

Word AppleScript

このドキュメントに記載されている情報 (URL などのインターネット Web サイトに関する情報を含む) は、
情報提供の目的で発行されているものであり、将来予告なしに変更することがあります。

別途マイクロソフトのライセンス契約上に明示の規定のない限り、このドキュメントはこれらの特許、
商標、著作権、またはその他の無体財産権に関する権利をお客様に許諾するものではありません。

Microsoft、Excel、Entourage、PowerPoint は、米国 Microsoft Corporation およびその関連会社の登録商
標または商標です。

Apple、Apple ロゴ、Mac、Mac OS は、米国 Apple Inc. の米国およびその他の国における登録商標または
商標です。

© 2008 Microsoft Corporation. All rights reserved.

Word AppleScript

Word AppleScript の基本	4
文書を開く	5
コレクションとクラスのリスト	11
検索と置換	16
削除する	20
ブックマーク	34

Word AppleScript の基本

Microsoft Word AppleScript トピックでは、Visual Basic for Applications (VBA) で一般的によく使用される サブルーチン (マクロ) と、これに相当する AppleScript サブルーチンについて説明します。

スクリプトをテストする際に **activate** コマンドを使用する

Macintosh スクリプト エディタ でスクリプトをテストするときは、各スクリプトの先頭に単独行として **activate** コマンドを指定することができます。これは `tell application "Microsoft Word"` ブロックの先頭行となります。これにより Word が前面に表示されるため、動作を確認しやすくなります。各トピックで紹介するサンプル スクリプトには **activate** コマンドは含まれていませんが、必要に応じて使用することもできます。

Word が前面に表示されているときに、保存したスクリプトを [スクリプト] メニューから実行する場合は、**activate** を指定する必要はありません。他の場所からスクリプトを実行していて、Word を前面に表示する必要がない場合も同様です。

参考資料

Word でスクリプトの作成を開始する前に、[Word 2004 AppleScript リファレンス](http://download.microsoft.com/download/A/0/7/A07D8BDE-8903-4A78-BD84-E2CF6A9DCF33/Word2004AppleScriptRef_J.pdf) (http://download.microsoft.com/download/A/0/7/A07D8BDE-8903-4A78-BD84-E2CF6A9DCF33/Word2004AppleScriptRef_J.pdf) の「Word Dictionary の使用」(特に「text range オブジェクトを操作する」) をお読みになることをお勧めします。

このドキュメントは、AppleScript よりも Word オブジェクト モデルに慣れていないユーザーを対象にしていますが、すべてのスクリプト作成者に役立つ情報が記載されています。

同じ名前のクラスとプロパティ

アプリケーション開発者は、クラスとプロパティに同じ名前を使用しないように努めています。

VBA の **Range** オブジェクトに対応する AppleScript のクラスは **text range** で、これは Text Suite にあります。VBA の多くのクラスで使用される **Range** プロパティに対応する AppleScript のプロパティは、ほぼ決まって **text object** であり、これは **text range** クラスのオブジェクトを返します。

同様に、AppleScript では、**font** クラスは他のクラスの **font object** プロパティの "型" になります。

テキスト範囲の変更

スクリプトで定義するテキスト範囲は、動的には扱われません。テキスト範囲を操作することはできませんが、テキスト範囲の開始位置または終了位置、あるいはその内容を変更すると、そのテキスト範囲を参照する変数は、その範囲を返さなくなります。変数を再設定して結果が返されるようにするか、別の方法で参照先を維持する必要があります。

そのため、VBA マクロを AppleScript に変換するときに変更を要することもあります。ほとんどの場合対処方法があります。範囲の変更については、「範囲を使用する」および「ブックマーク」を参照してください。

文書を開く

ドキュメントを作成する

新しい空白のドキュメントを作成するコードは、AppleScript と Visual Basic for Applications (VBA) とでは異なります。以下の例を比較してください。

VBA

```
Application.Documents.Add
```

AppleScript

```
make new document
```

AppleScript では、通常、**with properties** パラメータを使用して、ドキュメント作成時にプロパティの大半を指定します。要素の追加が必要な場合は後から行います。

Word で新しい文書を作成する

Microsoft Word で AppleScript を使用する場合、上記の方法とは異なります。

文書の作成時にプロパティを指定することはできないため、文書を作成してから変更する必要があります。手順は VBA と同じで、**Add** メソッドを使用して文書を作成した後で変更します。これは VBA のスクリーン作成者が使い慣れている手順です。

VBA で `Documents.Add` に指定できる引数は、**Template**、**NewTemplate**、**DocumentType**、および **Visible** のみです。**Document** の 50 以上のプロパティは、後から指定する必要があります。これは AppleScript でも同様です。

たとえば、次の VBA コードを使用して、フォントといくつかのコンテンツを設定できます。

```
Set NewDoc = Documents.Add
With NewDoc.Content
    .Font.Name = "Arial"
    .Text = "Here is some Text."
End With
```

Word AppleScript

同じことを AppleScript で実行する場合、次のコードが考えられます。

```
tell application "Microsoft Word"
    set newDoc to make new document with properties ↵
        {text object: {font object:{name:"Arial"},↵
        content:↵ "Here is some text."}}
end tell
```

メモ **text object** は Dictionary に読み取り専用と記載されていますが、このようなプロパティは、通常、作成時に設定できます。

しかし、上のコードを使用しても、取得されるのは空白の新しい文書です。VBA コードと同じ結果を得るには、次の例に示すように、まず **document** を作成してから、文書の **text object** プロパティを設定する必要があります。

```
tell application "Microsoft Word"
    set newDoc to make new document
    tell newDoc's text object
        set name of font object to "Arial"
        set content to "Here is some text."
    end tell
end tell
```

VBA の Documents.Add ステートメントで使用できる上記の 4 つの引数が、AppleScript で使用できるかどうかを検討してみましょう。

.Visible = False をマクロで使用することはめったにありません。また、マクロで使用することはあっても、AppleScript でこの引数を使用して非表示の文書を作成することはできません。ただし、文書の window 1 の **collapsed** プロパティを *true* に設定して、ドキュメントを作成してから最小化することができます。

VBA では、引数 **NewTemplate** を *True* に設定すると、事前に新しいドキュメントをテンプレートに設定できますが、AppleScript ではできません。AppleScript で同じことを行うには、新しいドキュメントを後からテンプレートとして保存します。**save as** コマンドを使用する際に、**file format** パラメータを *template* に設定します。

NewTemplate:=True を使用している VBA マクロを AppleScript に変換する場合は、このステートメントを無視して、保存時に `save as file format format template` を使用します。

また、Macintosh で使用できる **DocumentType** の値は、既定値 *wdNewBlankDocument* を除くと *wdNewWebPage* だけです。DocumentType:= wdNewWebPage を使用している VBA マクロを AppleScript に変換する場合は、このステートメントを無視して、保存時に `file format format HTML` を使用します。

次の例に示すように、文書の作成後に **view** も *online view* に設定することができます。

```
set view type of view of window 1 of newDoc to online view
```

テンプレートから文書を作成する

attached template プロパティを使用する

AppleScript でも、Visual Basic for Applications (VBA) と同様に、**Add** メソッドの引数 **Template** を使用してテンプレートから文書を作成できます。**make new document** と **with properties** パラメータを使用した場合はうまく動作しませんが、次の例のように、新しい文書の読み取り専用ではない **attached template** プロパティを、任意のテンプレートに設定することができます。

```
tell application "Microsoft Word"
    set newDoc to make new document
    set attached template of newDoc to ~
    "Mac HD:Applications:" & ~
    "Microsoft Office 2004:My Templates:Test Template.dot"
    get name of attached template of newDoc
end tell
```

メモ Dictionary でテンプレートの "名前" を指定するように示されている場合、それは完全なパスを意味しています。この例ではパスを短く分割していますが、コロン区切りでテンプレートのパス全体を入力できます。また、Macintosh スクリプト エディタ に POSIX file "" as alias as Unicode text を入力し、この引用符の間にファイルをドラッグすることもできます。

