaceAll(myStr, ",", "% ")
display dialog myStr
```

```
on replaceAll(motoStr, findStr, repStr)
set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {findStr}
set motoStr to text items of motoStr
set AppleScript's text item delimiters to {repStr}
set motoStr to motoStr as string
set AppleScript's text item delimiters to OriginalDelimiters
return motoStr
end replaceAll
```

下記ハンドラで set myCount to my countFields("元の文字", "区切り文字") で変数 myCount には元の文字を区切り文字で分割した数が入ります。

set myStr to my nthFields("元の文字", "区切り文字", アイテム数) で元の文字を区切り文字で分割した2つめ等を取り出せます。

```
set myStr to "あいう,かきく,さしす"
set myCount to my countFields(myStr, ",")
set myStr to my nthFields(myStr, ",", 2)
display dialog "合計" & (myCount as string) & "個で2つめは:" & myStr
```

```
on countFields(myStr, sep)
set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {sep}
set myStr to text items of myStr
return count myStr
end countFields

on nthFields(myStr, sep, num)
set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {sep}
set myStr to text items of myStr
set AppleScript's text item delimiters to OriginalDelimiters
return item num of myStr
end nthFields
```

エラー処理など

try文でエラー処理

Illustrator に AppleScript を送信する際テキストフレームにテキストをセットするスクリプトだったとします。そのときもしテキストフレームが無ければエラーになります。下記 try ~ on error ~ end try でエラー処理をすることができます。

```
with timeout of 1 second--タイムアウトを1秒に設定
tell application "Adobe Illustrator"
tell document 1
try--tryスタート
set contents of text frame 1 to "あいうえお"
on error--もしエラーがあれば下を実行
display dialog "失敗しました。"
end try--try文終わり
end tell
end tell
end timeout--タイムアウト設定ここまで
```

タイムアウトの設定

Illustrator や InDesign などは連続して AppleScript を送信しつづけるとだんだん遅くなったり EPS を配置する時に「配置できません」とダイアログが表示されたりします。またスクリプトを送信しているのに応答しない事もしばしばあります。そういう時 AppleScript は応答待ちになってしばらく止まるのですが with timeout of 1 second をすると1秒だけしか待たないようになります。

ちょっとした便利スクリプト

アスキーナンバーで文字を指定したり逆にアスキーナンバーを調べます。

```
set myChar to ASCII character (34)
set myNum to ASCII number "A"
--ちなみに改行はreturnでタブはtabです。
```

下記はクリップボードを操作します。コピーした文字列を取り出したり、クリップボードにテキストを入れて、ペーストすることができます。

```
set the clipboard to "123"--クリップボードに文字列を入れる
```

```
set myTable to the clipboard--クリップボードから文字列を取り出す
```

下記は SORT です。早くありませんが、リストを並べ替えます。

```
on SORT(myList)
set myCount to count items of myList
repeat
set myFLG to false
repeat with N from 1 to myCount - 1
set buf1 to item N of myList
set buf2 to item (N + 1) of myList
if buf1 < buf2 then
set item N of myList to buf2
set item (N + 1) of myList to buf1
set myFLG to true
end if
end repeat
if myFLG is false then
exit repeat
end if
end repeat
return myList
end SORT
```

第3章

AppleScriptの練習

AppleScript の練習にもってこいの教材を考えました。「じゃんけん」です。「さいしよはぐー」からはじまりぐーちよきばーの中から自分の手を選びます。コンピュータの手も同様にランダムに作ります。最後に勝ち負け判定をして終わります。ポイントは「さいしよはぐー」で「ぐー」を選ばなければ何度も繰り返す事と「あいこ」ならば同様に何度も繰り返す事です。

ASじゃんけんスクリプト

下記のようなじゃんけんするスクリプトを書いてみました。簡単で楽しいのでスクリプトの練習にはもってこいです。

まず「さいしよはぐー」からはじめます。「ぐー」を選ばなければ「最初はぐーだよ」とダイアログを出します。次にコンピュータの手を決めます。random number from 1 to 3でランダムな数字を出してから決めます。最後に勝ち負け判定をして終了です。

```
set AAA to choose from list {"ぐー", "ちょき", "ぱー"} with prompt "さいしよはぐー!"
if AAA is not {"ぐー"} then
  display dialog "最初はぐーだよ"
end if

set AAA to choose from list {"ぐー", "ちょき", "ぱー"} with prompt "じゃんけん"
set myJanken to AAA--わたしの手はmyJankenに入る
set BBB to random number from 1 to 3--1から3のランダムな数字がBBBに入る
if BBB is 1 then--もし1なら
  set asJanken to {"ぐー"}--変数asJankenを{"ぐー"}にする
else if BBB is 2 then
  set asJanken to {"ちょき"}
else
  set asJanken to {"ぱー"}
end if

if myJanken is asJanken then
  display dialog "あいこだよ"
else if myJanken is {"ぱー"} and asJanken is {"ぐー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
else if myJanken is {"ぐー"} and asJanken is {"ちょき"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
else if myJanken is {"ちょき"} and asJanken is {"ぱー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
else if myJanken is {"ぱー"} and asJanken is {"ちょき"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
else if myJanken is {"ぐー"} and asJanken is {"ぱー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
else if myJanken is {"ちょき"} and asJanken is {"ぐー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
end if
```

さてこのスクリプトには欠点がいくつもあります。まず「さいしよはぐー」の部分ですが「ぐー」を出さなくてもじゃんけんに進めます。「ぐー」をださない限り進めなくしたいものです。

```
repeat
  set AAA to choose from list {"ぐー", "ちょき", "ぱー"}
  with prompt "さいしよはぐー!"
  if AAA is not {"ぐー"} then--ぐーでなければ
    display dialog "最初はぐーだよ"--ダイアログを出す
  else--そうでなければ (ぐーなら)
    exit repeat--repeatを抜ける
  end if
end repeat
```

上のようにrepeatを入れぐーならばrepeatを抜かれるようにすれば「さいしよはぐー」の部分は完成です。

次の欠点は勝っても負けてもあいこでもじゃんけんが終了してしまう事です。こちらrepeat文で直す事が出来ます。

```
repeat
  set AAA to choose from list {"ぐー", "ちょき", "ぱー"} with prompt "じゃんけん"
  set myJanken to AAA
  set BBB to random number from 1 to 3
  if BBB is 1 then
    set asJanken to {"ぐー"}
  else if BBB is 2 then
    set asJanken to {"ちょき"}
  else
    set asJanken to {"ぱー"}
  end if

  if myJanken is asJanken then
    display dialog "あいこだよ"
  else if myJanken is {"ぱー"} and asJanken is {"ぐー"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
    exit repeat
  else if myJanken is {"ぐー"} and asJanken is {"ちょき"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
    exit repeat
  else if myJanken is {"ちょき"} and asJanken is {"ぱー"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
```

```
exit repeat
else if myJanken is {"ぱー"} and asJanken is {"ちょき"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
  exit repeat
else if myJanken is {"ぐー"} and asJanken is {"ぱー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
  exit repeat
else if myJanken is {"ちょき"} and asJanken is {"ぐー"} then
  display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
  exit repeat
end if
end repeat
```

これでいちおう完成ですが、勝ち負け判定に同じ処理を何度も書いています。こういうのはスクリプト的に非常によろしくないですし、そもそも勝ち負け判定自体もif文がいっぱいですスマートではありません。

まず勝ち負け判定ですが「あいこ」なら繰り返しでそれ以外なら勝ちか負けなので繰り返しを抜けます。そして「勝ち」かどうか判定できれば「あいこ」と「勝ち」以外は自動的に「負け」なので判定する必要はありません。これをスクリプトにすると下記のようになります。

```
if myJanken is asJanken then
  display dialog "あいこだよ"
else
  if myJanken is {"ぱー"} and asJanken is {"ぐー"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
  else if myJanken is {"ぐー"} and asJanken is {"ちょき"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
  else if myJanken is {"ちょき"} and asJanken is {"ぱー"} then
    display dialog "わたしの手は" & asJanken & "でした。あなたの勝ちです。"
  else
    display dialog "わたしの手は" & asJanken & "でした。あなたの負けです。"
  end if
  exit repeat
end if
```

これで多少はスマートになりましたがまだまだ工夫の余地がありそうです。私がいろいろ考えたところなんとか下記のようにになりました。みなさんもいろいろ工夫してみてください。

```
set MSGSTR to "さいしよはぐー!"
repeat
  set asJanken to item (random number from 1 to 3) of {"ぐー"}, {"ちょき"}, {"ぱー"}
  set myJanken to choose from list {"ぐー", "ちょき", "ぱー"} with prompt MSGSTR
  if myJanken is {"ぐー"} and MSGSTR is "さいしよはぐー!" then
    set MSGSTR to "じゃんけんぽん!"
  else if MSGSTR is "さいしよはぐー!" then
    display dialog "さいしよはぐーだよ!"
  else if myJanken is asJanken then
    set MSGSTR to "あいこーでしょ!!"
  else if {myJanken, asJanken} is {"ぱー"}, {"ぐー"} or -
    {myJanken, asJanken} is {"ぐー"}, {"ちょき"} or -
    {myJanken, asJanken} is {"ちょき"}, {"ぱー"} then
    set FLG to "勝ち"
    exit repeat
  else
    set FLG to "負け"
  end if
  exit repeat
end repeat
display dialog "わたしの手は" & asJanken & return & "あなたの手は" & -
myJanken & return & "あなたの" & FLG & "です。"
```

第4章 アプリケーション コントロール

AppleScriptではアプリケーションの対応状況によって使える機能が違い、命令方法も違ってきます。たとえばIllustratorなら画像を配置しクリッピングマスクの設定等がありますが、InDesignではありません。Quarkならtext boxやPicture boxなのですがInDesignではtext frameだったりrectangleだったりします。それらの違いがAppleScriptの難しさなのですが、get propertiesを使って調べたり、用語辞書を調べたり、またはメーカーが提供しているリファレンスを読んだりして調べていきます。

階層構造 tell ~ end tell

AppleScriptでアプリケーションをコントロールするときは階層になっています。"tell"や"of"でつないでいきます。例えばInDesignでは

```
tell application "Adobe InDesign CS2_J" -- InDesignの
tell document 1 -- 最前面のドキュメントの
tell page 1 -- 1ページ目の
tell text frame 1 -- 最前面のテキストフレームの
set contents to "あいうえお" -- 内容を"あいうえお"にする。
end tell
end tell
end tell
end tell
```

というように階層になっています。InDesignのドキュメントの1ページ目のテキストフレームという意味。ofを使うと下ののように簡略化して書けます。

```
tell application "Adobe InDesign CS2_J"
tell text frame 1 of page 1 of document 1
set contents to "あいうえお"
end tell
end tell
```

文字スタイルの数を取り出すときもドキュメントの文字スタイルの数というように階層をたどって取り出します。

```
tell application "Adobe InDesign CS2_J"
tell document 1
set myChrCount to count character styles -- 文字スタイルの数を数える。
end tell
end tell
```

プロパティを調べる・セットする get properties

AppleScriptでアプリケーションをコントロールする最も簡単な方法は適当にオブジェクト(テキストなど)をInDesignやIllustratorで作ってプロパティを調べる方法です。調べたらそれを変更できるかやってみます。

Illustratorに"あいうえお"というテキストを作りプロパティを見ましょう。

```
tell application "Adobe Illustrator" -- Illustratorの
tell document 1 -- 最前面のドキュメントの
tell page item 1 -- 最前面のアイテムの
get properties -- プロパティを調べる
end tell
end tell
end tell
```

このget propertiesは「QuarkXPressを操ろう」をNiftyのFPRINTフォーラムで連載されてい鎌田さんゆずりの方法でこれを初めて知って以来いろんなオブジェクトにget propertiesしてきました。いまAppleScriptが使えるようになったのもこのget propertiesのおかげです。

さていろいろな値が加えてきました。抜粋します。

```
contents:"あいうえお",
text orientation:horizontal,
position:{86.22216796875, 641.115234375},
width:60.0,
height:15.0361328125,
name:""
```

```
class:text frame,
index:1
```

まず注目はclassです。このテキストのクラス名はtext frameだということがわかります。indexは1なのでpage item 1ではなくtext frame 1で指定する方が良いでしょう。contents:"あいうえお"とあります。これを変えてみましょう。変更するにはAppleScriptで変数に値を入れる方法と同じです。

```
set 変数 to ○○
```

でしたね。では

```
tell application "Adobe Illustrator" -- Illustratorの
tell document 1 -- 最前面のドキュメントの
tell text frame 1 -- 最前面のtext frameの
set contents to "かきくけこ" -- contentsを"かきくけこ"にする。
end tell
end tell
end tell
```

これを動かしてください。文字内容が変わりましたね。

```
tell application "Adobe Illustrator"
tell document 1
tell text frame 1
set position to {100, 100}
end tell
end tell
end tell
```

を動かすと文字の場所が変わります。場所とcontentsを一度にセットすることも出来ます。