ただし、上記のコードを実行すると問題が発生します。スクリプトを実行しても、文書は変更されず、Test Template.doc のスタイルもマクロも追加されません。

attached template プロパティの問題を回避する

この問題を回避するには、attached template を設定した後に、**insert file** を使用してテンプレートを挿入します。これで、ヘッダーとフッター、スタイル、マクロを含むテキストを取得できます。ここで取得できないものはページ設定 (余白、行間など) のみです。ヘッダーの間隔を広げるなど、標準テンプレートと異なる設定を使用したテンプレートの場合は、テンプレートを暫定的に **document** として開くコードを追加して、そのページ設定を取得できるようにします。

コードは次のように記述します。

```
set templatePath to "Mac HD:Applications:" & ~
    "Microsoft Office 2004:My Templates:Test Template.dot"
tell application "Microsoft Word"
    set newDoc to make new document
    set attached template of newDoc to templatePath
    insert file at text object ~
        of newDoc file name templatePath
end tell
```

セキュリティ設定でマクロの警告がオンに設定されている場合、テンプレートにマクロが含まれていると、attached template を文書に設定するときにマクロの警告が表示されます。セキュリティ上の制限により、**application** の **automation security** プロパティを使用しても、一時的に警告をオフにすることはできません。

テンプレートのパスは 2 回使用する必要があるため、パスに対して変数 `templatePath` を設定します。

特別なページ設定を取得する

page setup 設定 (余白など) を含める必要がある場合は、以下の長いバージョンを使用します。

```
set templatePath to "Mac HD:Applications:" & -
    "Microsoft Office 2004:My Templates:Test Template.dot"
tell application "Microsoft Word"
    activate
    set newDoc to make new document
    set attached template of newDoc to templatePath

    set theTemplate to attached template of newDoc
    -- a template object now
    set editableTemplate to open as document theTemplate
    -- a document
    set pageSetupProps to properties -
        of (get page setup of editableTemplate)
    set lineNumberProps to properties -
        of (get line numbering of pageSetupProps)
    close editableTemplate saving no
    tell page setup of newDoc
        set its properties to pageSetupProps
        tell its line numbering
            set its properties to lineNumberProps
        end tell
    end tell

    insert file at text object -
        of newDoc file name templatePath
end tell
```

この長いバージョンでは、テンプレートが **open as document** コマンドによって **document** として開かれるため、**page setup** などの **document** プロパティを使用できます。

その次の行では、**page setup** の **properties** プロパティを使用していますが、これですべてのプロパティのレコードが返されます。この方法は効率が良く、AppleEvent を 30 回送信するところを 1 回送るだけで済みます。また、スクリプト内で 1 つずつ指定したり、それぞれに変数を設定したりする必要もありません。**properties** プロパティは、すべての Microsoft Office アプリケーションのほとんどすべてのクラスで使用できます。

page setup の **line numbering** プロパティは参照であるため、**line numbering** のすべてのプロパティも取得します。取得しない場合、別の文書の **page setup** プロパティをこの **properties** レコードに設定しようとしても、別の文書の項目へ設定する時点でうまくいきません。

必要な情報をすべて取得した後、テンプレートを閉じます。

Word AppleScript

次に続くコードは興味深いものです。newDoc の **page setup** プロパティは、プロパティ レコード全体を含む変数に設定できます。**line numbering** プロパティも同様です。このプロパティを設定すると、30 個の変数で 30 個のプロパティを指定するという面倒な作業から解放されます。30 個のプロパティに含まれるのは、**class** という読み取り専用プロパティ 1 つだけです。これは、line numbering プロパティでも同様です。

これで、テンプレート ファイルを挿入できます。

マクロのセキュリティ警告を表示しない

マクロのセキュリティ警告を表示したくない場合やオフにしたくない場合、またはテンプレートを一時的に開きたくない場合などは、次のコードを make new document with properties ステートメントの代わりに使用します。

```
do Visual Basic "Documents.Add Template:=\"Mac HD:\" & _  
  \"Applications:Microsoft Office 2004:My Templates:Test Template.dot\""
```

このステートメントによって、Documents.Add Template を使用するマクロが変換され、Microsoft Word 2004 for Mac で正しく機能します。

既存の文書を開く

AppleScript では既存の文書を簡単に開くことができます。コードは Visual Basic for Applications (VBA) と類似しています。以下の例を比較してください。

VBA

```
Documents.Open FileName:="Mac HD:Folder:Filename.doc"
```

AppleScript

```
tell application "Microsoft Word"  
  open alias "Mac HD:Folder:Filename.doc"  
end tell
```

VBA コードでは、**Open** メソッドを使用しますが、AppleScript では、(Standard Suite の) **open** コマンドを使用します。正式な AppleScript では、エイリアス参照形式を使用しますが、Microsoft Word の場合は、VBA と同じく、強制型変換によって、パスのテキストだけを使用できるようになっています。また、AppleScript では必要に応じて、リスト内の文書をすべて同時に開くことができます。

文書を開いて変更する

文書を変更したり、文書から情報を取得したりする必要がある場合があります。次の例は、このようなタスクを実行する VBA コードです。

```
Dim oDoc As Document
Set oDoc = Documents.Open FileName:="Mac HD:Folder:Filename"
With oDoc
    .HyphenationZone = InchesToPoints(0.25)
    .HyphenateCaps = False
    .AutoHyphenation = True
End With
```

VBA では、変数 (参照) を **Document** オブジェクトを返す `Open` ステートメントの結果に設定し、その後、**Document** のプロパティを設定するか、メソッドを適用します。

AppleScript では、コマンドの結果に変数を設定することはできませんが、Standard Suite の **open** コマンドは結果を返さないため、変数を設定できません。

open コマンドに関する問題を回避する

アプリケーション開発者は、言語の一貫性を保持するために Standard Suite のコマンドをできる限り使用して、Apple が提供するモデルに従っています。ただし、アプリケーション開発者が独自のパラメータを Standard Suite のコマンドに追加することは可能で、実際に Word 開発者は 10 個のパラメータを追加しています。これらはすべて、VBA の **Open** メソッドから派生しています。ただし、アプリケーション開発者にとって、**open** が結果を返さないという点では変わりません。

次のコードを使用すると、文書を変更する際に **open** コマンドの制限によって生じる問題を解決できます。

```
tell application "Microsoft Word"
    open " Mac HD:Folder:Filename.doc"
    set theDoc to active document
    tell theDoc
        set hyphenation zone ↵
            to (inches to points inches 0.25)
        set hyphenate caps to true
        set auto hyphenation to true
    end tell
end tell
```

このコードでは、**document** を開いて active document (前面にある文書) に変数を設定します。

開いている文書で open コマンドを使用する

文書が開かれていない場合、**open** コマンドはアプリケーションの前面にファイルを開きます。一方、文書が既に開かれている場合は前面に開かれず、目的としていたものとは別の文書进行操作することになります。

Word AppleScript

文書が開かれている場合は、**open** コマンドの制限に対応した次のコードを使用します。

```
tell application "Microsoft Word"
  try
    set theDoc to document "Filename.doc"
  on error
    open "Mac HD:Folder:Filename.doc"
    set theDoc to active document
  end try
  tell theDoc
    set hyphenation zone ↵
      to (inches to points inches 0.25)
    set hyphenate caps to true
    set auto hyphenation to true
  end tell
end tell
```

コレクションとクラスのリスト

AppleScript にはコレクション オブジェクトがありません。その代わりに、各クラスの複数形 (リスト) を使用します。複数形が存在する場合は、Dictionary に必ず記載されています。ごくまれに複数形がないものがあり、その場合はリストを取得できません。

複数形がないのは、**application** や **selection** などの、オブジェクトを 1 つしか持たないクラスだけです。これ以外の場合、クラスは別のクラスの要素 (Dictionary エントリの「contained by」セクションに記載) またはアプリケーション自体の要素です。

every [クラス名] 構文

複数形と同義で一般的に使用されている **every [クラス名]** という表現があります。Visual Basic for Applications (VBA) でコレクションの各メンバに対して反復処理を行う For Each ループは、AppleScript では以下のステートメントのように repeat ループに置き換えます。

- repeat with someObject in (every element)
- repeat with i from 1 to (count every element)

これらの構文の動作を理解するために、フィールドのリンク解除を行う次の例を比較してください。

VBA

```
For Each oField In ActiveDocument.Fields
  oField.Unlink
Next oField
```

AppleScript

```
tell application "Microsoft Word"
    set allFields to (every field of active document)
    repeat with oField in allFields
        unlink oField
    end repeat
end tell
```

この VBA マクロでは、**Fields** コレクションを操作します。一方 AppleScript の例では、**every field** 構文を使用します。

リストで使用できないことが多いコマンド

AppleScript では、コマンドはリスト全体に一度に作用するので、次の例のように複数形を使うと repeat ループを使用しなくてもよいように見えます。

```
tell application "Microsoft Word"
    unlink (every field of active document)
end tell
```

しかし、コマンドはリストに対して常に機能するわけではありません。

上に示した例では、明示的な **get** または変数を使用してもエラーが返されます。このエラーは、`{field 1 of active document, field 2 of active document, ...}` がリンク解除メッセージを理解しないことを示しています。ドキュメントを確認すると、最初のフィールドは実際にリンク解除されていますが、残りは解除されていないことがわかります。これは、**unlink** コマンドがリストに対して機能するようには実装されていないという単純な理由によるものです。

Microsoft Word、Excel、PowerPoint では、ほとんどの場合、リストに対してコマンドを使用することはできません。ただし、Entourage は別です。コマンドでは大部分で、派生元となった VBA メソッドが再現されており、メソッドではリストは使用されません。したがって、通常はリストに対して repeat ループを使用する必要があります。

repeat ループを使用するときに変数に複数形を割り当てる

上に示した AppleScript の例では repeat ループを使用しています。この例の `allFields` のように、ループの前に変数を設定してそれを参照する必要があります。

変数なしで次のコードを使用すると、明示的な **get** を使用してもエラーが返されます。

```
repeat with oField in (get every field of active document)
    unlink oField
end repeat
```

このエラーは、**field** がリンク解除メッセージを理解していないことを示しています。これは、フィールドがリンク解除されるたびに再インデックスされ、インデックスが残りのフィールド数を超えるまで 1 つおきにスキップされることが原因です。変数を使用しない場合は、次のコードを使用する必要があります。

Word AppleScript

```
tell application "Microsoft Word"
    repeat with i from (count (get every field ↵
        of active document)) to 1 by -1
        set oField to item i of (get every field ↵
            of active document)
        unlink oField
    end repeat
end tell
```

上に示した例では、次の要素が必要です。

- 明示的な **get**
- repeat with i from ループ
- 最後の項目から最初の項目まで -1 ずつ戻る count

変数を使用しない場合、get every field of active document を 2 回使用して、毎回 AppleEvent を送る必要があります。AppleEvent は必然性が高いものではなく、また処理が遅いため、前の allFields を使用した例のように変数を使用するようにしてください。

また、次の例のように repeat with i ループを使用することもできます。

```
tell application "Microsoft Word"
    set allFields to (every field of active document)
    repeat with i from 1 to (count allFields)
        set oField to item i of allFields
        unlink oField
    end repeat
end tell
```

他のクラスの要素であるクラスを使用する

次の例は、ドキュメント内のすべてのヘッダーとフッターのリンクを解除する VBA マクロです。この例では、いくつかの点で VBA と AppleScript でのオブジェクトの違いを確認することができます。

```
Dim oField As Field
Dim oSection As Section
Dim oHeader As HeaderFooter
Dim oFooter As HeaderFooter

For Each oSection In ActiveDocument.Sections

    For Each oHeader In oSection.Headers
        If oHeader.Exists Then
            For Each oField In oHeader.Range.Fields
                oField.Unlink
            Next oField
        End If
    Next oHeader
```

Word AppleScript

```
For Each oFooter In oSection.Footeres
    If oFooter.Exists Then
        For Each oField In oFooter.Range.Fields
            oField.Unlink
        Next oField
    End If
Next oFooter

Next oSection
```

この例では、ドキュメントの本文ではなく、ヘッダーとフッターの **Ranges** 内でフィールドのリンクを解除しています。ところが、以下の理由で AppleScript ではまったく異なるコードになります。

- コレクション オブジェクトがない
- ヘッダーとフッターのリストを作成できない

header footer クラスは、他のどのクラスの要素でもなく、また **section** の要素でもないため、想定したとおりにこのリストを作成することはできません。このような問題のため、AppleScript でのタスクの実現が困難になっています。

header footer が他のクラスの要素でないのは、読み取り専用の要素を持つことができないという AppleScript のルールに起因すると考えられます (読み取り専用のプロパティは可能です)。オブジェクトは、0 から無限の数まで要素を持つことができ、`make new element at someObject` を使用するだけで、新しい要素を作成することができます。

その一方で、Word、Excel、PowerPoint のオブジェクト モデルには、**Add** メソッドを持たない多くのコレクション オブジェクトが含まれています。使用できるのは、与えられているものだけです。各ドキュメントの **Section** の **HeaderFooters** コレクション オブジェクトは、この種のオブジェクトです。使用できるのは、最大で3つのヘッダーとフッター (*primary*、*first page*、と *even pages*) です。独自のものを追加することはできません。

ヘッダーとフッターは無数に作成できないため、AppleScript では、**header footer** オブジェクトは **section** の要素にすることができません。これらは、オブジェクト モデルから分離した自立オブジェクトである必要があります。Dictionary には、**base object** から継承されたものであると記載されています。

これらのクラスに対して専用のコマンドを使用する

これらのオブジェクトを取得するには、`header footers of section 1 of active document` のようなステートメントではなく、専用コマンドである **get header** と **get footer** を使用します。

また、**index** パラメータを使用して目的のヘッダーまたはフッターを指定する必要があります。この構文は、次の例のように、VBA で **Section** の **Headers** プロパティと **Footers** プロパティにインデックスを使用する場合と似ています。

```
ActiveDocument.Sections(1).Headers(wdHeaderFooterPrimary)
```

Word AppleScript

AppleScript での同等のコードは次のようになります。

```
get header (section 1 of active document) index ↵
    header footer primary
```

ただし、すべてのヘッダーとフッターを取得する簡単な方法はありません。前の VBA の例の `ActiveDocument.Sections(1).Headers` に相当するコードは AppleScript にはありません。

そのため、すべてのヘッダーとフッターをインデックスで取得する必要があります。それには次の 2 つの方法があります。

- それぞれをハンドラを使用して実行し、テキスト オブジェクト (**range**) とフィールドを取得する。
- それぞれを単一項目のリストとしてリストの中かっこで囲み、これらのリストを 1 つのリストに連結し、このリストに対して repeat ループを実行する。

次のコードでは 2 番目の方法を実行しています。

```
tell application "Microsoft Word"
    set theSections to every section ↵
        of active document
    repeat with theSection in theSections
        set theHeaderFooters to ↵
            {get header theSection index ↵
                header footer primary} & ↵
            {get header theSection index ↵
                header footer first page} & ↵
            {get header theSection index ↵
                header footer even pages} & ↵
            {get footer theSection index ↵
                header footer primary} & ↵
            {get footer theSection index ↵
                header footer first page} & ↵
            {get footer theSection index ↵
                header footer even pages}
        repeat with theHeaderFooter in theHeaderFooters
            set theFields to every field ↵
                of text object of theHeaderFooter
            repeat with theField in theFields
                unlink theField
            end repeat
        end repeat
    end repeat
end tell
```