```
tell application "Adobe Illustrator"
tell document 1
tell text frame 1
set properties to {position:{100, 100}, contents:"かきくけこ"}
end tell
end tell
end tell
```

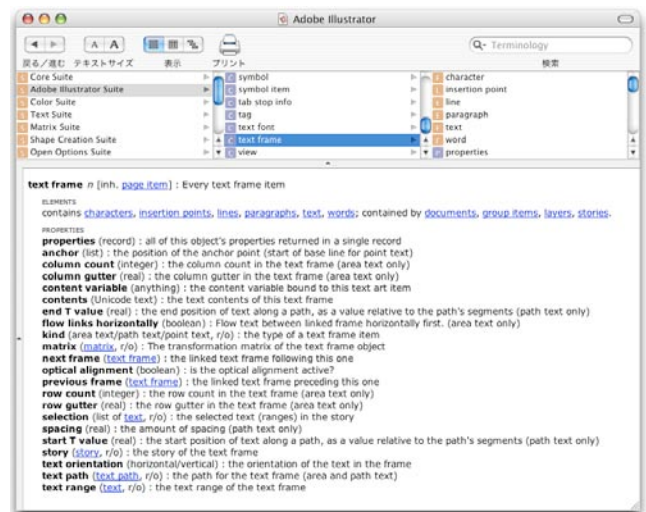
もちろんget propertiesだけではAppleScriptでInDesignをコントロールするだけの情報を調べるのは困難です。text frame 1でget propertiesしてもフォントやカラーのプロパティはありませんでした。実際フォントやカラーはtext frameのプロパティではなくtext frameの要素であるparagraphやcharacterのプロパティです。しかし、それを調べるのはget propertiesだけでは無理です。このようにもつといろいろ調べたい場合は後述する用語辞書やリファレンスを見ます。

用語辞書の調べ方

AppleScriptからアプリケーションをコントロールするときは用語辞書を見て使えるコマンドやセットできるプロパティを調べます。

用語辞書を見るにはまずスクリーン編集プログラムのウィンドウメニューからライブラリを選びます。ライブラリウィンドウ上部の+ボタンをクリックしてInDesignやIllustratorを選択します。ライブラリウィンドウに登録されたらダブルクリックしてください。用語辞書が開きます。

試しにAdobe Illustrator Suiteのクラスでtext frameを見てみましょう。



Class text frame: Text frame item

Plural form:

text frames

Elements:

character by numeric index, as a range of elements,
before/after another element, satisfying a test
insertion point by numeric index, as a range of elements,
before/after another element, satisfying a test

line by numeric index, as a range of elements,
before/after another element, satisfying a test
paragraph by numeric index, as a range of elements,
before/after another element, satisfying a test
text by numeric index, as a range of elements,
before/after another element, satisfying a test
word by numeric index, as a range of elements,
before/after another element, satisfying a test

Properties:

<Inheritance> page item [r/o] -- subclass of page item
properties record
-- all of this object's properties returned in a single record
anchor list
-- the position of the anchor point (start of base line for point text)
column count integer
-- the column count in the text frame (area text only)
column gutter real
-- the column gutter in the text frame (area text only)
content variable anything
-- the content variable bound to this text art item
contents Unicode text
-- the text contents of this text frame
end T value real
-- the end position of text along a path, as a value relative to
the path's segments (path text only)
flow links horizontally boolean
-- Flow text between linked frame horizontally first. (area text only)
kind area text/path text/point text [r/o]
-- the type of a text frame item
matrix matrix [r/o]
-- The transformation matrix of the text frame object
next frame text frame
-- the linked text frame following this one
optical alignment boolean
-- is the optical alignment active?
previous frame text frame
-- the linked text frame preceding this one
row count integer
-- the row count in the text frame (area text only)
row gutter real
-- the row gutter in the text frame (area text only)
selection a list of text [r/o]
-- the selected text (ranges) in the story
spacing real
-- the amount of spacing (path text only)
start T value real
-- the start position of text along a path, as a value relative to
the path's segments (path text only)
story story [r/o]
-- the story of the text frame
text orientation horizontal/vertical
-- the orientation of the text in the frame
text path text path [r/o]
-- the path for the text frame (area and path text)
text range text [r/o]
-- the text range of the text frame

大量にでてきますが--より右に書いてあるのが解説です。

Elementsは要素です。character(キャラクター)が書いてありますが、text frameの要素にcharacterがあるということです。これをスクリプトにすると下記のようにさらに深い階層になります。characterの用語辞書はText Suiteに書かれています。

```
tell application "Adobe Illustrator"  
  tell document 1  
    tell text frame 1  
      tell character 2  
        get properties  
      end tell  
    end tell  
  end tell  
end tell
```

Propertiesがプロパティですね。この値を変更することによって、Illustratorを操ることが出来ます。

```
contents Unicode text -- the text contents of this text frame
```

はプロパティ contentsの値はユニコードテキストだということと、--以下の説明をみるとtext frameの内容だということがわかります。

```
row count integer -- the row count in the text frame (area text only)
```

というのはrow count(段組みの段数)がinteger (整数) で返ってくるこ

がわかります。[r/o]と書いてあるのはread onlyのため変更することは出来ません。

```
kind area text/path text/point text [r/o] -- the type of a text frame item
```

kindというのは種類でarea textかpath textかpoint textで値が返ってきます。テキストボックスを作ればarea textでクリックして文字を作ればpoint textで、ベジェ曲線に沿うようなのはpath textです。こちらも[r/o]なので変更不可です。(実はIllustrator10では変更できましたが変更すると変な形になっていました。)

コマンドを送る

get propertiesでtext frameのプロパティをいろいろ変更できました。ではtext frameを作成するにはどうすれば良いでしょう。これにはコマンドを使います。用語辞書のCore Suiteのコマンドにmakeがあります。

```
make: Make a new element
```

```
make
```

```
new type class -- the class of the new element.
```

```
at location reference -- the location at which to insert the element
```

```
[with data anything] -- the initial data for the element
```

```
[with properties record] -- the initial values for the properties of the element
```

```
Result: reference -- to the new object(s)
```

makeというコマンドの次の行にnew type classとありさらに次にat location referenceとあります。これは

```
make new 作りたいオブジェクトのクラス at 場所
```

という書き方になります。その後の[]でくくられている部分は省略可能です。Resultにreferenceとありますがこれは返り値で新しく出来たオブジェクトが返ってきます。では作りたいオブジェクトのクラスですが、先ほどget propertiesで調べたtext frameクラスを渡してみましょう。場所は最前面につくるのならbeginningでいいでしょう。

```
tell application "Adobe Illustrator"
```

```
  tell document 1
```

```
    make text frame at beginning
```

```
  end tell
```

```
end tell
```

ぽつんと小さなオブジェクトが出来ました。これがスクリプトから作成したtext frameです。さきほど省略したwith propertiesを使ってみましょう。with properties recordと書いてあるのでwith properties {○○: ~ ~}という使い方になります。text frameのプロパティは先ほど調べました。その中でcontentsを使ってみましょう。

```
tell application "Adobe Illustrator"
```

```
  tell document 1
```

```
    make text frame at beginning with properties {contents:"あああ"}
```

```
  end tell
```

```
end tell
```

ほかにもコマンドには

```
close: --ドキュメントを閉じる
```

```
close document 1 saving no--保存せずに
```

```
count reference -- オブジェクトの数を数える
```

```
count text frames
```

```
delete: -- オブジェクトを消す
```

```
delete text frame 1
```

```
duplicate: --複製する
```

```
duplicate text frame 1 to end
```

```
exists: --オブジェクトが存在するか?
```

```
if exists text frame 1 then ~ end if
```

```
move: --オブジェクトを動かす
```

```
move text frame 1 to end of layer 1
```

```
open: --ファイルを開く
```

```
open alias "Macintosh HD:test.ai"
```

```
save: --ファイルを保存する
```

```
save document 1 in file "Macintosh HD:test.ai" as Illustrator
```

などがあります。まだまだ他にもいろいろありますので後述します。

リファレンスの調べ方

用語辞書よりもっと詳しい解説が見たい場合は各メーカーが提供しているリファレンスやガイドブックを見ます。

AdobeInDesignやAdobeIllustratorなどはスクリプトリファレンスやスク립ティングガイドなどがインストーラCDに入っています。リファレンスは用語辞書で命令方法やテキストフレームやカラーといったオブジェクトの指定方法や使えるプロパティを調べるのに役に立ちます。(英語の場合もあります。)スク립ティングガイドは初心者からAppleScriptを扱う解説が詳しく載っています。特にInDesignCS2のスク립ティングガイドは非常に丁寧に解説されており。サンプルも多数掲載されています。私自身も知らないことやあやふやなことがたくさん載っていて、まだまだ勉強しなくてはと実感させられます。このガイドをくわしく読むだけで十分AppleScriptの勉強になりますし、この原稿を読む必要もなくなるぐらいです(笑)

第5章

Illustratorを コントロールする

IllustratorはAppleScriptで大部分コントロールする事ができます。うまくつくれば自動組版も十分可能です。IllustratorはAppleScript以外にもJavaScriptやVBScriptでも記述できます。JavaScriptではWindows、Mac共に動かす事ができます。

本章ではdocument、layer、text frameなど分類して書こうと思ったのですがかたきでサンプルをなるべくたくさん掲載するにどまりました。解説も不十分ですが、なんとかヒントにでもなればよいと思います。

選択されたオブジェクトを調べる

下記は選択されたアイテムの場所を調べX軸に10mmをプラスしている。AppleScriptを送信する際単位はptになるので10に2.83464をかける事になる
set pt to 2.83464--変数ptに2.83464を入れておく

```
tell application "Adobe Illustrator"
  tell document 1
    set selectedItemList to every page item whose selected is true
    --選択されたオブジェクトを調べる (listで返ってくる)
    repeat with selectedItem in selectedItemList
      --選択オブジェクト分繰り返す。
      set myPos to position of selectedItem--座標を調べる (listで返る)
      set myX to item 1 of myPos--X座標を取り出す
      set myY to item 2 of myPos--Y座標を取り出す
      set position of selectedItem to {myX + 10 * pt, myY}
      --新しい座標 (X+10×2.83464) にする
    end repeat
  end tell
end tell
```

複製・移動する

複製する

```
tell application "Adobe Illustrator"
  tell document 1
    duplicate text frame 1 to end --背面に複製
    duplicate text frame 1 to beginning --前面に複製
    duplicate path item 1 to end of layer 2
    --レイヤー 2の背面に複製
  end tell
end tell
```

移動する

```
tell application "Adobe Illustrator"
  tell document 1
    move text frame 1 to end --背面に移動
    move text frame 1 to beginning --前面に移動
    move path item 1 to end of layer 2
    --レイヤー 2の背面に移動
  end tell
end tell
```

オブジェクトが存在するか

```
tell application "Adobe Illustrator"
  tell document 1
    if exists path item 1 then
      delete path item 1
    end if
  end tell
end tell
```

オブジェクトを作成する

rectangleを作成

```
tell application "Adobe Illustrator"
```

```
tell document 1
  make new rectangle at beginning with properties ~
  {bounds:{0, 200, 200, 50}, fill color:{cyan:100, magenta:100, yellow:0, black:0}}
end tell
end tell
```

text frameを作成 (ポイントテキストとエリアテキスト)

```
tell application "Adobe Illustrator"
  tell document 1
    --ポイントテキスト
    make new text frame at beginning with properties ~
    {position:{0, 200}, contents:"あいうえお"}
    --エリアテキスト
    make new text frame at beginning with properties~
    {geometric bounds:{100, 90, 160, 180}, contents:"かきくけこ", kind:area text}
  end tell
end tell
```

path itemを作成 (entire pathで線の両端の座標を指定する。)

```
tell application "Adobe Illustrator"
  tell document 1
    make new path item at end with properties ~
    {stroke color:{cyan:C, magenta:M, yellow:Y, black:K}, ~
    stroke width:W, entire path:{{class:path point info, anchor:{X1, Y1}}, ~
    {class:path point info, anchor:{X2, y2}}}}
  end tell
end tell
```

変倍する

あるgroup itemをX=0pt Y=10pt 幅=100pt 高さ=200ptにしたいとき

```
set properties to {position:{0, 10}, width:100, height:200}
```

とやってしまうと大きさや場所は望みの場所に来るのですが線幅やアピアランスが消えてしまいます。解決するには

```
scale horizontal scale Wper vertical scale Hper line scale Wper ~
about top left with transforming objects, transforming fill patterns,~
transforming fill gradients and transforming stroke patterns
```

を使って変倍したあとpositionで移動させます。

scaleのオプションパラメータは以下のとおり

horizontal scale 120-左右の変倍比率 (120だと120%)

vertical scale 120- 天地の変倍比率

line scale 120- 線幅の変倍比率

about top left-左上基準で変倍 bottom rightで右下基準 centerで中心基準

with transforming objects--効果も変倍する

transforming fill patterns--塗りのパターンも変倍する

transforming fill gradients --グラデも変倍する

and transforming stroke patterns-線のパターンも変倍する

下記が変倍のサンプルスクリプトです。希望の大きさと現在の大きさの比率を出して変倍してから移動しています。

```
tell application "Adobe Illustrator"
  tell document 1
    tell group item 1
      set properties to {position:{0, 10}, width:100, height:200}
      --これをすると線幅やアピアランスが消えてしまう
      -- (Illustrator10では大丈夫)
      set myCB to control bounds
      --オブジェクトを選択してドラッグするときに表示される輪郭。
      --オブジェクトよりひとまわり大きい。
      set myGB to geometric bounds
      --オブジェクトのアンカーポイントぴったりのサイズ
      set myVB to visible bounds
      --オブジェクトに線の設定がされていれば線幅も加えたサイズになる。
      set myX to item 1 of myGB
      set myY to item 2 of myGB
      set myW to (item 3 of myGB) - myX
      set myH to (item 4 of myGB) - myY
      set Wper to 100 / myW * 100
      set Hper to 200 / myH * 100
      scale horizontal scale Wper vertical scale Hper line scale Wper ~
      about top left with transforming objects, transforming fill patterns,~
      transforming fill gradients and transforming stroke patterns
      set position to {0, 10}
    end tell
  end tell
end tell
```

document

作成する

```
tell application "Adobe Illustrator"
  make new document with properties (color space:CMYK)
end tell
```

保存せずに閉じる

```
tell application "Adobe Illustrator"
  close document 1 saving no
end tell
```

保存する

```
set fileRef to "Macintosh HD:test.eps" as text
tell application "Adobe Illustrator"
  --通常のAI保存
  save document 1 in file fileRef
  --eps保存
  save document 1 in file fileRef as eps with options ~
  {class:EPS save options, compatibility:Illustrator 10, embed all fonts:true}
  --PDF保存
  save document 1 in file fileRef as eps with options ~
  {class:PDF save options, compatibility:Acrobat 4, preserve editability:true}
end tell
```

gif書き出し

```
set SaveFolder to choose folder with prompt "保存フォルダを選択してください"
tell application "Adobe Illustrator"
  activate
  tell document 1
    set saveF to (SaveFolder as text) & "myPicture.gif"
    export to file saveF as GIF with options ~
    {class:GIF export options, horizontal scaling:200, vertical scaling:200}
  end tell
end tell
```

ロックされていない一旦すべて非表示にして、その後レイヤーを順に表示してgif出力。Web用のボタンをつくるのに便利なスクリプトです。

```
set SaveFolder to choose folder with prompt "保存フォルダを選択"
tell application "Adobe Illustrator"
  activate
  tell document 1
    set countLayer to count layer--レイヤーの数を数える
    repeat with t from 1 to countLayer--レイヤー分繰り返す
      if layer t is not locked then --もしロックされていないならば
        --ロックされていないレイヤーをぜんぶ非表示にする
        set visible of layer t to false
      end if
    end repeat
    repeat with t from 1 to countLayer--レイヤー分繰り返す
      if layer t is not locked then--もしロックされていないならば
        set visible of layer t to true--レイヤーを表示する
        set OutName to name of layer t--レイヤー名を取り出す
        set saveF to (SaveFolder as text) & (OutName as text) & ".gif"
        --レイヤー名でファイル名を作成
        export to file saveF as GIF--レイヤー名でGIF書き出し
        --export to file saveF as JPEG/Photoshop/SVG/PNG8/PNG24
        --などいろいろ選べる
        set visible of layer t to false--レイヤーを非表示にする
      end if
    end repeat
  end tell
end tell
```

layer

作成する

```
tell application "Adobe Illustrator"
  tell document 1
    make new layer at beginning with properties (name:"TESTレイヤー ")
  end tell
end tell
```

レイヤーのロック

```
tell application "Adobe Illustrator"
  tell document 1
    --レイヤー 1をロックする
    set locked of layer 1 to true
    --"画像"という名前のレイヤーをロックする
    set locked of layer "画像" to true
  end tell
```

```
end tell
```

path item

色や線を設定する

```
set pt to 2.83464
tell application "Adobe Illustrator"
  tell document 1
    tell path item 1
      set filled to false --塗りをなしにする
      set fill color to {cyan:0, magenta:0, yellow:0, black:0}
      --塗りを白にする
      set stroked to true --線の設定をありにする
      set stroke color to {cyan:0, magenta:100, yellow:50, black:0}
      --線をM100Y50にする
      set stroke width to 0.2 * pt
    end tell
  end tell
end tell
```

いっぺんに設定するにはset propertiesを使う