他のクラスの要素にはならず、専用コマンドでしか使用できないクラスが他にもあるので注意してください。このようなクラスは数多く存在します。

これらのクラスに対して存在 (exists) テストを行わない

AppleScript では、VBA で実行するような **header footers** に対する `exists` テストは不要です。VBA モデルでは、既定で存在するのはプライマリ ヘッダーとフッターのみです (これらは空のこともあります)。そのため、*first page* と *even pages* のヘッダーとフッター (定数 `wdHeaderFooterFirstPage` と `wdHeaderFooterEvenPages`) が存在するかどうかをチェックする必要があります。

AppleScript では、コマンド `get header` と `get footer` でパラメータ *header footer first page* と *even pages* を適用して結果を取得できるため、これらはすべて、空であっても、既定で存在します。text object of theHeaderFooter がなく、theFields の空のリストになることには何の問題もありません。

検索と置換

文字列の検索と置換を行うために、Microsoft Word のマクロを使用することは珍しくはありません。Visual Basic for Applications (VBA) と AppleScript では、検索と置換の実装が異なります。

グローバルな検索と置換

Word の VBA では、文書の特定の部分 (本文、ヘッダー、テキスト ボックスなど) でのみ検索と置換を行います。一方、AppleScript では、ヘッダー、フッター、テキスト ボックスを含め、文書全体でグローバルな検索と置換を実行できます。以下の例を比較してください。

VBA

```
Sub ReplaceEverywhere(FindText As String, ReplaceText as String)
    Dim oSection As Section
    Dim oShape As Shape
    Dim oHF As HeaderFooter

    ReplaceInRange FindText, ReplaceText, ActiveDocument.Content
    For Each oShape In ActiveDocument.Shapes
        If oShape.TextFrame.HasText Then
            ReplaceInRange FindText, ReplaceText,
                oShape.TextFrame.TextRange
        End If
    Next oShape
    For Each oSection In ActiveDocument.Sections
        For Each oHF In oSection.Headers
            ReplaceInRange FindText, ReplaceText,
                oHF.Range
            For Each oShape In oHF.Shapes
                If oShape.TextFrame.HasText
                    Then ReplaceInRange FindText,
                        ReplaceText, _
                        oShape.TextFrame.TextRange
                End If
            Next oShape
        Next oHF
    Next oSection
End Sub
```

Word AppleScript

```
Next oHF
For Each oHF In oSection.Footers
    ReplaceInRange FindText, ReplaceText,
        oHF.Range
    For Each oShape In oHF.Shapes
        If oShape.TextFrame.HasText Then
            ReplaceInRange FindText,
                ReplaceText,
                oShape.TextFrame.TextRange
        End If
    Next oShape
Next oHF
Next oSection
End Sub

Sub ReplaceInRange(FindText As String, ReplaceText as String, _
    oRange as Range)
    With oRange.Find
        .Format = False
        .Text = FindText
        .Replacement.Text = ReplaceText
        .Wrap = wdFindStop
        .Execute Replace:=wdReplaceAll
    End With
End Sub
```

AppleScript

```
on ReplaceEverywhere(findText, replaceText)
    local theShape, theRange, theHeaderFooters, theHeaderFooter
    tell application "Microsoft Word"
        --first find and replace in the body (main story) of document
        set theRange to text object of active document
        my ReplaceInRange(findText, replaceText, theRange)

        --nowin all shapes
        set allShapes to (every shape of active document)
        repeat with theShape in allShapes
            set theTextFrame to (text frame of theShape)
            if has text of (text frame of theShape) then
                --note:'text range', not 'text object'
                of text frame
                set theRange to text range of text frame
                of theShape
                my ReplaceInRange(findText, replaceText,
                    theRange)
            end if
        end repeat

        -- now in the headers and footers of each section
```

```
set allSections to every section of active document
repeat with theSection in allSections
    set theHeaderFooters to {get header theSection index -
        header footer primary} & -
        {get header theSection index -
        header footer first page} & -
        {get header theSection index -
        header footer even pages} & -
        {get footer theSection index -
        header footer primary} & -
        {get footer theSection index -
        header footer first page} & -
        {get footer theSection index -
        header footer even pages}

    repeat with theHeaderFooter in theHeaderFooters
        set theRange to text object -
            of theHeaderFooter
        my ReplaceInRange(findText, -
            replaceText, theRange)

        --now in their shapes
        set allShapes to (every shape -
            of theHeaderFooter)
        repeat with theShape in allShapes
            if has text of (text frame of theShape) -
                then
                    --note:'text range', not -
                    'text object' -
                    of text frame
                    set theRange to text range -
                        of text frame -
                        of theShape
                    my ReplaceInRange(findText, -
                        replaceText, theRange)
                end if
            end repeat
        end repeat
    end repeat
end tell
end ReplaceEverywhere

on ReplaceInRange(findText, replaceText, theRange)
    tell application "Microsoft Word"
        set findObject to find object of theRange
        tell findObject
            set format to false
            set content to findText
            set content of its replacement to replaceText
            set wrap to find stop
        end tell
    end tell
```

```
execute find findObject replace replace all
end tell
end ReplaceInRange
```

tell ブロックでコマンドとパラメータを配置する

この例では、各 `range` に対して操作を実行する `ReplaceInRange` ハンドラに、**findObject** (`text range` の **findObject** プロパティ) に対する `tell` ブロックがあることがわかります。

VBA では、**Execute** メソッドを、すべてのプロパティと同じように `with` ブロック内に配置することもできます。一方、AppleScript では、**execute find** コマンドは、同じダイレクトパラメータ (**findObject**) とプロパティを使用しているにもかかわらず、動作しないように見えます。このコマンドは、`tell` ブロックの外側に配置され、**findObject** がダイレクトパラメータとして明確に参照されている場合に動作します。

`tell` ブロックのターゲットとなるダイレクトパラメータを持つコマンドは、ターゲットオブジェクトに対して問題なく動作するはずですが、AppleScript ではうまくいかないことがあります。実際には、次の例のように、用語 `it` を使用してダイレクトオブジェクトを限定することで、**execute find** を `tell` ブロック内に配置できます。

```
tell findObject
    set format to false
    set content to findText
    set content of its replacement to replaceText
    set wrap to find stop
    execute find it replace replace
all
end tell
```

同じ名前のクラスとプロパティ

Drawing Suite の **text frame** クラスには、(**text object** プロパティではなく) **text range** プロパティがあり、クラスの **range** を表します。一般的に、クラスとプロパティには同じ名前を付けないように、開発者は十分な配慮をしていました。このため、クラス **find** の **findObject** プロパティ、型 **font** の **font object** プロパティ、型 **text range** の **text object** プロパティという命名規則が、さまざまなクラスで使用されています。しかし、この例では、そのような命名規則は適用されていません。

text frame の **text range** プロパティに `whose` フィルタを実行する場合や **text frame** をターゲットにした `tell` ブロックでそのプロパティを参照する場合は、キーワード `its` を含めてください。

同じ名前を共有するプロパティとクラスの例がもう 1 つあります。**findObject** に対する `tell` ブロックの `ReplaceInRange` ハンドラでは、`its replacement` を使用しています。これは、**replacement** がクラス名でもあるためです。`its` を指定しない場合、スクリプトにエラーが発生します。このキーワードは、**findObject** の **replacement** プロパティを使用することを指定するもので、指定しないと、アプリケーションの **replacement** 要素 (クラス) が優先されます。

頻繁に使用するハンドラをコピーしてペーストする

このハンドラのペアを、スクリプトの"ライブラリ"にあるかのように、いつでも使用できるようにします。つまり、最上位のコマンドがなく、ハンドラ (サブルーチン) と必要なスクリプト プロパティだけを含む状態で、部分的なスクリプトを保存しておきます。これによって、この2つのハンドラをスクリプトにコピーしてペーストできます。文字列の検索と置換が必要な場合は、ReplaceEverywhere ハンドラを呼び出します。次の例では、"hot" を "cold" に置き換えます。

```
my ReplaceEverywhere("hot", "cold")
```

慣例としてはこの"最上位"のコマンドをスクリプトの最上位に置き、ハンドラを最下位に置きますが、AppleScript ではその逆もできます。スクリプトではコンパイル時に、何がどこにあるかをすべて"学習"します。