```
set pt to 2.83464
tell application "Adobe Illustrator"
  tell document 1
    tell path item 1
      set properties to {filled:false, ~
        fill color:{cyan:0, magenta:0, yellow:0, black:0}, ~
        stroked:true, ~
        stroke color:{cyan:0, magenta:100, yellow:50, black:0}, ~
        stroke width:0.2 * pt}
    end tell
  end tell
end tell
```

ガイドにする

```
tell application "Adobe Illustrator"
  tell document 1
    set guides of path item 1 to true
  end tell
end tell
```

text frame

縦組か横組かしらべます。

```
tell application "Adobe Illustrator"
  tell document 1
    tell layer 1
      tell text frame 1
        get text orientation --vertical=縦--horizontal=横
      end tell
    end tell
  end tell
end tell
```

paragraph 段落

段落に対して行揃えの設定をします。ほかにもreadingで行間の設定や段落スタイルの設定等が出来ます。すいません省略します。

```
tell application "Adobe Illustrator"
  tell document 1
    tell text frame 1
      tell paragraph 1
        set justification to right
        --right以外にも下記の種類がある
        --center/full justify/full justify last line center/
        --full justify last line left/full justify last line right
        --/left/right
      end tell
    end tell
  end tell
end tell
```

テキストボックスに1行になるよう長体

長体方法のうちのひとつ。テキストボックスに収まるように、まずテキストボックスの幅を測り、1行のときの文字列の幅も測り比率をだして長体をかけます。ポイントはテキストボックスを複製し、幅を10倍にします。(それで1行になると思われるため) さらに文字列をアウトラインし正確な文字幅を求めます。あとは比率を出してやるだけです。

もっと普通に長体をかける方法もありますがInDesignの長体を参照していただければと思います。

```
tell application "Adobe Illustrator"
  tell document 1
    set mySelObjList to selection --選択されたオブジェクトを調べる
    repeat with mySelObj in mySelObjList --選択分繰り返す
      if class of mySelObj is text frame then --もしtext frameなら
        my chouTai(mySelObj)
      end if
    end repeat
  end tell
end tell

on chouTai(mySelObj)
  tell application "Adobe Illustrator"
    set myW to width of mySelObj --text frameの幅を調べる
    --長体horizontal scaleを調べる (平体はvertical scale) --
    --Illustrator10ではscalingで縦横の比率がとれる
    set SCL to horizontal scale of paragraph 1 of mySelObj
    --横幅10倍サイズで複製する
    set dupTextObj to duplicate mySelObj to beginning of document 1 -
      with properties (width:myW * 10)
    --長体も10倍になるので長体を101%にする (なぜか100%にはならない)
    if SCL = 100 then
      set SCL to 101
    end if
    set horizontal scale of paragraph 1 of dupTextObj to SCL
    --行数を調べる
    set Pcount to count paragraph of dupTextObj
    if Pcount > 1 then --行数が2行以上なら
      display dialog "1行のテキストのみに有効です。"
      return --処理終了
    end if
    --行揃えを左にしておく。両揃えだと正しくはかれない
    set justification of paragraph 1 of dupTextObj to left
    --アウトライン作成
    set outLinePath to convert to paths dupTextObj
    --アウトラインの領域を調べる
    set myBounds to control bounds of outLinePath --control/visible
    --長さを取り出す
    set myW2 to (item 3 of myBounds) - (item 1 of myBounds) --(width of TBOX)
    delete outLinePath --アウトラインは消す
    if myW < myW2 then --もしアウトラインの方がtext frameよりも長ければ
      set myWhi to myW / myW2 --比率を求めて
      set SCL to SCL * myWhi - 1 --誤差が出るので1%マイナスする
      set horizontal scale of paragraph 1 of mySelObj to SCL
    end if
  end tell
end chouTai
```

character 文字

テキストにいろいろな設定をします。Illustrator10ならtext frameをtext art itemに変えて記述してください。

```
set pt to 2.83464
set Q to 0.25 * pt
tell application "Adobe Illustrator"
  tell document 1
    set mySelObjList to selection--選択されたオブジェクトを調べる
    repeat with mySelObj in mySelObjList--選択分繰り返す
      if class of mySelObj is text frame then--もしtext frameなら
        tell mySelObj
          --プロパティを調べる
          get properties of character 1
          --内容を変更する
          set contents to "あいうえお" & return & "かきくけこ"
          --フォントを変える
          set text font of characters 1 thru 5 to -
            text font "Osaka" of application "Adobe Illustrator"
          --サイズを変える
          set size of characters 1 thru 5 to 13 * Q
          --ベースラインシフト
          set baseline shift of character 1 of paragraph 1 to 2
          --横変倍
          set horizontal scale of character 1 of paragraph 1 to 200
          --縦変倍
          set vertical scale of character 2 of paragraph 1 to 200
        end tell
      end if
    end repeat
  end tell
end tell
```

```
--カーニング
set kerning of character 2 of paragraph 1 to -300
--塗り
set fill color of character 3 of paragraph 2 to -
  {cyan:0, magenta:50, yellow:100, black:0}
--線幅
set stroke weight of character 3 of paragraph 2 to 0.2
--線の色
set stroke color of character 3 of paragraph 2 to -
  {cyan:0, magenta:100, yellow:100, black:0}
--テキストフレームの幅を変更 (長体がかかる)
--これを行うと線幅等が消えてしまう要注意
set width to 10 * pt
end tell
end if
end repeat
end tell
end tell
```

オーバーフローを調べる

paragraphを合体したもの (全テキスト) とlineを合体したもの (見えているテキスト) を比べてオーバーフローしていたら警告する。

```
tell application "Adobe Illustrator"
  tell document 1
    repeat with tFrame in every text frame
      if kind of tFrame is {area text} then
        if my overflow(tFrame) then
          display dialog "オーバーフロー：" & contents of tFrame
        end if
      end if
    end repeat
  end tell
end tell

on overflow(tFrame)
  set OriginalDelimiters to AppleScript's text item delimiters
  set AppleScript's text item delimiters to ""
  tell application "Adobe Illustrator"
    set myLastP to (contents of every paragraph of tFrame) as string
    set myLastL to (contents of every line of tFrame) as string
    set AppleScript's text item delimiters to OriginalDelimiters
    if myLastP is not myLastL then
      set AppleScript's text item delimiters to OriginalDelimiters
      return true
    else
      set AppleScript's text item delimiters to OriginalDelimiters
      return false
    end if
  end tell
end overflow
```

文字を挿入する

文字列挿入。下記のスクリプトではtext frame "A"というようにtext frameを名前で指定しているのがポイント。名前をつけるにはレイヤーパレットでオブジェクトをダブルクリックすれば名前をつけられる。

```
tell application "Adobe Illustrator"
  tell document 1
    tell text frame "A"
      --1文字目を"あいうえお"にする
      set contents of character 1 to "あいうえお"
      --再終行を"かきくけこ"にする
      set contents of paragraph -1 to "かきくけこ"
    end tell
  end tell
end tell
```

テキスト全部保存

```
set TexData to ""
set FileName to (choose file name) as string
tell application "Adobe Illustrator"
  tell document 1
    set boxnum to count text frames
    repeat with N from 1 to boxnum
      set TexData to TexData & contents of text frame N & return & return
    end repeat
  end tell
end tell
```

```

end repeat
end tell
end tell

```

```

open for access file FileName with write permission
write TexData to file FileName
close access file FileName

```

連続数字を等幅半角字形にする

```

tell application "Adobe Illustrator"
tell document 1
set selectedItems to selection
if selectedItems is {} then
display dialog "text frameを選択ツールで選択してください。"
return
end if
repeat with selItem in selectedItems
if class of selItem is text frame then
set Cnum to count characters of selItem
repeat with N from 1 to Cnum - 1
set TXT1 to contents of character N of selItem
set TXT2 to contents of character (N + 1) of selItem
if TXT1 is in "0123456789," and TXT2 is in "0123456789," then
set alternate glyphs of character N of selItem to
half width --等幅半角字形
set alternate glyphs of character (N + 1) of selItem to
half width --等幅半角字形
end if
end repeat
end if
end repeat
end tell

```

画像を配置する埋め込みとリンク

下記は画像を配置してさらに50%にサイズ変更する。

```

set filePath to (choose file) as string
set X to 10
set Y to 10
set mySize to 50
set pt to 2.83
set mySize to mySize * pt
set X to X * pt
set Y to Y * pt
tell application "Adobe Illustrator"
--下記はリンク
set placedRef to make new placed item in document 1 with
properties {file path:alias filePath}
--下記は埋め込み
--set placedRef to make new group item at end with data alias filePath
set PW to width of placedRef
set PH to height of placedRef
if PW > PH then
set buf to PW
else
set buf to PH
end if
set Percent to mySize / buf
set W to PW * Percent
set H to PH * Percent
set properties of placedRef to {position:{X, Y}, height:H, width:W}
end tell

```

クリッピングマスクする

サンプルスクリプトをはしょって申し訳ないのですが下記はgroup itemに画像を配置しクリッピングマスクを適用する例です。

```

tell application "Adobe Illustrator"
tell document 1
make new group item at end with properties {name:KeyWord}
--レイヤーの一番後ろにグループが作られる
make new placed item at end of group item KeyWord with
properties {file path:alias Fpath, position:{X, Y}}
set clipped of group item KeyWord to true
end tell

```

```

end tell

```

画像の回転

```

回転
tell application "Adobe Illustrator"
tell document 1
rotate placed item 1 angle 5
end tell
end tell

```

トンボ作成

下記でトンボのカラーや線幅をカスタマイズすることができます。

```

global lineW, C, M, Y, K, pt
set lineW to 0.05
--色を入力単位%
set C to 100
set M to 100
set Y to 100
set K to 100
set pt to 2.83466796875

```

```

tell application "Adobe Illustrator"
tell document 1
set selectedItems to selection
if selectedItems is {} then
display dialog "四角をを選択ツールで選択してください。"
return
end if
repeat with selItem in selectedItems
set myB to geometric bounds of selItem
my makeTombo(myB)
end repeat
end tell
end tell

```

```

on makeTombo(myB)
set X1 to (item 1 of myB) / pt
set Y1 to (item 2 of myB) / pt
set X2 to (item 3 of myB) / pt
set y2 to (item 4 of myB) / pt
set myGroup to my makeGroup()
my makeLine(X1 - 13, Y1, X1 - 3, Y1, myGroup) --横罫左上
my makeLine(X1 - 13, Y1 + 3, X1, Y1 + 3, myGroup) --横罫左上
my makeLine(X1, Y1 + 13, X1, Y1 + 3, myGroup) --縦罫左上
my makeLine(X1 - 3, Y1 + 13, X1 - 3, Y1, myGroup) --縦罫左上
my makeLine(X1 - 13, y2, X1 - 3, y2, myGroup) --横罫左下
my makeLine(X1 - 13, y2 - 3, X1, y2 - 3, myGroup) --横罫左下
my makeLine(X1, y2 - 13, X1, y2 - 3, myGroup) --縦罫左下
my makeLine(X1 - 3, y2 - 13, X1 - 3, y2, myGroup) --縦罫左下
my makeLine(X2 + 13, Y1, X2 + 3, Y1, myGroup) --横罫右上
my makeLine(X2 + 13, Y1 + 3, X2, Y1 + 3, myGroup) --横罫右上
my makeLine(X2, Y1 + 13, X2, Y1 + 3, myGroup) --縦罫右上
my makeLine(X2 + 3, Y1 + 13, X2 + 3, Y1, myGroup) --縦罫右上
my makeLine(X2 + 13, y2, X2 + 3, y2, myGroup) --横罫右下
my makeLine(X2 + 13, y2 - 3, X2, y2 - 3, myGroup) --横罫右下
my makeLine(X2, y2 - 13, X2, y2 - 3, myGroup) --縦罫右下
my makeLine(X2 + 3, y2 - 13, X2 + 3, y2, myGroup) --縦罫右下
my makeLine(X1 - 13, Y1 + (y2 - Y1) / 2, X1 - 3, Y1 + (y2 - Y1) / 2, myGroup)
--横罫左中
my makeLine(X1 - 8, Y1 + (y2 - Y1) / 2 - 5, X1 - 8, Y1 + (y2 - Y1) / 2 + 5, myGroup)
--縦罫左中
my makeEllipse(X1 - 8 - 1.5, (Y1 + (y2 - Y1) / 2) + 1.5, 3, 3, myGroup)
--○丸左中
my makeLine(X2 + 13, Y1 + (y2 - Y1) / 2, X2 + 3, Y1 + (y2 - Y1) / 2, myGroup)
--横罫左中
my makeLine(X2 + 8, Y1 + (y2 - Y1) / 2 - 5, X2 + 8, Y1 + (y2 - Y1) / 2 + 5, myGroup)
--縦罫左中
my makeEllipse(X2 + 8 - 1.5, (Y1 + (y2 - Y1) / 2) + 1.5, 3, 3, myGroup)
--○丸右中
my makeLine(X1 + (X2 - X1) / 2, Y1 + 13, X1 + (X2 - X1) / 2, Y1 + 3, myGroup)
--縦罫中上
my makeLine(X1 + (X2 - X1) / 2 - 5, Y1 + 8, X1 + (X2 - X1) / 2 + 5, Y1 + 8, myGroup)
--横罫中上
my makeEllipse(X1 + (X2 - X1) / 2 - 1.5, Y1 + 8 + 1.5, 3, 3, myGroup) --○丸中上
my makeLine(X1 + (X2 - X1) / 2, y2 - 13, X1 + (X2 - X1) / 2, y2 - 3, myGroup)

```

```

--縦野中下
my makeLine(X1 + (X2 - X1) / 2 - 5, y2 - 8, X1 + (X2 - X1) / 2 + 5, y2 - 8, myGroup)
--横野中下
my makeEllipse(X1 + (X2 - X1) / 2 - 1.5, y2 - 8 + 1.5, 3, 3, myGroup) --○丸中下
end makeTombo

```

```

on makeLine(X1, Y1, X2, y2, myGroup)
set X1 to X1 * pt
set Y1 to Y1 * pt
set X2 to X2 * pt
set y2 to y2 * pt
set W to lineW * pt
tell application "Adobe Illustrator"
tell document 1
make new path item at end of myGroup with properties --
{stroke color:{cyan:C, magenta:M, yellow:Y, black:K}, --
stroke width:W, entire path:{{class:path point info, --
anchor:{X1, Y1}}, {class:path point info, anchor:{X2, y2}}}}
end tell
end tell
end makeLine

```

```

on makeEllipse(X1, Y1, W, H, myGroup)
tell application "Adobe Illustrator"
tell document 1
set X1 to X1 * pt
set Y1 to Y1 * pt
set myW to W * pt
set myH to H * pt
set W to lineW * pt
make new ellipse at end of myGroup with properties --
{filled:false, stroked:true, --
stroke color:{cyan:C, magenta:M, yellow:Y, black:K}, --
width:myW, height:myH, position:{X1, Y1}, stroke width:W}
end tell
end tell
end makeEllipse

```

```

on makeGroup()
tell application "Adobe Illustrator"
tell document 1
if exists group item "TOMBO" then
set myGroup to group item "TOMBO"
else
set myGroup to make group item at beginning --
with properties {name:"TOMBO"}
end if
end tell
end tell
return myGroup
end makeGroup

```

線幅を変える

全てのpath itemの線の太さ調べ、特定の太さの線の線幅を変更します。

```

tell application "Adobe Illustrator"
tell document 1
set pathCount to count path items
repeat with pathNum from 1 to pathCount
set pathStrokeWidth to stroke width of path item pathNum
if 0.6 < pathStrokeWidth and pathStrokeWidth < 0.7 then
set stroke width of path item pathNum to 0.283
else if 0.7 < pathStrokeWidth then
set stroke width of path item pathNum to pathStrokeWidth / 2
else
set stroke width of path item pathNum to 0.283
end if
end repeat
end tell
end tell

```

ほかできること、できそうにないこと

文字列の検索置換や段落・文字スタイルの適応（おそらくInDesignと同じ）タブの設定やシンボルの配置。さらに詳しく見ていくとグラデーションやブラシなどクラスはいっぱいあるので、AppleScriptでコントロールできるはず、だが省略。。

したくてもできない事はIllustratorでドキュメントを開くときリンクが見つからないとかエラーダイアログが出てしまうときがある。ダイアログを出さずにオープンする事ができない。cmなど特殊な文字は流し込みに失敗する。レイヤー名やオブジェクト名に日本語を使うとAppleScriptで値を取り出すとき文字化けする。アピアランスがコントロールできない。などなど。わたしの思い違いも多々あると思うのでおかしな点をご指摘ください。

処理が止まってしまうとき

ダイアログがでてAppleScriptの応答待ちで処理が止まってしまうときは、第2章のエラー処理などで紹介したtime outの設定をつかい1秒応答がなければIllustratorを落とす方法で逃げる。しかしその場合もAppleScriptからIllustratorにQuitコマンドを送っても応答しないのでshellから強制的に落とす。興味ある方は調べてみてください。

第6章

InDesignを コントロールする

InDesignもIllustrator同様AppleScriptでコントロールする事ができます。Illustratorよりも自動化しがいがあるかもしれません。JavaScriptで記述するとWin、Mac共に動かせる事ができるのですが、selectionをするだけでページ数の多いドキュメントではずいぶん時間がかかりました。そのためMacはAppleScriptでWinはVBScriptで書く方が高速です。InDesignには独自のインターフェイスが用意されていてAppleScriptで複雑なダイアログを出す事ができますが、今回は省略しています。興味ある方は調べてみてください。

選択されたオブジェクトを調べる

下記はまず選択されたアイテムを調べる。アイテムはたとえ1つだけ選択していてもリストで返ってくるので、リスト内のアイテム分繰り返す。（それが1つめのrepeat）そこから選択オブジェクトのクラスがtext columnならテキストが選択されていると見て処理を続ける。

```

tell application "Adobe InDesign CS2"
tell document 1
--文字スタイルを変数CSに入れておく
set CS to character style "文字スタイル 1"
--選択オブジェクトを変数selectedItemListに入れる
set selectedItemList to selection
if selectedItemList is {} then --もしなければ
return --処理終了
end if --選択されたオブジェクトを調べる（listで返ってくる）
repeat with selectedItem in selectedItemList
--選択オブジェクト分繰り返す。
set myClass to class of selectedItem
--もし選択オブジェクトが文字列なら
if myClass is text column then
--文字数分繰り返す
repeat with C from 1 to count characters of selectedItem
--もし文字が "0123456789"の中のどれかなら
if contents of character C of selectedItem is in "0123456789" then
--文字スタイルをつける
set applied character style of character C of selectedItem to CS
end if
end repeat
end if
end repeat
end tell

```

選択オブジェクトのクラスはよく使うものとして

```

if myClass is rectangle then--画像ボックス等四角です。
if myClass is text frame then--テキストフレーム
if myClass is text column then--テキスト
if myClass is table then--表

```

ほかにもわからないアイテムはpage itemで調べられます。ドキュメントの1ページ目に1つのオブジェクトを作り下記のようなスクリプトをつかうとクラスが何か調べる事が出来ます。

```
--ページアイテムのクラスを調べる時に使います。
```

```
tell application "Adobe InDesign CS2_J"
  tell page 1 of document 1
    get class of page item 1
  end tell
end tell
```

Document

作成する (Illustratorと違ってミリ単位で指定できる。)

```
tell application "Adobe InDesign CS2_J"
  make document with properties {page height:297, page width:210}
end tell
```

作成する2プロパティも同時設定

```
tell application "Adobe InDesign CS2_J"
  set myDocument to make document
  --ドキュメントを作成 (作成されたドキュメントは変数myDocumentに入る)
  set properties of document preferences of myDocument to ~
    {pages per document:10, pages per spread:1, ~
    page width:"210 mm", page height:"297 mm"}
  --プロパティの設定。document preferencesを使う。
  --pages per document      ドキュメントのページ数
  --pages per spread        スプレッド当たりのページ数 (2なら見開き)
end tell
```

保存せずに閉じる (Illustratorと同じ)

```
tell application "Adobe InDesign CS2_J"
  close document 1 saving no
end tell
```

保存する

```
set Fpath to "macintosh HD:test.indd"
tell application "Adobe InDesign CS2_J"
  save document 1 to Fpath
end tell
```

すべてのドキュメントをPDF書き出し

```
tell application "Adobe InDesign CS2_J"
  set Dcount to count document
  repeat with N from 1 to Dcount
    set myName to name of document N as string
    set myPath to (file path of document N) as string
    set myPDFpreset to PDF export preset ["PDF/X-1a"]
    set newFile to myPath & myName & ".pdf"
    export document N format PDF type to newFile using myPDFpreset
  end repeat
end tell
```

マスターページの横ガイドを消す

```
tell application "Adobe InDesign CS2_J"
  tell active document
    set spCount to count master spread
    --マスターページのスプレッド数を調べる
    repeat with N from 1 to spCount
      --マスターページ分繰り返す
      tell master spread N
        delete (every guide whose orientation is horizontal)
        --横ガイドを全て消す
      end tell
    end repeat
  end tell
end tell
```

ガイドを作成する

写真レイヤーにガイドを作成します。

```
tell application "Adobe InDesign CS2_J"
  set LayObj to layer "写真" of active document
  --写真レイヤーを変数LayObjへ入れる
  tell active document
    tell page 1
      set myGuide to make guide with properties ~
        {orientation:horizontal, location:40, item layer:LayObj}
      --写真レイヤーに横ガイドをひく
    end tell
  end tell
end tell
```

レイヤーを作成する

IMGレイヤーが存在するか調べ、なければ作成します。

```
tell application "Adobe InDesign CS2_J"
  tell active document
    if not (exists (layers whose name is "IMG")) then
      --同じ名前のレイヤーがあればエラーになるのでまず調べる
      make layer with properties {name:"IMG"}
      --なければレイヤーをつくる
    end if
  end tell
end tell
```

線の作成

"Y100M100"のカラーがあるか調べ、なければカラーを作成、そのあとで線を作ります。塗りなしはfill color:"None"で指定できます。(InDesign2.0ではfill color:none)

```
tell application "Adobe InDesign CS2_J"
  set LayObj to layer "LINE" of active document
  set Ws to 0.2
  set colName to "M100Y100"
  set C to 0
  set M to 100
  set Y to 100
  set K to 0
  tell active document
    try
      set myCol to color colName
      --変数myColにカラーオブジェクトを入れる
    on error--失敗したら
      set myCol to make color with properties ~
        {name:colName, color value:{C, M, Y, K}}
      --変数myColにカラーオブジェクトを作る
    end try
  end tell
  tell page 1 of active document
    make graphic line with properties ~
      {geometric bounds:{"12 mm", "10 mm", "109 mm", "10 mm"}, ~
      fill color:"None", stroke color:myCol, stroke weight:Ws, ~
      item layer:LayObj}
    --ライン作成。塗りがなしはfill color:"None"
    --2.0ではfill color:none
  end tell
end tell
```

ページ数を調べる

```
tell application "Adobe InDesign CS2_J"
  tell active document
    set PageCount to count page
  end tell
end tell
```

スプレッド1は何ページあるか

スプレッド1が見開きか単ページか調べる事は非常に重要です。それによってページにアイテムを配置する配置方法が変わってくるからです。

```
tell application "Adobe InDesign CS2_J"
  tell active document
    tell spread 1--スプレッド1の
      set sp1Count to count pages--ページ数を調べる
    end tell
  end tell
end tell
display dialog "スプレッド1は" & sp1Count & "ページです"
```

スクリプトラベル

ウィンドウメニューの自動化=>スクリプトラベルパレットを表示しオブジェクトを選択した状態でスクリプトラベルにオブジェクトの名前を入力する事ができる。例えばtext frameを選択し"Name"とスクリプトラベルに名前をつけると下記のように名前で指定できる。

```
tell application "Adobe InDesign CS2_J"
  tell document 1
    tell page 1
      tell text frame "Name"
```

```

set contents to "商品名"
end tell
end tell
end tell
end tell

```

masterページアイテムを上書きする

マスターページを上書き可能な状態にするのはスクリプトの定番手法。

```

tell application "Adobe InDesign CS2_J"
tell document 1
override item 1 of master page items of page 1 destination page page 1
end tell
end tell

```

トンボ作成

Illustratorとけっこう違う部分があります。ラインの作成、カラーの指定、グループの作り方などさまざまなスクリプト勉強になります。

```

global lineW, mypt, myCol
--グローバル変数の宣言。線幅とポイント数、色はグローバルにした
set lineW to 0.1
--線幅0.1
set mypt to 2.83
--ポイントは2.83
tell application "Adobe InDesign CS2_J"
tell active document
set selectedItems to selection
--選択アイテムを調べる
if selectedItems is {} then
--選択アイテムが無ければ
display dialog "フレームを1つだけ選択してください。"
return
end if
if not (exists (layers whose name is "TOMBO")) then
--トンボレイヤーがなければ
make layer with properties {name:"TOMBO"}--レイヤーを作る
end if
set myCol to color "Registration"
--変数myColはレジストレーションにしておく
if number of selectedItems is 1 then
--選択アイテムが1つなら
set selItem to item 1 of selectedItems
--選択アイテムの1つめを取り出す。(変数selectedItemsはリストだから)
set myParent to parent of selItem
--選択アイテムの親オブジェクトを調べる。
--(ページ番号が入ると思われる。)
--※こまかいエラー処理をしていないのはご容赦
set myBounds to geometric bounds of selItem
--選択アイテムの領域
set X to item 2 of myBounds--X座標
set Y to item 1 of myBounds--Y座標
set W to (item 4 of myBounds) - (item 2 of myBounds)--幅
set H to (item 3 of myBounds) - (item 1 of myBounds)--高さ
set gltm to {} as list--変数gltmをリストとして初期化しておく。
--左上
set gltm to gltm & my drowLine(X - 13, X - 3, Y, Y, myParent)
--オリジナル関数dowLineを呼び出し、
--結果(ラインオブジェクトが返ってくる)を変数gItemに入れる
set gltm to gltm & my drowLine(X - 13, X, Y - 3, Y - 3, myParent)
set gltm to gltm & my drowLine(X - 3, X - 3, Y - 13, Y, myParent)
set gltm to gltm & my drowLine(X, X, Y - 13, Y - 3, myParent)
--上センター
set gltm to gltm & my drowLine(X + W / 2, X + W / 2, Y - 13, Y - 3, myParent)
set gltm to gltm & my drowLine(X + W / 2 - 5, X + W / 2 + 5, Y - 7, Y - 7, myParent)
set gltm to gltm & my drowEllipse(X + W / 2 - 2, Y - 7 - 2, 4, 4, myParent)
--右上
set gltm to gltm & my drowLine(X + W + 3, X + W + 13, Y, Y, myParent)
set gltm to gltm & my drowLine(X + W, X + W + 13, Y - 3, Y - 3, myParent)
set gltm to gltm & my drowLine(X + W + 3, X + W + 3, Y - 13, Y, myParent)
set gltm to gltm & my drowLine(X + W, X + W, Y - 13, Y - 3, myParent)
--左センター
set gltm to gltm & my drowLine(X - 13, X - 3, Y + H / 2, Y + H / 2, myParent)
set gltm to gltm & my drowLine(X - 7, X - 7, Y + H / 2 - 5, Y + H / 2 + 5, myParent)
set gltm to gltm & my drowEllipse(X - 7 - 2, Y + H / 2 - 2, 4, 4, myParent)
--右センター
set gltm to gltm & my drowLine(X + W + 3, X + W + 13, Y + H / 2, Y + H / 2, myParent)

```

```

set gltm to gltm & my drowLine(X + W + 7, X + W + 7, Y + H / 2 - 5, Y + H / 2 + 5, myParent)
set gltm to gltm & my drowEllipse(X + W + 7 - 2, Y + H / 2 - 2, 4, 4, myParent)
--左下
set gltm to gltm & my drowLine(X - 13, X - 3, Y + H, Y + H, myParent)
set gltm to gltm & my drowLine(X - 13, X, Y + H + 3, Y + H + 3, myParent)
set gltm to gltm & my drowLine(X - 3, X - 3, Y + H + 13, Y + H, myParent)
set gltm to gltm & my drowLine(X, X, Y + H + 13, Y + H + 3, myParent)
--下センター
set gltm to gltm & my drowLine(X + W / 2, X + W / 2, Y + H + 13, Y + H + 3, myParent)
set gltm to gltm & my drowLine(X + W / 2 - 5, X + W / 2 + 5, Y + H + 7, Y + H + 7, myParent)
set gltm to gltm & my drowEllipse(X + W / 2 - 2, Y + H + 7 - 2, 4, 4, myParent)
--右上
set gltm to gltm & my drowLine(X + W + 3, X + W + 13, Y + H, Y + H, myParent)
set gltm to gltm & my drowLine(X + W, X + W + 13, Y + H + 3, Y + H + 3, myParent)
set gltm to gltm & my drowLine(X + W + 3, X + W + 3, Y + H + 13, Y + H, myParent)
set gltm to gltm & my drowLine(X + W, X + W, Y + H + 13, Y + H + 3, myParent)
make group with properties {group items:gltm}
--変数gItemをグループ化する。

```

```

end if
end tell
end tell

```

--オリジナル関数dowLine

```

on drowLine(X1, X2, Y1, Y2, myParent)

```

```

set buf to {Y1, X1, Y2, X2}

```

```

tell application "Adobe InDesign CS2_J"

```

```

tell myParent--変数myParentにはおそらくページが入っている。

```

```

make graphic line with properties {geometric bounds:buf,
fill color:"None", stroke color:myCol, stroke weight:lineW * mypt,
item layer:"TOMBO"}

```