ハンドラ呼び出して my を使用する

厳密にいうと、ReplaceEverywhere("hot", "cold") への呼び出しがそれ自体アプリケーションの tell ブロック内になければ、my は必要ありません。しかし、ほとんどの場合はこの呼び出しを別の Word の tell ブロック内で行うため、my が必要です。つまり、アプリケーションではなく、スクリプトが"親"なので、スクリプトでは用語 ReplaceEverywhere をアプリケーションのキーワードと見なしません。その結果、失敗してエラーが発生します。

したがって、サブルーチンを呼び出す場合は、常に my を使用することをお勧めします。

削除する

インデックスを使用してオブジェクトを参照する

AppleScript では、すべてのオブジェクト参照はインデックスによって行われ、項目を削除するたびに再評価されます。

逆方向に移動するループを使用して項目を削除する

誤って項目をスキップした場合、最後の項目のインデックスが既存の項目のインデックスよりも大きくなると、結果として、スクリプト エラーが返されます。これを解決するには、"by -1" で逆方向に移動する repeat ループを使用して、項目を逆方向に反復処理します。

次の例では、表の最初の列に特定のテキスト文字列を含むすべての行を削除しています。

VBA

```
Sub DeleteRows()  
  
Dim TargetText As String  
Dim oRow As Row  
  
If Selection.Information(wdWithInTable) = False Then Exit Sub
```

Word AppleScript

```
TargetText = InputBox$("Enter target text:", "Delete Rows")

For Each oRow In Selection.Tables(1).Rows
    If oRow.Cells(1).Range.Text = TargetText & vbCr
        & Chr(7) Then oRow.Delete
Next oRow

End Sub
```

AppleScript

```
set cellEndChar to ASCII character 7
tell application "Microsoft Word"
    if (get selection information type ~
        (with in table)) = "false" then return
    display dialog ~
        "Delete rows with this text in cell 1:" ~
        default answer "Enter target text" with icon 1
    set targetText to text returned of result

    set theRows to every row of table 1 of selection
    repeat with i from (count theRows) to 1 by -1
        set theRow to item i of theRows
        if content of text object of cell 1 of theRow = ~
            (targetText & return & cellEndChar) then
            delete theRow
        end if
    end repeat
end tell
```

特定の文字を呼び出すコマンドを使用しない

AppleScript の **ASCII character 7** は、Visual Basic for Applications (VBA) の **Chr(7)** と同じで、Microsoft Word において、改行記号の後で表のセルの最後を示す文字として使用される非表示の文字です。改行記号は、VBA では vbCr または **Chr(13)** で、AppleScript では return または **ASCII character 13** です。**ASCII character** は Standard Additions の Dictionary にあるコマンドです。この Dictionary には、スクリプティング機能追加の組み込みの Macintosh コレクションが含まれています。この文字を表示するには、次の手順に従ってください。

1. [Word] メニューの [環境設定] をクリックします。
2. [Word 環境設定] ダイアログ ボックスの [作成および校正ツール] で、[表示] をクリックします。
3. [表示] ダイアログ ボックスの [編集記号の表示] で、[段落] のチェック ボックスをオンにします。

Word AppleScript

スクリプティング機能追加を呼び出す場合、オーバーヘッドが生じてスクリプトの実行が遅くなるため、これらの文字を `repeat` ループで何度も呼び出さないようにする必要があります。これを解決するには、スクリプトの最初でスクリプティング機能追加に変数を設定し、1 回だけ呼び出しを実行します。

同様に、AppleScript の定数 `return`、`space`、`tab` を使用し、対応する **ASCII character** コマンド (**13**、**31**、**9**) を使用しないという要件もよくあります。

Word 2004 より前のバージョンでは、Word `tell` ブロックで `tab` を使用できません。これは、**tab** クラスとして解釈されるためです。このクラスは、現在 **tab stop** と呼ばれており、`tab` 自体は通常のテキスト文字になっています。

また、AppleScript では、`return` 文字と **return** コマンドが区別されます。上に示したコード (`targetText & return & cellEndChar`) で、この文字を確認できます。

繰り返しループでの項目の削除方法

AppleScript の `repeat` ループと VBA の `For Each` ループにも違いがあります。AppleScript では、構文 `repeat with theRow in the Rows`、または厳密には `repeat with theRow in (every row of table 1 of selection)` を使用する場合、リストのインデックス付けが `repeat with i from 1 to (count theRows)` ループに従って実行され、カウントが追跡されます。これにより、反復処理ごとに項目のリストが確認または更新されますが、インデックスは確認されません。

このため、項目の削除に `delete` ループを使用する場合はいつでも、`(count theRows) to 1 by -1` のように逆方向に反復処理する必要があります。このようにしないと、項目を削除するたびに次の反復処理で 1 つの項目がスキップされます。

たとえば、`[a, b, c]` の 3 つの項目がある場合、`item 1 (a)` を削除すると、`item 2` は元の `item 3 (c)` になり、`item 1` は元の `item 2 (b)` になり、`[b, c]` のようになります。したがって、`Item 1` はスキップされます。このように、2 番目の項目はすべてスキップされ、その結果、インデックスが最後に残っている項目よりも大きくなると、エラーが返されます。

このため、項目を削除する場合は、必ず逆方向に反復処理する必要があります。この方法で、確認が行われていないすべての項目では、最後まで元のインデックスが維持されます。

最後に、情報の種類 `with in table` (`with` と `in` の間にはスペースがあります) は、ブール値 `true` と `false` を返すのではなく、文字列 `true` と `false` を返すことに注意してください。

範囲を使用する

AppleScript では、範囲は動的に扱われないため、範囲に変更を加えたときは新しい変数に割り当てる必要があります。

変更を加えた範囲をキャプチャして新しい変数に割り当てる

次の例では、`Text Range` オブジェクトを使用して、重複する段落を削除します。以下の例を比較してください。

Visual Basic for Applications (VBA)

```
Dim AmountMoved As Long
Dim myRange As Range

'start with first paragraph and extend range down to second

Set myRange = ActiveDocument.Paragraphs(1).Range
AmountMoved = myRange.MoveEnd(unit:=wdParagraph, Count:=1)

'loop until there are no more paragraphs to check

Do While AmountMoved > 0

    'if two paragraphs are identical, delete second one
    'and add the one after that to myRange so it can be checked

    If myRange.Paragraphs(1).Range.Text = _
        myRange.Paragraphs(2).Range.Text Then
        myRange.Paragraphs(2).Range.Delete
        AmountMoved =
            myRange.MoveEnd(unit:=wdParagraph, Count:=1)
    Else
        'if two paragraphs aren't identical, add the one
        'after that to my range, so it can be checked, and
        'drop the first one, since it is no longer of
        'interest.
        AmountMoved =
            myRange.MoveEnd(unit:=wdParagraph, Count:=1)
        myRange.MoveStart unit:=wdParagraph, Count:=1
    End If

Loop
```

AppleScript

```
tell application "Microsoft Word"
    --start with first paragraph and extend range down to second
    set myRange to text object of paragraph 1 ~
        of active document
    set rangeEnd to end of content of myRange

    --in AppleScript a new range is made and returned, cannot alter
    --ranges in place, so redefine myRange to the new range
    set myRange to (move end of range myRange ~
        by a paragraph item count 1)
    set newRangeEnd to end of content of myRange
    set amountMoved to newRangeEnd - rangeEnd
```

```
set rangeEnd to newRangeEnd

--loop until there are no more paragraphs to check
repeat while amountMoved > 0

    --if two paragraphs are identical, delete second one
    --and add the one after that to myRange so it
    --can be checked
    if content of text object of paragraph 1 of myRange =
        content of text object of paragraph 2
        of myRange then
        delete text object of paragraph 2 of myRange
        set myRange to text object of paragraph 1
        of myRange
        set rangeEnd to end of content of myRange
        set myRange to (move end of range myRange
            by a paragraph item count 1
        set newRangeEnd to end of content of myRange
        set amountMoved to newRangeEnd - rangeEnd
        set rangeEnd to newRangeEnd

    else
        --if two paragraphs aren't identical, add
        --the one after that to my range, so it can
        --be checked, and drop the first one, since
        --it is no longer of interest.
        set myRange to (move end of range myRange
            by a paragraph item count 1
        try
            set newRangeEnd to end of content
            of myRange
            set amountMoved to newRangeEnd - rangeEnd
            set rangeEnd to newRangeEnd
            set myRange to (move start
                of range myRange by a paragraph
                item count 1
        on error -- errors because can't get
            --newRangeEnd when move end of range
            --is missing value at end of document
            set amountMoved to 0
        end try
    end if
end repeat

end tell
```