```

--"TOMBO"レイヤーに線を引く

```

```

end tell

```

```

end tell

```

```

end drowLine

```

```

on drowEllipse(X1, Y1, W1, H1, myParent)

```

```

set buf to {Y1, X1, Y1 + H1, X1 + W1}

```

```

tell application "Adobe InDesign CS2_J"

```

```

tell myParent

```

```

make ovals with properties {geometric bounds:buf,
fill color:"None", stroke color:myCol, stroke weight:lineW * mypt,
item layer:"TOMBO"}
--マルを作成する

```

```

end tell

```

```

end tell

```

```

end drowEllipse

```

ライブラリの配置

ライブラリの配置はあっけないほど簡単です。下記では"Nomio1"という名前のライブラリを配置しています。

```

tell application "Adobe InDesign CS2_J"
place asset asset "Nomio1" of library 1 on document 1
end tell

```

スニペットの配置と書き出し

スニペットとはInDesignでオブジェクトを選択しDesktopにドラッグ&ドロップした時にできるオブジェクトです。これを別ドキュメントにドロップすれば同じ場所に同じオブジェクトが再現できます。ライブラリのようなものですが1つのオブジェクトごとにファイルで管理できるので自動組版システムを作るにはマスターページよりも使いやす場合があります。配置するのは下記のとおりで簡単です。

```

set mySnippet to "Macintosh HD:日程.indd" as string
tell application "Adobe InDesign CS2_J"
tell page 1 of document 1
place mySnippet
end tell
end tell

```

こちらは出力です。出力も簡単なのですが書き出す時にレイヤーが非表示だと非表示のスニペットになってしまいます。要注意です。

```

tell application "Adobe InDesign CS2_J"
tell document 1
tell layer 2 of page 1
export text frame 1 format InDesign snippet to mySnippet
end tell
end tell

```

```
end tell
```

画像のフィット

```
tell application "Adobe InDesign CS2_1"
  tell active document
    fit rectangle 1 given content to frame
    -- fit rectangle 1 given content to frame/内容をフレームにあわせる
    -- fit rectangle 1 given center content/内容を中央に
    -- fit rectangle 1 given proportionally/短辺合わせ
    -- fit rectangle 1 given frame to content/フレームをあわせる
    -- fit rectangle 1 given fill proportionally/長辺合わせ
  end tell
end tell
```

選択画像の回転

こういう細かいスクリプトはアプリケーションのPresetsのScriptsフォルダに入れておくとウィンドウメニューの自動化=>スクリプトパレットに表示されるのでかなり便利。ショートカットも割り当てることができる。

```
tell application "Adobe InDesign CS2_1"
  tell active document
    tell selection
      set myAngle to rotation angle as real
      --現在の回転を取り出す
      set rotation angle to myAngle - 0.1
      --現在の回転から-0.1ひいたものにする。
    end tell
  end tell
end tell
```

画像の配置

単純に配置するだけなら下記で十分

```
set inputFile to "Macintosh HD:test.jpg"
tell application "Adobe InDesign CS2_1"
  tell active document
    tell page 1
      set EPSobj to place inputFile
    end tell
  end tell
end tell
```

領域を作っておきその領域に配置した後縮小する。

```
set inputFile to "Macintosh HD:test.jpg"
set X to 10
set Y to 20
set W to 50
set H to 60
set myPer to 70
tell application "Adobe InDesign CS2_1"
  tell active document
    tell page 1
      set myRect to make rectangle at beginning with properties {
        geometric bounds:{Y, X, Y + H, X + W}, stroke weight:0,
        fill color:"None", stroke color:"None"}
      --まずはボックスを作る
      set EPSobj to place inputFile on myRect
      --そのボックスに画像を入れる
      tell EPSobj
        set properties to {vertical scale:myPer, horizontal scale:myPer}
        --画像のサイズを変更する
      end tell
      fit myRect given center content
      --画像をセンターに入れる
    end tell
  end tell
end tell
```

選択ファイルを配置 & 保存

ファイルを選択しそのファイルを新規ドキュメントに配置し適切なサイズにドキュメントサイズを変更した後選択ファイルのフォルダに保存する。

```
set this_item to choose file with prompt "配置ファイルを選択"
set Fpath to this_item as string
--変数Fpathにフルパスを入れる
set Fpath to Fpath & ".indd"
--フルパスに拡張子.inddを付けておく
```

```
set Finfo to info for this_item
--ファイルインフォメーションを調べる
set FType to file type of Finfo
--ファイルインフォメーションのファイルタイプを変数FTypeに入れる
set Fkind to kind of Finfo
--ファイルインフォメーションのファイルカインドを変数FKindに入れる
--↓下記のような条件があれば。。。意外と条件にあわないEPSが多い。
if FType is "EPSF" or FType is "EPSP" or FType is "PDF" or FType is "8BPS" or
  FType is "TEXT" or Fkind is "Adobe Photoshop ファイル" or
  FType is "TIFF" or Fkind is "TIFF 書類" or Fkind is "EPS ファイル" then
  my setEPS2(this_item, Fpath)
end if
```

```
--↓オリジナル関数setEPS2
on setEPS2(this_item, Fpath)
  tell application "Adobe InDesign CS2_1"
    --ドキュメントを作成
    make document
    tell active document
      set myRect to make rectangle at beginning with properties {
        geometric bounds:{0, 0, 100, 100},
        stroke weight:0, fill color:"None", stroke color:"None"}
      --とりあえずrectangle (ボックス)を作成100×100サイズ
      --線や塗りはなし
      set EPSobj to place this_item on myRect
      --ファイルを作成したボックスに配置する
      -- (配置画像を変数EPSobjに入れる)
      set myB to geometric bounds of EPSobj
      --配置画像 (変数EPSobj) の領域を調べる
      --geometric bounds は{Y1,X1,Y2,X2}で返ってくる
      set H to (item 3 of myB) - (item 1 of myB)--高さを取り出す
      set W to (item 4 of myB) - (item 2 of myB)--幅を取り出す
      if H < 297 and W < 210 then --A4縦より小さいなら
        set PW to 210
        set PH to 297
      else if H < 210 and W < 297 then --A4横より小さいなら
        set PW to 297
        set PH to 210
      else if H < 364 and W < 257 then --B4縦より小さいなら
        set PW to 257
        set PH to 364
      else if H < 257 and W < 364 then --B4横より小さいなら
        set PW to 364
        set PH to 257
      else --それ以外
        if H > W then --A3縦
          set PW to 297
          set PH to 420
        else --A3横
          set PW to 420
          set PH to 297
        end if
      end if
      set properties of document preferences to {page height:PH, page width:PW}
      --ドキュメントのサイズを変更する。
      set geometric bounds of myRect to {0, 0, PH, PW}
      --rectのサイズをページサイズにする
      fit myRect given center content
      --画像をセンターあわせにする。
      save to Fpath
      --Fpathに保存する。
    end tell
  end tell
end setEPS2
```

EPSのページ配置

フォルダ内のEPSファイルを次々にページに配置していきます。InDesignでrectangleを作る場合、座標をスプレッドで指定するため、(横組で右ページX座標が0ならX=210で指定する。) まずページ自体の座標を調べその座標を基準にrectangleを作成しています。

```
set myPer to 100
set NURI to 3 --塗り足し (0にすると塗り足しなし)

tell application "Adobe InDesign CS2_1"
  set myVersion to version
```

```

if "4.0." is not in myVersion then
--InDesignのバージョン確認
display dialog "Adobe InDesign CS2で使用してください。-
InDesign2.0が起動している場合は一旦終了し、スクリプトも-
再起動してください。"
return
end if
end tell

set myFol to choose folder with prompt "フォルダを選択"
set myFol to myFol as string
set myPage to 1--変数myPageを1にしておく。
repeat with myFile in list folder myFol without invisibles
--フォルダ内のアイテム分繰り返す
--以降のif文までは選択ファイル配置&保存を参照
set this_item to (myFol & myFile) as alias
set FType to file type of (info for this_item)
set Fkind to kind of (info for this_item)
if FType is "EPSF" or FType is "EPSP" or FType is "PDF" or -
FType is "8BPS" or FType is "TEXT" or -
Fkind is "Adobe Photoshop ファイル" or FType is "TIFF" or -
Fkind is "TIFF 書類" or Fkind is "EPS ファイル" then
my setEPS(myPage, myFol & myFile, NURI, myPer)
--オリジナル関数setEPSを呼び出す
set myPage to myPage + 1
--変数myPageに1をプラスする。
end if
end repeat

on setEPS(myPage, inputFile, NURI, myPer)
try
tell application "Adobe InDesign CS2_J"
tell active document
if not (exists of page myPage) then
make page at end
end if
set myB to bounds of page myPage
set Y to (item 1 of myB) - NURI
set X to (item 2 of myB) - NURI
set H to (item 3 of myB) - (item 1 of myB) + NURI * 2
set W to (item 4 of myB) - (item 2 of myB) + NURI * 2
tell page (myPage)
set myRect to make rectangle at beginning with properties -
{geometric bounds:{Y, X, Y + H, X + W}, stroke weight:0, -
fill color:"None", stroke color:"None"}
set EPSobj to place inputFile on myRect
tell EPSobj
set properties to -
{vertical scale:myPer, horizontal scale:myPer}
end tell
fit myRect given center content
end tell
end tell
end tell
end try
end setEPS

```

PDFのページ配置

まずAcrobatでPDFを開きページ数を数えます。そのあとでInDesignのドキュメントにPDFを1ページずつ配置していきます。さきほどのEPS配置と良く似ていますがPDFの処理が付け加えられています。

```

set myPer to 100
set NURI to 3--塗り足し (0にすると塗り足しなし)
set ANS to display dialog "ドキュメントの何ページ目から配置しますか?" -
default answer "1"
set inddPage to text returned of ANS
set inddPage to inddPage as integer
set pdfPage to 1
set inputFile to choose file with prompt "pdfファイルを選択"
set Fkind to kind of (info for inputFile)
if Fkind is "PDF 書類" or Fkind is "Adobe PDF document" then
--AcrobatでPDFを開きページ数を数える
tell application "Adobe Acrobat 7.0 Professional"
open inputFile
set AllPage to count page of document 1
close document 1

```

```

end tell
repeat AllPage times --AllPage分繰り返す。
set myError to my setEPS(pdfPage, inddPage, inputFile, NURI, myPer)
set pdfPage to pdfPage + 1
set inddPage to inddPage + 1
if myError is "ERROR" then
exit repeat
end if
end repeat
end if

--オリジナル関数setEPSの宣言
on setEPS(pdfPage, inddPage, inputFile, NURI, myPer)
--ここより下でend tryより上を試す (エラーが出てもしまらないようにするため)
try
--"Adobe InDesign CS2_J"を呼び
tell application "Adobe InDesign CS2_J"
set page number of PDF place preferences to pdfPage
set PDF crop of PDF place preferences to crop media
--"最前面のドキュメント"を呼び
tell active document
--もし変数myPageのページが存在しないなら
if not (exists of page inddPage) then
--最後にページをつくる
make page at end
end if
--ページの領域を変数myBに入れる{Y1,X1,Y2,X2}で返ってくる
set myB to bounds of page inddPage
--Y1より変数NURIを引いた分だけ変数Yに入れる
set Y to (item 1 of myB) - NURI
--X1より変数NURIを引いた分だけ変数Xに入れる
set X to (item 2 of myB) - NURI
--Y2よりY1を引き変数NURI*2を足した分だけ変数Hに入れる
set H to (item 3 of myB) - (item 1 of myB) + NURI * 2
--X2よりX1を引き変数NURI*2を足した分だけ変数Wに入れる
set W to (item 4 of myB) - (item 2 of myB) + NURI * 2
--"ページ"変数myPageを呼び
tell page (inddPage)
--画像ボックスを作り変数myRectに入れる
-- (画像ボックスのプロパティは領域{Y, X, Y + H, X + W},
--線幅0mm, ぬりなし、線の色もなし)
set myRect to make rectangle at beginning with properties -
{geometric bounds:{Y, X, Y + H, X + W}, stroke weight:0, -
fill color:"None", stroke color:"None"}
--変数myRectの中にinputFileを配置し結果を変数EPSobjに入れる
set PDFobj to place inputFile on myRect
--EPSobjを呼び
tell PDFobj
--水平方向の比率と垂直方向の比率を変数myPerにする
set properties to {vertical scale:myPer, horizontal scale:myPer}
end tell
--myRectにセンター揃えの命令を送る
fit myRect given center content
end tell
end tell
end tell
return ""
on error
return "ERROR"
end try
end setEPS

```

名刺の10枚面付け

名刺を10枚分A4の用紙に面付けします。X,Yの数字を変更すれば位置を変える事ができます。

```

set inputFile to choose file with prompt "ファイルを選択してください"
tell application "Adobe InDesign CS2_J"
tell page 1 of document 1
set X to 14
set Y to 11
set W to 91
set H to 55
repeat with N1 from 1 to 2
set X2 to X + W * (N1 - 1)
repeat with N2 from 1 to 5
set Y2 to Y + H * (N2 - 1)

```

```

make new rectangle at beginning with properties ~
    (geometric bounds:{Y2, X2, Y2 + H, X2 + W}, ~
    stroke color:"None", fill color:"None")
set placeFile to place inputFile on rectangle 1
fit rectangle 1 given center content
end repeat
end repeat
end tell
end tell

```

名刺の10枚面付け

上と同じスクリプトですが一生懸命考えると6行になりました。

```

set inputFile to choose file with prompt "ファイルを選択してください"
repeat with N1 from 1 to 10
    tell application "Adobe InDesign CS2_1" to make new rectangle at beginning~
        of page 1 of document 1 with properties ~
        (geometric bounds:{(10 + ((N1 - 1) div 2) * 55), (15 + (N1 mod 2) * 91), ~
        (10 + ((N1 - 1) div 2) * 55) + 55, (15 + (N1 mod 2) * 91) + 91}, ~
        stroke color:"None", fill color:"None")
    tell application "Adobe InDesign CS2_1" to place inputFile on rectangle 1 ~
        of page 1 of document 1
    tell application "Adobe InDesign CS2_1" to fit rectangle 1 of page 1 ~
        of document 1 given center content
end repeat

```

縦組にする

```

tell application "Adobe InDesign CS2_1"
    tell active document
        tell page 1
            set story orientation of story preferences of ~
            parent story of text frame 1 to vertical
        end tell
    end tell
end tell

```

特殊なノンブル

WORDのPDFが元原稿で、それにあわせて全角数字でどうしてもノンブルを入れなくては行けない時のサンプルです。これは単ページのドキュメントのセンター下にテキストフレームを作りそこに全角数字のノンブルを入れます。

```

tell application "Adobe InDesign CS2_1"
    tell active document
        set PS to paragraph style "ノンブル"
        set countPage to count pages
        repeat with N from 1 to countPage
            tell page N
                set myStr to my HanToZen(N) as string
                --ページ番号をオリジナル関数HanToZenを使って全角にする。
                set Tobj to make text frames at beginning with properties ~
                (contents:myStr, visible bounds:~
                {297 - 15, 210 / 2 - 20, 297 - 10 + 5, 210 / 2 + 20})
                --テキストフレームを作成し文字を入れる
                set applied paragraph style of paragraph 1 of Tobj to PS
                --スタイルを当てる。
            end tell
        end repeat
    end tell