これら2つの例を比べると、AppleScript では範囲を置き換えることはできず、同じ範囲が維持される点が異なります。範囲は動的には扱われません。範囲を別の範囲に変更することはできますが、変数 myRange を変更した範囲に再設定する必要があります。

Word AppleScript

再設定する場合には、新しい範囲を取得できるように、範囲を変更するコマンドが新しい範囲を返す必要があります。AppleScript では、**set range** などのコマンドで変更された範囲を返します (VBA ではこの結果を返す必要はありません)。そして、**set range** の実行前に範囲が割り当てられていた変数と同じ変数 (myRange など) に、新しい範囲を割り当て直します。後は VBA と同じです。

set range コマンドを使用する

たとえば次のコードでは、作業中の文書の 3 番目の段落末が範囲の最後となるように myRange を再定義します。

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.SetRange Start:=myRange.Start, _
                End:=ActiveDocument.Paragraphs(3).Range.End
```

VBA では、3 番目の段落末が範囲の最後となるように、myRange の定義を変更するだけです。

SetRange によって結果を返すコードは不要です。myRange が現在の範囲となるため、このコードは必要ありません。

AppleScript では、myRange を **text object** of paragraph 1 (段落 1 のテキスト範囲) に設定してさらに段落を含めるよう拡張することはできません。少なくとも **set range** コマンドでは不可能です。これを実行しようとする、Microsoft Word がクラッシュします (**move end of range** コマンドの場合は当てはまりません)。myRange 変数の参照先は最初の段落のままであり、この変数に同時に他の範囲を指定することはできません。

この問題を解決するには、次の例に示すように、同じ開始位置と終了位置を設定した別の **range** を作成し、その範囲を変更します。

```
tell application "Microsoft Word"
    set par1Range to text object of paragraph 1 of active document
    set myRange to create range active document start (start of content ~
        of par1Range) end
    (end of content of par1Range)
    set myRange to set range myRange start (start of content of myRange) ~
        end (end of content of (get text object of paragraph 3 ~
            of active document))
    content of myRange
end tell
```

このコードは、クラッシュすることなく動作します。ここでは、myRange 変数を par1Range と同じサイズに設定します。ただし、特定の段落を指定するのではなく、同じ開始位置と終了位置を指定します。その後、**set range** を使用して終了位置を再定義できます。ただし、paragraph 3 自体を指定するのではなく、整数を指定します。**set range** コマンドは、結果を新しいテキスト範囲で返します。現在の範囲は変更されません。

範囲を myRange として引き続き参照するには、myRange をこの結果の値に再定義する必要があります。または、別の変数をこの結果の値に設定することもできますが、VBA マクロで myRange により範囲を参照している場合は、スクリプトの変更も最小限にすることをお勧めします。

移動文字数を取得する

次の例のように、**set range** をまったく使用しない方法もあります。

```
tell application "Microsoft Word"
    set par1Range to text object of paragraph 1 ↵
        of active document
    set par3Range to text object of paragraph 3 ↵
        of active document
    set myRange to create range active document start ↵
        (start of content of par1Range) end ↵
        (end of content of par3Range)
    content of myRange
end tell
```

move end of range および **move start of range** のようなコマンドは、通常であれば VBA の **MoveEnd** および **MoveStart** と同じように動作しますが、新しい範囲を返す必要があるという点では、VBA のように移動文字数を返すことができません。ただし移動文字数の特定は難しいことではなく、範囲のテキストオブジェクトの **end of content** を移動前と移動後の両方で取得できるので、一方からもう一方を引いて差分を求めることができます (amountMoved)。この差分が移動量になります。

この場合は、減算を行った後に、`set rangeEnd to newRangeEnd` などのときに `rangeEnd` と `newRangeEnd` の変数を更新する必要があります。更新しないと変数が不足することになります。したがって、同じことを行うのに数行のコードを追加する必要があります。

VBA では、次のコードで移動文字数を取得できます。

```
Set myRange = ActiveDocument.Paragraphs(1).Range
AmountMoved = myRange.MoveEnd(unit:=wdParagraph, Count:=1)
```

このコードは、`myRange` を最初の段落の **range** に設定した後、**MoveEnd** を使用して次の段落の前まで移動します。ここでも、新しい範囲が現在の範囲となります。`myRange` は動的な参照として引き続き存在するため、再定義する必要はありません。**MoveEnd** メソッドにより、前方への移動文字数を表す整数を返すことができます。

今回は、AppleScript で独自の範囲を作成する必要はありません。最初の段落の **text object** に設定されたテキスト範囲に対し、**move end of range** を使用できます。クラッシュは起こりません。ただし、この場合も現在の範囲を変更することはできません。次の例に示すように、`myRange` または別の変数を設定して、**move end of range** で返される新しい範囲を取得する必要があります。

```
tell application "Microsoft Word"
    set myRange to text object of paragraph 1 of active document
    set myRange to (move end of range myRange ↵
        by a paragraph item count 1)
end tell
```

Word AppleScript

これは問題なく動作しますが、コマンドでは新しい範囲を結果として返す必要があるため、移動量を返すことができません。このため、スクリプトの大部分が移動量の計算に費やされることとなります。移動のたびに `rangeEnd` を取得して再定義するという方法は難しくはありませんが、やや面倒です。このトピックの最後に、VBA と AppleScript コマンドとの違いを踏まえて改良したコードを紹介していますが、そのシンプルな方法を採用することによって、この問題も解決されます。

エラーをトラップしてスクリプトを終了する

この例でもう 1 つ注意する必要があるのは、文書の最後の処理です。VBA では、最終の段落記号で最後の `MoveEnd` を実行してもエラーにはならず、単に 0 が返されます。AppleScript では、`move end of range` は同様にエラーにはなりません、`missing value` が返されます。これは、作成できず存在しない新しい範囲に対する結果であり、一種の null 値です。その次の行で、存在しない範囲の `get end of content` を試行するときエラーになります。

この問題を解決するには、`try/on error` ブロックでエラーをトラップし、`amountMoved` 変数を便宜上 0 に設定します。他の方法もありますが、この方法では VBA マクロと同じ構造を維持できます。また、使用される頻度は少ないものの、AppleScript の `repeat/while` ループを使用してみる機会にもなります。

スクリプトを書き直して、出現する `move end`、`move start`、および `delete` をすべて保持し、`get end of content` および `amountMoved` の計算をすべて省略することもできます。この場合は、最後にトラップしたエラーに応じて、`while` を使用しない単純な `repeat` ループでスクリプトを終了します。

このタスクの最適なコード例

次の例は、AppleScript に合わせて改良した最適なコードです。

```
tell application "Microsoft Word"
    --start with first paragraph and extend range down to second
    set myRange to text object of paragraph 1 of active document
    set myRange to (move end of range myRange by a paragraph item count 1)

    --loop until there are no more paragraphs to check
    repeat
        --if two paragraphs are identical, delete second one
        --and add the one after that to myRange so it can be checked
        if content of text object of paragraph 1 of myRange = ~
            content of text object of paragraph 2 of myRange then
            delete text object of paragraph 2 of myRange
            set myRange to text object of paragraph 1 of myRange
            set myRange to (move end of range myRange ~
                by a paragraph item count 1)
        else
            --if two paragraphs aren't identical, add the one
            --after that to my range, so it can be checked,
            --and drop the first one, since it is no longer
            --of interest.
            set myRange to (move end ~
                of range myRange by a paragraph item count 1)
```

Word AppleScript