--オリジナル関数HanToZen
on HanToZen(myStr)
    set findList to {"1", "2", "3", "4", "5", "6", "7", "8", "9", "0"}
    --半角文字リストを作成
    set repList to {" 1", " 2", " 3", " 4", " 5", " 6", " 7", " 8", " 9", " 0"}
    --全角文字リストを作成
    repeat with N from 1 to count findList
        set myStr to my FindAndRep(item N of findList, item N of repList, myStr)
        --リスト分検索置換
    end repeat
    return myStr
end HanToZen

```

```

--オリジナル関数FindAndRep (検索置換)
on FindAndRep(findStr, repStr, motoStr)

```

```

set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {findStr}
set motoStr to text items of motoStr
set AppleScript's text item delimiters to {repStr}
set motoStr to motoStr as string
set AppleScript's text item delimiters to OriginalDelimiters
return motoStr
end FindAndRep

```

段落1行に長体

下記は段落(paragraph)の文字列と見せかけの行(line)の文字列を比べ、同じになるまで3%きざみで長体をかけます。

```

tell application "Adobe InDesign CS2_1"
    tell document 1
        set selectedItems to selection
        if selectedItems is {} then
            return
        end if
        set selItem to item 1 of selectedItems
        set myPer to 97
        --まずスタートパーセントを97にする。
        repeat
            --繰り返し
            set pStr to every paragraph of selItem as string
            --段落のテキストを取り出す
            set Lstr to line 1 of selItem as string
            --ライン (みかけの行) のテキストを取り出す
            if pStr is not Lstr then--もし段落と見かけの行が違うなら
                --変倍をかける
                set horizontal scale of paragraph 1 of selItem to myPer
                --パーセントから3を引いておく
                set myPer to myPer - 3
            else--段落と見かけの行が同じなら
                exit repeat--繰り返しを抜ける
            end if
        end repeat
    end tell
end tell

```

改ページ文字を入れる

ページの最後に「改頁」という文字を挿入します。なんでこんなことするかというとこのテキストをマイクロソフトのワードにペーストし改頁を本物の改ページに置き換えます。するとワードの索引作成機能を使えるというわけです。ワードの方が便利なきももあるというサンプルですね。ちなみにワードの索引作成機能は私は良く知らないので質問しないでください。

このスクリプトのポイントはページの最初から処理を行うと「改頁」を入れた瞬間テキストが次ページにあふれどこが改ページなのかわからなくなるのでページの後ろから処理していったる事。

```

tell application "Adobe InDesign CS2_1"
    tell document 1
        set PageCount to count pages
        repeat with N from 0 to PageCount - 1
            set P to PageCount - N
            tell page P
                set Tcount to count text frames
                --テキストフレームの数を数える
                repeat with T from 1 to Tcount
                    set myB to visible bounds of text frame T
                    set W to (item 4 of myB) - (item 2 of myB)
                    set H to (item 3 of myB) - (item 1 of myB)
                    --テキストフレームの大きさを調べて
                    if H < W and 162 < H then
                        --大きければ本文のテキストフレームとする。
                        set myCha to character -1 of text frame T
                        --最後の文字を取り出す。
                        set character -1 of text frame 1 to myCha & "<改頁>"
                        --最後の文字を最後の文字+<改頁>にする。
                    end if
                end repeat
            end tell
        end repeat
    end tell
end tell

```

searchをつかった検索置換

searchはInDesignの検索置換です。かなりいろいろできるのではまります。下記スクリプトでは(1)をOpenTypeの(1)にしています。必ず検索置換設定を記録して最後に復帰しておく事と、検索前に検索置換設定を空にしておくことが必要です。

```
tell application "Adobe InDesign CS2_J"
  set oldFindPreferences to properties of find preferences
  --検索設定を変数oldFindPreferencesに入れておく
  set oldChangePreferences to properties of change preferences
  --置換設定を変数oldFindPreferencesに入れておく
  tell active document
    tell page 1
      my FindRep1(text frame 1, "(1)", "<2474>")
      --オリジナル関数を呼び出す
    end tell
  end tell
  set properties of find preferences to oldFindPreferences
  --検索設定を元に戻す
  set properties of change preferences to oldChangePreferences
  --置換設定を元に戻す
end tell

on FindRep1(TF, findStr, repStr)
  tell application "Adobe InDesign CS2_J"
    set find preferences to nothing
    --検索設定をなしにする。
    set change preferences to nothing
    --置換設定をなしにする。
    tell active document
      search TF for findStr replacing with repStr
    end tell
  end tell
end FindRep1
```

段落スタイル

同じくsearchを使って検索しているのですが、下記は"■数字."の文字列を検索し見つければ"■見出し"という段落スタイルを付けています。

```
tell application "Adobe InDesign CS2_J"
  set oldFindPreferences to properties of find preferences
  set oldChangePreferences to properties of change preferences
  tell active document
    tell page 1
      my setPstyle(text frame 1, "■^9.", "■見出し")
    end tell
  end tell
  set properties of find preferences to oldFindPreferences
  set properties of change preferences to oldChangePreferences
end tell

on setPstyle(TF serchString, styleName)
  tell application "Adobe InDesign CS2_J"
    set find preferences to nothing
    set change preferences to nothing
    tell active document
      set PS1 to paragraph style styleName
      search TF with find attributes {find text:serchString} --
        with change attributes {applied paragraph style:PS1}
    end tell
  end tell
end setPstyle
```

文字スタイル

下記は文字スタイルのサンプルです。

```
tell application "Adobe InDesign CS2_J"
  tell active document
    set CS1 to character style "文字スタイル 1"
    tell text frame 1 of page 1
      set applied character style of characters 1 thru 2 to CS1
    end tell
  end tell
end tell
```

文字スタイルのフォントを変更

文字スタイルに設定されているフォントを変更します。

```
tell application "Adobe InDesign CS2_J"
  tell document 1
    tell character style "英語"
      set applied font to "Times New Roman PS MT"
      set point size to "6 Q"
    end tell
  end tell
end tell
```

字取り変更

1ビームカーソルでテキストを選択し、今9字取りの文字列を7字取りに変更といったことができるスクリプト

```
tell application "Adobe InDesign CS2_J"
  tell document 1
    --最小3文字
    set ANS to display dialog "何字取りを何字取りにしますか? --
      " default answer "9/7" buttons {"OK"}
    set myStr to text returned of ANS
    set motoJIDORI to 9
    set newJidori to 7
    set motoJIDORI to my nthFields(myStr, "/", 1)
    set newJidori to my nthFields(myStr, "/", 2)
    set selectedItems to selection
    if selectedItems is {} then
      return
    end if
    set selItem to item 1 of selectedItems
    tell selItem
      set jidori of every character whose jidori is motoJIDORI to newJidori
    end tell
  end tell
end tell

on nthFields(myStr, sep, num)
  set OriginalDelimiters to AppleScript's text item delimiters
  set AppleScript's text item delimiters to {sep}
  set myStr to text items of myStr
  set AppleScript's text item delimiters to OriginalDelimiters
  return item num of myStr
end nthFields
```

字形変更

選択されたテキストフレーム内の文字を1文字づつあたっていく連続数字なら等幅半角字形に1ケタ数字なら等幅半角字形で前後に4分アキを入れる。本当に1文字づつあたっていく超ベタなスクリプト。ほんとはずかしい。serchを使った方が早いですね。興味ある方はチャレンジしてみてください。

```
set findList to {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"}
set findListCount to count findList

tell application "Adobe InDesign CS2_J"
  tell document 1
    set selectedItems to selection
    if selectedItems is {} then
      display dialog "テキストフレームを選択ツール (黒矢印) で--
        選択してください。"
      return
    end if
    repeat with selItem in selectedItems
      if class of selItem is not text frame then
        display dialog "テキストフレームを選択ツール (黒矢印) で--
          選択してください。"
        return
      else
        set Pnum to count paragraphs of selItem
        repeat with P from 1 to Pnum
          set myPara to paragraph P of selItem
          set FLG to false
          repeat with FL from 1 to findListCount
            if item FL of findList is in myPara then
              set FLG to true
              exit repeat
            end if
          end repeat
          if FLG is true then
```

```

set Cnum to count characters of paragraph P of selItem
repeat with N from 1 to Cnum
  if N = 1 then
    set TXTpre to "★"
  else
    set TXTpre to contents of character (N - 1) of ↵
    paragraph P of selItem
  end if
  set TXT to contents of character N of paragraph P of selItem
  if N = Cnum then
    set TXTaf to "★"
  else
    set TXTaf to contents of character (N + 1) of ↵
    paragraph P of selItem
  end if
  if TXTpre is in "0123456789 0 1 2 3 4 5 6 7 8 9,." and ↵
  TXT is in "0123456789 0 1 2 3 4 5 6 7 8 9,." then
    set glyph form of character (N - 1) of paragraph P of ↵
    selItem to monospaced half width form --等幅半角字形
    set glyph form of character N of paragraph P of ↵
    selItem to monospaced half width form --等幅半角字形
  else if TXTpre is not in "0123456789 0 1 2 3 4 5 6 7 8 9"↵
  and TXT is in "0123456789 0 1 2 3 4 5 6 7 8 9"↵
  and TXTaf is not in "0123456789 0 1 2 3 4 5 6 7 8 9" then
    set properties of character N of paragraph P ↵
    of selItem to {glyph form:monospaced half width form,↵
    trailing aki:0.25, leading aki:0.25}
  end if
end repeat
end if
end repeat
end if
end repeat
end tell
display dialog "終了しました"

```

text frameに (Tag付き) テキストを読み込む

スクリプトラベルで"L01spec"と名前をつけたtext frameにファイルtempL01.txtを流し込みます。

```

my test("L01spec", "TAKEUCHI:作業中:temp:tempL01.txt")

on test {myLabel, FilePath}
  set myLabel to myLabel as string
  set FilePath to FilePath as string
  tell application "Adobe InDesign CS2_J"
    tell text frame myLabel of document 1
      tell text 1
        place FilePath
      end tell
    end tell
  end tell
  return "true"
end run

```

ルビ

```

tell application "Adobe InDesign CS2_J"
  tell document 1
    tell text frame 1
      set contents to "竹内 亨"
      set ruby flag of character 1 to yes
      set ruby string of character 1 to "たけ"
      set ruby flag of character 2 to yes
      set ruby string of character 2 to "うち"
      set ruby flag of character 4 to yes
      set ruby string of character 4 to "とおる"
    end tell
  end tell
end tell

```

テキストを表にペースト

クリップボードのテキストをInDesignCSの表の選択部分にペーストします。行や列の概念は無く3行4列の表なら単純に1番目から12番目までのセルにテキストを入れます。もしエクセルの表が2行6列でもセルの数は同じな

ので内容は入れ替わります。

```

set myTable to the clipboard
set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {return}
set myTable to text items of myTable
set AppleScript's text item delimiters to {tab}
set myTable to myTable as string
set myTable to text items of myTable
set AppleScript's text item delimiters to OriginalDelimiters

tell application "InDesign CS_J"
  set contents of selection to myTable
  --表組の内容を入れ替える。
end tell

```

選択された表を調べる

表の選択部分は表全体の何行目で何列目を調べたい時、下記スクリプトで選択された表の中の1つめのセルの名前を取り出せます。"1:3"といった感じで返ってくるので"区切りで分割します。もっと良い方法がありそうだけどわからなかった。。。

```

tell application "Adobe InDesign CS2_J"
  --選択した表のcell 1は表全体の何行目が調べる
  set myRow to index of parent row of cell 1 of selection as integer
  --選択した表のcell 1の名前を調べる ("1:3") とかで返ってくる
  get name of cell 2 of selection
end tell

```

表の罫線を変更する

表の罫線を変更します。選択したセルの下罫線だけを変更する設定でカスタマイズすればいろんなバリエーションを作れます。

```

set OriginalDelimiters to AppleScript's text item delimiters
set myWaight to "0.0 mm"
tell application "Adobe InDesign CS2_J"
  tell document 1
    set selectedItems to selection
    if selectedItems is {} then
      return
    end if
    set selItem to item 1 of selectedItems
    set startCell to name of selItem
    set AppleScript's text item delimiters to {":"}
    set startCell to text items of startCell
    set startX to item 1 of startCell
    set startY to item 2 of startCell
    set indRowCount to count rows of selItem
    set indColumnCount to count columns of selItem
    --set left edge stroke weight of column (startX) of selItem to myWaight
    --set right edge stroke weight of ↵
    column (startX + indColumnCount - 1) of selItem to myWaight
    set bottom edge stroke weight of row (startY + indRowCount - 1) of selItem↵
    to myWaight
  end tell
end tell
set AppleScript's text item delimiters to OriginalDelimiters

```

表内のあふれた文字に長体

表内の文字のオーバーフローに長体をかけます。オーバーフローのチェックにはoverflowsというプロパティを調べます。これはtext frameでも同じです。

```

tell application "Adobe InDesign CS2_J"
  set mySele to class of selection --選択されたアイテムが
  if mySele is cell or mySele is table then --もしセルかテーブルなら
    set secount to count cells of selection --セルの数を数える
    repeat with C from 1 to secount --セルの数分くりかえす
      if overflows of cell C of selection then --セルのテキストがあふれていたら
        repeat with myPer from 100 to 30 by -2 --100%から30%まで2%刻みで
          set horizontal scale of every paragraph of cell C of ↵
          selection to myPer --長体をかけていく
        if not overflows of cell C of selection then
          --セルのテキストが収まったら
          exit repeat --くりかえしを抜ける
        end if
      end repeat
    end repeat
  end repeat
end tell

```

```

end if
end repeat
else
display dialog "表を選択してください"
end if
end tell

```

疑似囲み罫

ページ内の数段落分をコラムのように囲む事はInDesignではできません。かわりに1セルだけの表を使うのですが、めんどろなのでスクリプトで選択範囲を表にします。それでもページをまたぐ囲みはできないのですが。。

```

tell application "Adobe InDesign CS2_J"
tell document 1
set PS to paragraph style "本文囲み内"
set newTable to convert to table selection column separator "☆" ~
row separator "☆"
一行区切り列区切りを (ありえなさそうな文字) "☆"にしています。
set applied paragraph style of paragraphs 1 thru -1 of ~
cell 1 of newTable to PS
set bottom edge stroke weight of cell 1 of newTable to 0.1
set right edge stroke weight of cell 1 of newTable to 0.1
set left edge stroke weight of cell 1 of newTable to 0.1
set top edge stroke weight of cell 1 of newTable to 0.1
set bottom inset of cell 1 of newTable to 3.25
set top inset of cell 1 of newTable to 3.25
set left inset of cell 1 of newTable to 0
set right inset of cell 1 of newTable to 0
set width of cell 1 of newTable to 130
end tell
end tell

```

表の行を追加する

```

tell application "Adobe InDesign CS2_J"
tell document 1
tell text frame 1
make row of table 1
set rowCount to count rows of table 1
end tell
end tell
end tell

```

値段表の値段を一律料金アップする

表の中の選択部分の値段を取り出し入力した値をプラスします。値段の,"のつけかたがベタです。。

```

set ANS to display dialog "値段を入力してください。" default answer "1000"
set plusNum to text returned of ANS
set plusNum to my replaceAll(plusNum, "円", "")
set plusNum to my replaceAll(plusNum, ",", "")
set plusNum to plusNum as integer

```

```

set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {""}

```

```

tell application "Adobe InDesign CS2_J"
tell active document
tell selection
set cellCount to count cells
repeat with N from 1 to cellCount
set myData to contents of cell N
set myData to my replaceAll(myData, "円", "")
set myData to my replaceAll(myData, ",", "")
set myData to myData as integer
set myData to myData + plusNum
set myData to myData as string
if length of myData = 4 then
set myData to (character 1 of myData & "," & ~
characters 2 thru 4 of myData & "円") as string
else if length of myData = 5 then
set myData to (characters 1 thru 2 of myData & "," & ~
characters 3 thru 5 of myData & "円") as string
else if length of myData = 6 then
set myData to (characters 1 thru 3 of myData & "," & ~
characters 4 thru 6 of myData & "円") as string
end if
end repeat
end tell
end tell

```

```

set contents of cell N to myData
end repeat
end tell
end tell
end tell
set AppleScript's text item delimiters to OriginalDelimiters

```

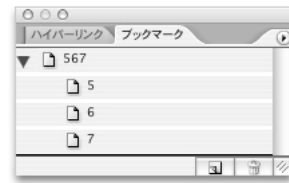
```

on replaceAll(motoStr, findStr, repStr)
set OriginalDelimiters to AppleScript's text item delimiters
set AppleScript's text item delimiters to {findStr}
set motoStr to text items of motoStr
set AppleScript's text item delimiters to {repStr}
set motoStr to motoStr as string
set AppleScript's text item delimiters to OriginalDelimiters
return motoStr
end replaceAll

```

PDFのブックマーク作成

ウィンドウ=>インタラクティブのブックマークパレットにブックマークを作成します。スクリプトを実行すると下図のような状態になり、PDFを作成するとこのブックマークは生きてきます。



プログラムとしてはhyperlinkオブジェクトをまず作成し、そこにリンク先のページやそのページの開き方の設定をしておきます。出来上がったhyperlinkオブジェクトの情報は見えないのですが、2回目に同じ名前のものを作ろうとするとエラーが出てしまいます。

次にブックマークを作成。そのプロパティに先ほど作成したhyperlinkオブジェクトを入れておくと出来上がりです。

作ったブックマークの中にもう一つブックマークを作ると上図のように入れ子になったブックマークが出来上がります。

```

tell application "Adobe InDesign CS2_J"
tell document 1
--★ここから下でページへのリンクの親メニューを作成できる
set theDest to make hyperlink page destination with properties ~
{destination page:page 5, view setting:fit window}
set oyaBookMark to make bookmark with properties ~
{destination:theDest, name:"567"}
--★ここから下でページへのリンクを作成できる
set theDest to make hyperlink page destination with properties ~
{destination page:page 5, view setting:fit window}
make bookmark at end of oyaBookMark with properties ~
{destination:theDest, name:"5"}
--★ここから下でページへのリンクを作成できる
set theDest to make hyperlink page destination with properties ~
{destination page:page 6, view setting:fit window}
make bookmark at end of oyaBookMark with properties ~
{destination:theDest, name:"6"}
--★ここから下でページへのリンクを作成できる
set theDest to make hyperlink page destination with properties ~
{destination page:page 7, view setting:fit window}
make bookmark at end of oyaBookMark with properties ~
{destination:theDest, name:"7"}
end tell
end tell

```

PDFハイパーリンク作成

こちらはハイパーリンク作成のサンプル。けっこう難しいです。最初にブックマークと同じくハイパーリンクオブジェクトを作成し、次にリンクテキストのリンク部分の領域を設定します。最後にリンクオブジェクトを作成し、その時、先ほど作成した2つの設定と下線の設定などを行って完成です。こんなところを読んでるあなたはかなりのマニア。

```

tell application "Adobe InDesign CS2_J"
tell document 1
get properties of hyperlink 1
--★ここから下でテキストをクリックしてページへのリンクを作成できる
set theDest to make hyperlink page destination with properties ~
{destination page:page 3, view setting:fit window}
set myhyperlink to make hyperlink text source with properties ~

```

```
{source text:text from character 1 to character -2 of paragraph 2 of ↵
text frame 1 of page 1}
make hyperlink with properties {border color:black, highlightinvert, ↵
border style:solid, hidden:false, destination:theDest, ↵
source:myhyperlink, name:"456789", visible:false, width:thin}
end tell
end tell
```

第7章 Acrobatと Photoshopほか

AcrobatもPhotoshopもスクリプト対応です。
AcrobatはAppleScript対応なのですがページ内の文字を調べる
事ができませんし、ブックマークを作ろうとすれば日本語が文字
化けします。
Photoshopはピクセルの色が取り出せないので画像によって処
理を変える自動補正のようなしくみは難しいかもしれません。

PDFにしおりをつける

各ページに飛ぶようにページ数のしおりをつくります。残念ながら日本語
のしおりはできません。

```
tell application "Adobe Acrobat 7.0 Professional"
tell document 1
set Cnum to count page
delete every bookmark
repeat with N from 1 to Cnum
set N2 to Cnum - N
set Bname to "P-" & N2 + 1
make bookmarks with properties ↵
(destination page number:N2 + 1, name:Bname)
end repeat
end tell
end tell
```

PDFをバラす

1つのPDFをページごとにバラバラにする。ほとんど使う事は無いが、そう
いう仕事があれば重宝。ファイルを開いて1ページ以外を全部削除して保存、
閉じる。またファイルを開いて2ページ以外全部削除して保存。。。われなが
らベタベタなスクリプト。duplicateは使えなかったのかどうかは不明。

```
set saveFolder to choose folder with prompt "PDF保存先フォルダを選択"
set saveFolder to saveFolder as string
tell application "Adobe Acrobat 7.0 Professional"
try
set TargetFile to file alias of document 1
set pageCount to count pages of document 1
on error
set TargetFile to choose file with prompt "PDFファイルを選択してください"
open TargetFile
delay 2
set pageCount to count pages of document 1
end try
close document 1
repeat with P1 from 1 to pageCount
open TargetFile
set pageCount2 to count pages of document 1
set FLG to false
repeat with P2 from 1 to pageCount2
if P1 is P2 then
set FLG to true
else
if FLG is false then
delete page 1 of document 1
else
delete page 2 of document 1
```

```
end if
end if
end repeat
set FName to saveFolder & P1 & ".pdf" as string
save document 1 to file FName
close document 1
end repeat
end tell
```

しおりのプロパティ変更

しおりをクリックした時のページの開き方を変更する。

```
tell application "Adobe Acrobat 7.0 Professional"
tell document 1
set bookmarkCount to count bookmark
repeat with N from 1 to bookmarkCount
set fit type of bookmark N to fit width
end repeat
end tell
end tell
```

簡易デジタル検版

下記スクリプトをアプリケーションで保存。ドロップレットになります。初
校と再校のJPEGを用意して、ドロップレットにドロップするとPhotoshop
で開き重ねて差分のみ表示します。

```
on open theList
if the number of theList is not 2 then
display dialog "画像は2つドロップしてください。"
else
tell application "Adobe Photoshop CS2"
open item 1 of theList
open item 2 of theList
set x2 to width of document 1 as pixels
set y2 to height of document 1 as pixels
select current document region {{0, 0}, {x2, 0}, {x2, y2}, {0, y2}}
activate
copy selection of current document
close document 2 saving no
paste
tell document 1
set blend mode of layer 1 to difference
merge visible layers
adjust layers using inversion
paste
make new art layer with properties {opacity:80}
select region {{0, 0}, {x2, 0}, {x2, y2}, {0, y2}}
fill selection with contents ↵
(class:RGB color, red:255, green:255, blue:255)
set properties of layer 3 to ↵
(background layer:false, blend mode:multiply)
move layer 3 to beginning
end tell
end tell
end if
end open
```

Excelサンプル。セル内の改行を
にする

Excelの表をtab区切りやcsvで書き出すときにセルない改行が入っていると
処理が面倒です。スクリプトであらかじめ置換します。

```
tell application "Microsoft Excel"
tell sheet 1
repeat with X from 1 to 100
set myStr to string value of cell X of row 1
if "update_date" is myStr then
set limitX to X
exit repeat
end if
end repeat
repeat with Y from 1 to 1000
set myStr to string value of cell 1 of row Y
if "" is myStr then
exit repeat
end if
repeat with X from 1 to limitX
```

```

set myStr to string value of cell X of row Y
if return is in myStr or tab is in myStr then
    set myStr to my replaceAll(myStr, ASCII character (10), "")
    set myStr to my replaceAll(myStr, return, "<br>")
    set myStr to my replaceAll(myStr, tab, "<tab>")
    set the formula of cell X of row Y to myStr
end if
end repeat
end repeat
end tell
end tell

```

```

on replaceAll(motoStr, FindStr, repStr)
    set OriginalDelimiters to AppleScript's text item delimiters
    set AppleScript's text item delimiters to {FindStr}
    set motoStr to text items of motoStr
    set AppleScript's text item delimiters to {repStr}
    set motoStr to motoStr as string
    set AppleScript's text item delimiters to OriginalDelimiters
    return motoStr
end replaceAll

```

Finder拡張子表示

```

set myFol to choose folder with prompt "フォルダを選択"
repeat with myFile in list folder myFol without invisibles
    set targetFile to (myFol as string) & (myFile) as alias
    tell application "Finder"
        set extension hidden of targetFile to false
    end tell
end repeat

```

Finderアクセス権変更

```

on run
    set curlItem to choose folder with prompt "フォルダを選択"
    my chmod777(curlItem)
end run

on open theList
    repeat with curlItem in theList
        my chmod777(curlItem)
    end repeat
end open

on FolderLoop(curlItem) --フォルダを送る
    set this_info to info for curlItem
    set folderFlg to folder of this_info
    if folderFlg then --フォルダだったら
        set myFol to curlItem as string
        repeat with myFile in list folder myFol without invisibles
            set curlItem to (myFol & myFile) as alias
            my FolderLoop(curlItem)
        end repeat
    else
        my chmod777(curlItem)
    end if
end FolderLoop

on chmod777(curlItem)
    set myPath to POSIX path of curlItem
    set cmdStr to "chmod -R 777 " & myPath
    do shell script cmdStr
end chmod777

```

mail新規メールに定型文挿入