```
        try
            set myRange to (move start of range myRange -
                by a paragraph item count 1)
        on error -- last paragraph
            --(missing value, so can't move start)
            exit repeat -- finish
        end try
    end if
end repeat
end tell
```

逆方向に反復処理を行って行を削除する

AppleScript では、逆方向に反復処理を行って行を削除しないと、項目のスキップによるエラーが発生しやすくなります。以下の例を比較してください。

VBA

```
Public Sub DeleteEmptyRows()

Dim oTable As Table, oRow As Range, oCell As Cell, Counter As Long, _
NumRows As Long, TextInRow As Boolean

' Specify which table you want to work on.
Set oTable = Selection.Tables(1)
' Set a range variable to the first row's range
Set oRow = oTable.Rows(1).Range
NumRows = oTable.Rows.Count
Application.ScreenUpdating = False

For Counter = 1 To NumRows

    StatusBar = "Row " & Counter
    TextInRow = False

    For Each oCell In oRow.Rows(1).Cells
        If Len(oCell.Range.Text) > 2 Then
            'end of cell marker is actually 2 characters
            TextInRow = True
            Exit For
        End If
    Next oCell

    If TextInRow Then
        Set oRow = oRow.Next(wdRow)
    Else
        oRow.Rows(1).Delete
    End If

Next Counter
```

Word AppleScript

```
Application.ScreenUpdating = True
```

```
End Sub
```

AppleScript

```
tell application "Microsoft Word"
  activate
  --Specify which table you want to work on
  set theTable to table 1 of selection -- or table 1 of active document
  set numRows to number of rows of theTable -- faster than counting rows
  set screen updating to false

  --iterate backwards because deleting!
  repeat with i from numRows to 1 by -1
    set rowText to text object of row i of theTable

    set status bar to "Row " & i
    set textInRow to false

    --you NEED the explicit gets if not setting variables!
    repeat with theCell in (get every cell of rowText)
      if (count of (get content of text object -
        of theCell)) > 2 then
        -- end of cell marker is 2 characters,
        -- count faster than length
        set textInRow to true
        exit repeat -- no need to check other cells
      end if
    end repeat

    if not textInRow then
      delete row 1 of rowText
    end if
  end repeat
  set screen updating to true
end tell
```

AppleScript の例では、項目の削除を行っているので、repeat ループを別の方法で逆方向に行う必要があることに注意してください。

AppleScript には、Visual Basic for Applications (VBA) の **.Next(wdRow)** に相当するものはありません。これがあると、ループ内の場所を問わず、または行を削除したかどうかに関係なく、次の行へ移動できます。

repeat ループはリスト内のインデックス順のオブジェクトにアクセスする

すべての repeat ループは、参照であるオブジェクトで構成されます。次の例のような事前に作成されたリストを反復処理する場合であっても同じです。

```
set allRows to every row of theTable
```

ここでは、多くの場合と同様に、参照にはインデックス {row 1 of table 1, row 2 of table 1, row 3 of table 1, ...} が使用されています。次のコードを記述し、リストの item i を取得してリスト作成時と同じ項目が返されることを期待しても、そのようにはなりません。

```
set allRows to every row of theTable
repeat with i from 1 to (countallRows)
    set theRow to item i of allRows
```

リストのインデックスはリスト内のオブジェクトを削除すると変化する

元の項目は、row i of theTable への参照です。ループ中にテーブルを変更した場合、row i of the table が再評価されるときには、前の row i と異なる内容になっています。その結果、項目がスキップされます。

これは、Microsoft Entourage のような、ほとんどのアプリケーション参照がデータベース内で一意の ID で "ハードコーディングされた" オブジェクトへの参照になっているアプリケーションには当てはまりません。これは、データベース アプリケーションを扱う際の "贅沢" であると言えます。オブジェクトへの参照はすべて、名前、インデックス、または他の識別子のどれを使用しても、"正規" の ID 参照に解決されます。たとえば、あるフォルダ内の allMessages のリストが {message id 12345, message id 12346, message id 12347, message id 12348} であるとし、このリスト内の 2 番目の項目を削除してもリストからは削除されないため、allMessages の item 3 を取得すると id 12348 ではなく id 12347 が取得されます。

Microsoft Word では、Finder や他のほとんどのアプリケーションと同様に、参照のインデックスが再評価され、項目がスキップされます。その後、最後のインデックスは存在しなくなるので、AppleScript で最後のインデックスを見つけられないというエラーが発生します。逆方向の反復処理では、削除された項目のインデックスよりも小さいインデックスは影響を受けないため、この問題が解決されます。

インデックスの代わりに一意の ID を使用する

active document のようなアプリケーション参照を使用するなどの方法により、各参照を再評価する Word の仕様を回避できます。

たとえば、次の手順を実行すると、Word では theDoc が再評価され、前面に移動した新しい文書に設定されるので、元の参照が失われます。

1. 変数 theDoc を、前面にある文書を示す active document に設定する。
2. 前面の文書の window 1 を Dock にしまう (collapse)。
3. theDoc を再度呼び出し、アクティブ化する。

Word AppleScript

この問題を回避するには、現在の **active document** を一意に識別するものを見つけ、変数設定時にその識別子で参照を指定します。どのような文書でも、最適な一意の ID はその文書の **name** です。2 つの文書に同じ名前を指定することはできません。変数は、次のようにして設定できます。

```
tell application "Microsoft Word"
    set theName to name of active document
    set theDoc to document theName
end tell
```

ここで、変数 theDoc は現在アクティブな文書に "ハードコーディングされた" 状態で保持されます。このウィンドウを Dock にしまい、他の文書が **active window** になっても、変数 theDoc は最初に設定されていた文書を引き続き参照します。この文書を再度アクティブにした場合や、文書に任意の操作を行った場合でも、この変数は同じ文書を参照したままです。

行削除時のエラーを防ぐために逆方向に反復処理する

一方、allRows のリストを含む例では、個々のリスト項目は、{row 1 of table 1, row 2 of table 1, ...} のように、リスト内のインデックス順の行を参照しています。このリストが呼び出されると、再評価されて項目がスキップされる結果となり、エラーが発生します。この問題を回避するには、逆方向に反復処理する必要があります。ところで、行を一意に識別する機能はありません。少なくとも **every row** をリストとして取得する場合には、一意に識別できません。行を削除すると常に再評価されるのは各行のインデックスであるため、逆方向の反復処理が唯一の方法になります。

メモ Dictionary の **table** の定義には、**row** 要素を "名前で" 取得できると記載されていますが、これは実際は "インデックスで" という意味です。行には **name** プロパティはありません。

repeat with i from numRows to 1 by -1 構文内の i は、明示的に増加させる必要がないカウンタであるため、次の VBA ステートメントに相当する AppleScript の初期化ステートメントも必要になります。

```
Set oRow = oTable.Rows(1).Range
```

これらの要件は、次の AppleScript の repeat ループ内で満たされます。