```

tell application "Mail"
    tell outgoing message 1
        set paragraph 1 to "株式会社 ○○御中" & return & "○○○さま" & return
        --tell paragraph 1
        --get properties
        --end tell
    end tell
end tell

```

classic環境のQuarkXPress

classic環境のQuarkをOSXのAppleScriptで動かすのは思ったほど難しくなく変更なしでも動きます。ただsortは失敗しました。あとファイルのあつかいはうまく動かない事があるので注意が必要です。classic環境のスク립ト編集プログラムで動かす方法もあります。最新のマシンだとかなり高速に動きます。

第8章 自動組版の注意点

自動組版システムを作成するにあたって思わぬトラブルや設計上の注意などいままでの経験で気がついた事を記録しておきます。

データベース設計

データベース設計時の注意点は毎週発行のチラシの場合は最新のDBが常によりとは限らない事です。先週のチラシは400gの商品で、今週から300gになる商品がでてくるかもしれないですし、9月の1週からリニューアルしてパッケージと商品名が変わるのですが8月末までは前のままの商品など。先週分のDBと今週分のDBは切り離して残しておく仕組みが必要です。

また、しっかり設計してがんじがらめになるよりもいくらかでも項目を増やせるようにしておく事が重要です。これは例えば求人情報誌で給料の項目が1つしかないと、大卒、短大、高卒、アルバイトなどなど複数の給料が入る事があったときに困るからです。不動産のDBだって、最寄り駅の項目を予備込みで2つ作っておくと、必ず3つの最寄り駅がある物件がでできます。最寄り駅まで徒歩20分だけ最寄りバス停までなら2分とかとにかくいろんな例外が出てきます。最初から確認していてもあとから必ず設計変更になるので、とにかくゆる〜く作っておくのがコツです。

また入力ミスや間違っで消すような事も必ずあるので更新履歴からいつでももとに戻せる仕組みも必要です。プルダウンを多用するとDB的には間違いない確実なデータが手に入りますが、入力がわずらわしくなるのであるべく単純に入力してあげられるようにすることも大切です。

理論上できるのと動かしきるのは大違い

自動組版となると必ず大量です。1ページだけ作れてもこのまま400ページとはいきません。1ページ流す場合と400ページ流す場合は別のものになります。大量処理中にさまざまなエラーがおこるからです。

IllustratorでのSTOP現象はIllustratorの章で説明しましたが、Illustratorが何らかの理由で落ちて再起動中にもAppleScriptはどんどんコマンドを送り、なにも文字を流せていないのに保存のコマンドをIllustratorに送ったりする事があります。これを防ぐには流し終わったIllustratorファイルに本当に流し終わっているか調べるコマンドをおくります。

またInDesignにaiファイルを100ほど配置すると動かなくなります。epsなら動きますがCS2にEPSを100枚配置すると1枚ほど抜ける事があります。この場合も配置した後で配置できたか確認するスク립トも書く必要があります。

ヒューマンエラー

人間は機械ではないのでヒューマンエラーは必ずおこります。その際に動かなくなって自動組版自体できなくなるとたいへんなので、まず人にやさしいプログラム（全角半角を間違えても自動で動いてくれるような。。。）を心がけます。またストップした場合もなるべく原因を特定できるようなヒントをプログラムからエラーメッセージとして出してあげたいものです。（現実にはさまざまなエラーを想定するのがかなり難しいです。）

事故

自動組版後にイレギュラーで手を加えた後でもう一度自動組版すると先祖帰りしてしまいます。この先祖帰りはたちが悪くてだれも見つける事はでき

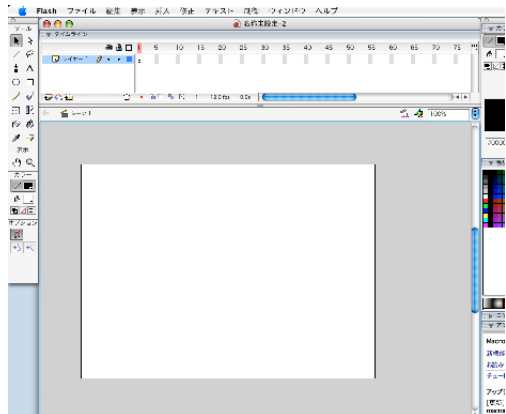
ません。というのも校正する際訂正箇所はかならずきっちり見るのですが、訂正箇所以外には目が見えないのです。これはだれもが全員そうです。まさか初校でOKだった箇所がおかしくなっているなど想像が付きません。とくにリンク画像があぶないです。マシながかわると同名で別のファイルにリンクし直したりします。inDesignなら警告がでますがIllustratorは警告が出ません。これを防ぐにはAcrobatの文書比較を利用した検版システムを使い、さらに一度手を加えた場所は上から自動でかぶせない事が重要です。

付録

デザイナーのための FLASH講座初級

ページが中途半端にあまったので、スクリプトのコメントをどんどん入れたかったのですが力つきました。。 すいません。そのかわりFLASHです。

FLASHの画面を見てください。はじめて触るソフトなのでおそらく見た事の無いパレットの多さに圧倒されそのままどうしてよいのかわからずWindowを閉じておしまい。というみなさんも多いのではないのでしょうか？しかしFLASHは考え方さえ理解できれば非常に簡単で良く出来たソフトなのです。最初の一步だけお手伝いできれば一週間もあればすぐにマスターできるでしょう。



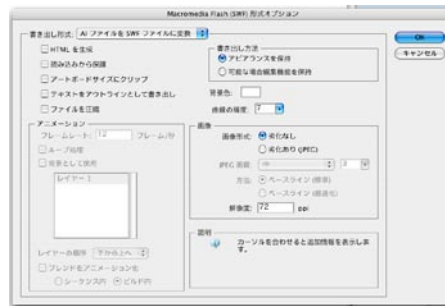
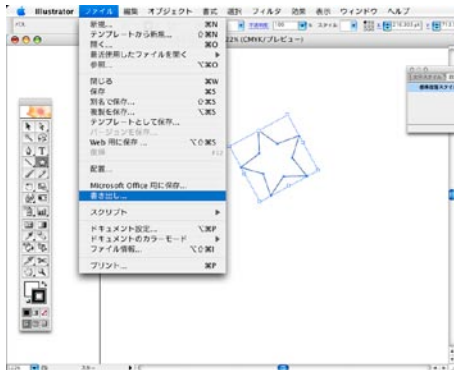
2つのマスターすべき事 (オブジェクトと時間)

まずはオブジェクトです。こちらは考えかたをグラフィックソフトに置き換えて考えると意外と簡単です。

FLASHがIllustratorやInDesignと違うのは時間の概念です。FLASHはアニメーションを作り出す事ができるソフトなので時間という考え方が重要です。こちらはちと難しいです。

オブジェクトの基本

まずIllustratorで適当な★を書きます。そして書き出しを選びSWF形式で書き出します。



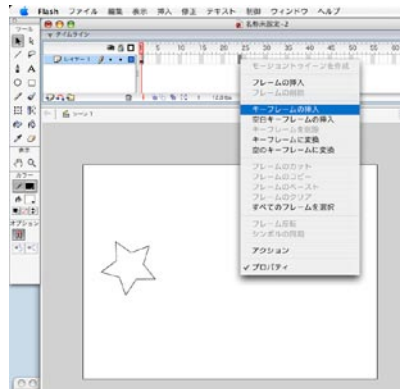
書き出しのオプションはいくつかあるのですが図のように設定します。

続いてFLASHでファイルメニューからライブラリに読み込みを選び先ほど作成したファイルを選びます。InDesign的に考えれば配置です。ただしちょっと違うのは1時的にライブラリに読み込む事です。

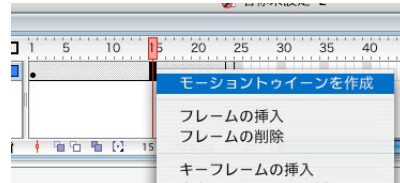
ウィンドウメニューからライブラリを選びライブラリパレットを表示させます。そこに先ほど読み込んだ"test.swf"が表示されていると思います。それをFLASHのメインウィンドウに配置します。

時間の基本

次に時間です。上部のタイムラインパレットを見てください。細かく区切られていますがこれがフレーム(コマ)です。12.0fpsと小さく書いてあると思いますが、これは1秒12フレームという事です。では2秒目の24フレームの部分をcontrolクリック(右クリック)してキーフレームの挿入を選んでください。



フレームができました。現在表示されているのは24フレーム目、つまり2秒後状態です。なにも動きが無ければ面白くないので★の位置を動かしてみてください。これでアニメーションを確認してみてください。タイムラインパレット上部の赤い目印はドラッグできます。ドラッグしてプレビューを確認してみます。23フレーム目と24フレーム目で★の位置が変化していると思います。さらに動きをスムーズにすればFLASHアニメの出来上がりです。1フレーム目から23フレーム目までの間をcontrolクリックし「モーショントウイーンを作成」を選びます。



1フレームと24フレームの間に→が出来たと思います。さっきのように赤い目印をドラッグしてみてください。こんどはスムーズに動きます。これで立派なFLASHアニメの完成です。ファイルを保存してファイルメニューのパブリッシュプレビューからFLASHを選んでみましょう。★がゆっくり繰り返しく思います。

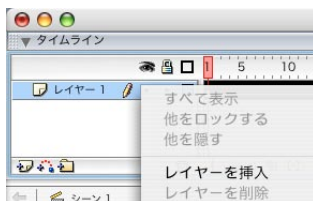
24フレーム目に作ったのはキーフレームでした。最初の1コマ目も実はキーフレームです。タイムラインパレットでキーフレームをクリックするとオブジェクトを動かす事が出来ますが、他のフレームでは動かせません。キーフレームはオブジェクトを置いたりすることが出来るフレームです。キーフレームとキーフレームの間はモーショントウイーンを作成すればスムーズに動きます。これもポイントです。

時間：練習(拡大と回転)

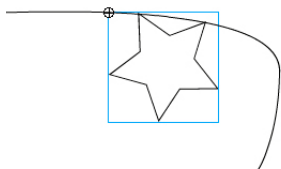
次は★を大きくしてみましょう。36フレーム目にキーフレームを作成しそこに出来ている★を大きくします。回転させてもかまいません。修正メニューの変形から伸縮と回転..を選択してください。さらに先ほどと同様に24~35フレームの部分をcontrolクリックしモーショントウイーンを作成します。これで大きくなる効果は完成です。

時間：練習（パスにそって移動）

オブジェクトを直線的な移動ではなくパスにそって移動させたいときは現在作業しているレイヤーをcontrolクリックして「モーションガイドを追加」を選びます。



適当にパスを描いてキーフレームのオブジェクトの原点をパスの上に乗るように移動させます。



これでプレビューするとパスにそったアニメーションが完成します。

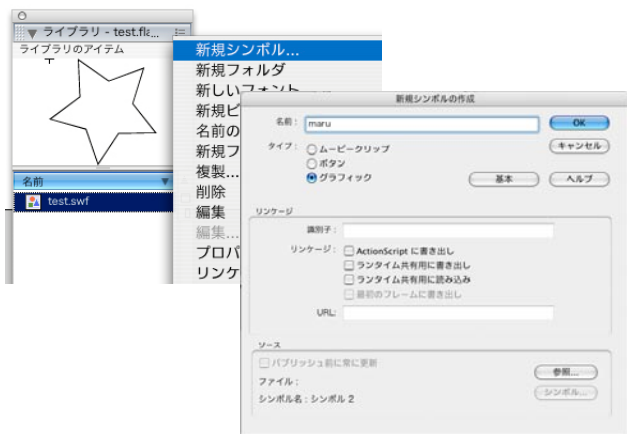
描かれたパスでは本番では表示されません。またガイドレイヤーの下にIllustratorのサブレイヤーのようにレイヤーを入ると2つ以上のアニメーションをパスに沿わせる事が出来ます。

※FLASHのパスですがIllustratorと違って黒矢印ツールではビットマップとして選択され白矢印ツールでパスとして扱う事が出来ます。要注意です。

オブジェクト：練習（FLASHに直接描く）

★のアニメーションができました。ただバックがさみしいので、ちょっとした効果を入れてみましょう。小さな円が大きくなるという効果で簡単なのですが結構効果的です。

ライブラリパレットのメニューから新規シンボルを選びます。シンボル名は"maru"にしましょう。タイプはグラフィックです。



シンボルができあがると画面の★が消えメイン画面真っ白になっています。タイムラインも作成したフレームが消えています。よくみると下図のようにシーン1 maruとなっています。これは"maru"の編集モードになっているという事です。シーン1をクリックすると先ほどの★の画面に戻ります。再び"maru"を編集するにはライブラリパレットの"maru"をダブルクリックします。



"maru"編集モードで真ん中に小さめの○を描いてください。塗りには透明線は黒にします。とはいつてもFLASHでは塗りを透明には出来ません。黒矢印ツールで選択すると線の部分と塗りの部分が別々に選択できます。これもIllustratorとの大きな違いです。黒矢印ツールで○の真ん中の塗り部分を選択してdeleteします。○は中央部に来るように移動してください。

オブジェクト：練習（小さなMOVIEを作る）

ライブラリパレットのメニューから新規シンボルを選びます。シンボル名は"marumovie"にしましょう。タイプはムービーです。この"marumovie"に先ほど作成した"maru"を配置します。（ドラッグドロップで配置できます。）

12フレーム目にキーフレームを作成しこの○を大きくします。おおきな○は線が太くてみっともないので下のパレット（プロパティパレット）の

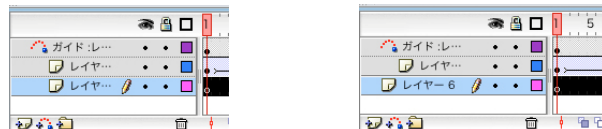
ルファ（透明度、ほかにカラーや明度等いろいろ設定できる。）を10%にします。（ちなみにインスタンスとパレットにあるのは"maru"の分身を置いていたと言った意味です。）



モーショントウインを作成します。○が大きくなりながら透明になるアニメーションが出来たと思います。

オブジェクト：練習（シーンに配置）

さてこの小さなmovieをメインのmovieに配置します。movieにmovieが配置できるのがFLASHの大きな特徴です。画面上部のシーン1をクリックしてシーン1へ行きます。さきほどの★のアニメになっていると思います。最下層にレイヤーを作ります。先ほどのガイドのサブレイヤーにならないように注意してください。そこに今作った"marumovie"を置きます。下図の左が失敗、右が成功です。



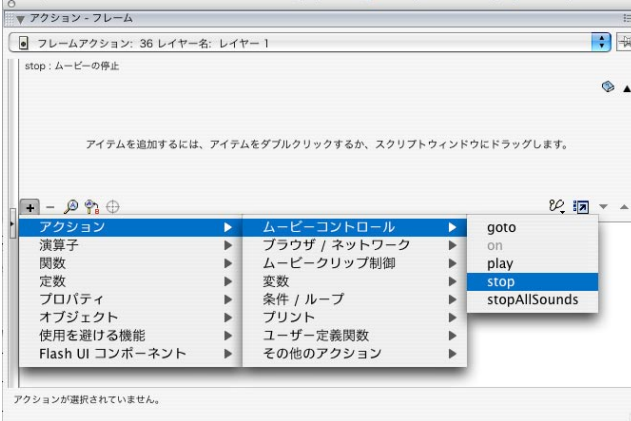
marumovieを1つだけでなくあちこちに配置します。大きさを変えても良いでしょう。（marumovieを配置したレイヤーのタイムラインが36フレーム分あればOKなのですが、もし1フレームしかないとき3秒に1回しか円が表示されません。その場合レイヤーの36フレーム目をcontrolクリックしてフレームを挿入を選択します。）

オブジェクト：練習（タイミングをずらす）

出来上がった○のムービーは全て同じタイミングで動くためタイミングをずらしたくなります。タイミングをずらすのはさほど難しくなく、新しいレイヤーを作りずらした部分にキーフレームを作成し、そこにムービーを配置するだけです。

時間：練習（ムービーが終わったらストップする）

ムービーのストップにはActionScriptを使います。★のレイヤーの36フレーム目のキーフレームをcontrolクリックしアクションを選択します。アクションパレットの+からアクション=>ムービーコントロール=>stopを選びます。アクションパレットにstop()が挿入されました。



★のアニメーションが動いた後ストップしますが、○は動きつづけるムービーをバックに配置したので動きつづけます。

作成する（パブリッシュ）

いちおうこれで完成としましょう。ファイルメニューからパブリッシュを選びます。htmlとswfファイルができます。パブリッシュ設定画面でいろいろ設定できますがとりあえず気にするのは互換性で前バージョンのswfとかならまあ安全でしょう。このあたりは時代の変化でかわりますので、その時々で判断する事になります。

ほかにも

あとはボタンの作成ができればひととおりのFLASHはできるようになります。FLASHのボタンは驚くほど簡単にきれいなものが作れます。こちらは割愛しますので入門書等を見てください。

すいません。時間切れです。。。

あとがき

このレジュメはAppleScriptのセミナーのためにわたしのホームページのTakeNoteというコーナーから情報を集め、加筆したものです。

いいわけがましいのですが、書くほどに自分の知識不足を思い知らされ、そして書かない方がよいのではと常に思います。ただAppleScriptを勉強する資料が非常に少なく、よくどうやって勉強するのか聞かれる事。それとAppleScriptに少しでも恩返ししたい事。そんなこんなでなんとかががんばって書いています。間違いや不適切な表現。もっと良い記述方法など多々あると思いますが、わたし程度の知識でもそれなりのものが作れると、逆に励みにしていただければと思います。

その他

スクリプトの内容はちゃんと整理整頓したかったのですが、かたっぱしからサンプルをいれた無秩序な仕上がりになってしまいました。しかも中途半端に2ページほどページが余りこれまた中途半端にFLASHなどをいれてしまいました。森田さんにセミナーのお誘いを受けた時に、来ていただける方とにかく満足してもらってお徳感を出したかったので、とにかく内容を盛りだくさんにしようと小さな文字でいっぱい詰め込みました。9/18からおおよそ3週間でここまで書き上げたのは自分でもなかなか満足です。(^^)

謝辞

今回、山口県印刷工業組合の内野卓理事のご厚意により、セミナーを開催することができました。ほんとうに厚く御礼もうしあげます。

またDTPエキスパート西日本会長・有馬哲也さん、Seed代表河原久美子さん、デザイナー北川訓子さんにも多大なるバックアップのおかげでこのようなセミナーが実現できました。感謝しています。

山口・北九州地区でセミナーの案内等に多くのご協力をいただいた瞬報社・中原さんに感謝いたします。

セミナー参加を快く応援してくれたわたしの勤める株式会社遊文舎と制作チームのみなさん。そしてセミナーというたいへん有意義な機会を与えていただいた森田デザインオフィス森田博巳さん。ほんとうにありがとうございます。

さらにこのレジュメをいま読んでいただいているみなさんとももちろん家族にも大感謝です。

たけうちとおる

<http://www.ne.jp/asahi/tan/puku/>

e-mail:abc68000@yahoo.co.jp

奥付

AppleScriptセミナーレジュメ

初版発行 2006年10月19日

著 者 たけうちとおる

発 行 人 山口県印刷工業組合

企 画 森田デザインオフィス

協 力 DTPエキスパートクラブ西日本

印 刷 トライス

本書の一部あるいは全部（プログラムを含む）を著作権法に定める範囲を超え、無断で複製・複写・転載・テープ化・ファイル化することを禁じます。

編著者および山口県印刷工業組合は本書の内容が正確であることに最善の努力を払いますが、内容の誤りあるいは欠落について保障はいたしません。また本書に含まれる情報あるいはプログラムの利用によって発生する損害に対して責任を負いません。あらかじめご了承ください。