```
set rowText to text object of row 1 of theTable
```

同様に、textInRow が **true** であるかどうかをチェックする必要はありません。i カウンタは自動で増加するため、true の場合でも何も操作は必要ありません。

screen updating 機能と **status bar** 機能は、AppleScript でも VBA と同様に動作します。ただし、前の調整を行ったことにより、ステータスバーに 1 までのカウントダウンが表示されます。これは終了まで、どのくらいの数が残っているかを示します。

検索と置換を使用する

検索と置換を使用してアイテムを削除し、repeat ループや逆方向への反復処理を回避できる場合があります。

このマクロでは、空の段落を文書から削除します。文書内の表の内側と周囲から空の行を削除するための追加コードも含まれていますが、ここでは、-1 によって逆方向に repeat ループの反復処理をする必要があります。

このマクロは Microsoft Word 2004 for Mac では動作しません。この問題は、Windows 版の Microsoft Word 用に作成されたマクロを使用する際にとどき発生します。これについても説明します。

Windows 版の Word では、[検索] ボックスでワイルドカードを使用する場合に、“段落記号”を意味する特殊文字 ^p を使用できないという問題を回避できます。たとえば、段落記号が 2 つ以上出現していることを意味するワイルドカード {2,} を使用するとします。Windows 版の Word では、[検索] ボックスで、「^p」の代わりに「^13」と入力して、式“^13{2,}”を作成し、[ワイルドカードを使用する]をクリックします。

しかし、これは Word 2004 では、UI、Visual Basic for Applications (VBA)、AppleScript のいずれにおいても動作しません。ワイルドカードを使用しない ^13 は使用できますが、ワイルドカードを使用する ^13{2,} は使用できません。UI では、ワイルドカードを使用した場合、使用している式が有効でないというエラーメッセージが表示されます。スクリプトでは、find が検索の実行メッセージを理解しないというエラーメッセージが表示されますが、これは UI と同じ内容のエラーです。Macintosh では、^13 は ^ と同じことであり、同じ制限を受けます。

これを解決するには、2 つの段落記号の連続 ^p^p を探し、これが見つからなくなるまで Replace All を繰り返します。検索と置換を使用すると、段落全体でループを実行するより処理速度がかなり速いため合理的です。次に、VBA マクロとこれを変更した AppleScript バージョンを示します。以下の例を比較してください。

VBA

```
'Note that using Find and Replace is dramatically faster
'than cycling through the Paragraphs collection

'Replace: ^13{2,} with ^p, which replaces all occurrences
'of two or more consecutive paragraph marks with one paragraph mark

With ActiveDocument.Range.Find
    .Text = "^13{2,}"
    .Replacement.Text = "^p"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchAllWordForms = False
    .MatchSoundsLike = False
```

Word AppleScript

```
.MatchWildcards = True
.Execute Replace:=wdReplaceAll
End With

'However, you can't use Find & Replace to delete the first or last
'paragraph in the document, if they are empty. To delete them:

Dim MyRange As Range
Set MyRange = ActiveDocument.Paragraphs(1).Range
If MyRange.Text = vbCr Then MyRange.Delete

Set MyRange = ActiveDocument.Paragraphs.Last.Range
If MyRange.Text = vbCr Then MyRange.Delete
```

AppleScript

```
tell application "Microsoft Word"
  -- replace ^p^p with ^p to replace all occurrences of two
  -- consecutive paragraph marks with one paragraph mark
  -- repeat until done

  repeat
    set textObject to (text object of active document)
    -- redo each time
    if (content of textObject) ¬
      does not contain (return & return) then
      exit repeat -- done
    end if

    set findObject to (find object of textObject)
    -- we need a separate executefind on it,
    -- so best set a variable to it so we just get
    -- it once
    tell findObject
      set {content, content of its replacement, ¬
        forward, wrap, format, match case, ¬
        match whole word, ¬
        match all word forms, match sounds like, ¬
        match wildcards} to {"^p^p", "^p", true, ¬
        find continue, false, false, false, ¬
        false, false, false}
    end tell
    execute find findObject replace replace all

    -- Find/Replace cannot delete first or last
    -- paragraph if empty, so:
    set myRange to text object of paragraph 1 ¬
      of active document
    if content of myRange = return then ¬
      delete myRange
```

```
try
    set myRange to text object ↵
        of paragraph -1 of active document
    if content of myRange = return then ↵
        delete myRange
end try
end repeat
end tell
```

置換する場合は、特殊文字 ^p を段落の最後に適切に挿入する必要があるということに注意してください。^13 を使用すると、一見問題がないように見えても実は壊れた段落記号が挿入されます。これは単なる改行記号で、Word の段落記号に保存されている追加情報がすべて含まれていない可能性があります。

この操作を実行する必要回数はわからないので、単純な repeat ブロックを使用します。次のステートメントや、同じ内容に設定した変数を使用すると、毎回再評価が行われず、repeat ループは終了しません。

```
repeat while (get text object of active document does not contain return & return)
```

代わりに、repeat ループ内で textObject 変数をリセットして、強制的に再評価を実行し、その条件の下で exit repeat を実行します。

これは、ループ内で、最初と最後の段落が空であれば削除するコードも実行する必要があることを意味しています。このようにしないと、最後の段落が空の場合、テキスト範囲には 2 つの段落記号が最後まで残り、repeat ループから抜け出すことができなくなります。

最後の空の段落を削除するコードは、try/end try で囲む必要があります。これは、myRange を削除しようとするエラーになるためです。段落 (単なる改行コード) は実際に削除されますが、エラーが発生します。したがって、簡単な try/error でこの問題を回避します。

上のスクリプトを使用して、文書全体ではなく、選択したテキストで検索と置換を実行するには、text object of active document の代わりに次のステートメントを使用します。

```
set textObject to (text object of selection) -- redo each time
```

ブックマーク

ブックマークを使用する

最初に、ブックマークを表示するように環境設定を変更します。Microsoft Word 2004 および 2008 では、次の手順で変更できます。

1. [Word] メニューの [環境設定] をクリックし、[表示] をクリックします。
2. [表示] グループの [ブックマーク] チェック ボックスをオンにします。

以下は、Visual Basic for Applications (VBA) と AppleScript のコード サンプルです。最初の 2 つのコードは、文書内でブックマークとして選択した位置 (プレースホルダ ブックマーク) にテキストを挿入します。以下の例を比較してください。

VBA

```
ActiveDocument.Bookmarks("myBookmark").Range.Text= "Inserted Text"
```

AppleScript

```
tell application "Microsoft Word"
    set content of text object of bookmark ↵
        "myBookmark" of active document to "Inserted Text"
end tell
```

次のコードは、ブックマークとして選択した範囲 (範囲選択ブックマーク) にテキストを挿入し、ブックマークを再作成します。以下の例を比較してください。

VBA

```
Dim bmRange As Range

Set bmRange = ActiveDocument.Bookmarks("myBookmark").Range
bmRange.Text = "Inserted Text"
ActiveDocument.Bookmarks.Add _
    Name:="myBookmark", _
    Range:=bmRange
```

AppleScript

```
tell application "Microsoft Word"
    set bmRange to text object of bookmark "myBookmark" of active document
    set content of bmRange to "Inserted Text"
    make new bookmark at active document with properties ↵
        {name:"myBookmark", text object:bmRange}
end tell
```

動的な範囲はブックマークのスクリプトに影響する

上の AppleScript コードを使用すると、文書内のまったく異なる場所にプレースホルダブックマークが挿入されます。AppleScript では、範囲を変更するとその範囲に対する変数を維持できません。範囲は動的ではありません。ブックマークの **text range** のテキストを変更するとすぐに、bmRange に割り当てられた範囲が失われます。

これを確認するために、2つのテストを実行します。まず、範囲選択ブックマークのコードブロックに次の記述を追加します。

1. tell ブロック内の、1行目で bmRange を定義した直後に、ステートメント properties of bmRange を挿入して、2行目を新たに作成します。
 2. 新たに作成した2行目の後にステートメント end tell を記述してスクリプトを終了します。
-

Word AppleScript

結果は、このテキストの範囲に対応する膨大な数のプロパティのレコードになります。

2 つ目のテストでは、元の範囲選択ブックマークのコード ブロックに次の記述を追加します。

1. tell ブロック内の、bmRange のコンテンツを "Inserted Text" に設定した直後に、ステートメント `properties of bmRange` を挿入して、3 行目を新たに作成します。
2. 新たに作成した 3 行目の後にステートメント `end tell` を記述してスクリプトを終了します。

結果は、**content** プロパティが変更された膨大な数のプロパティのレコードが返されると予測できますが、bmRange が存在しないため、何の結果も得られません。そのため、このコマンドを使用して、新しいブックマークの text object を定義することはできません。

ブックマークの場所を特定して独自のブックマークを作成する

別の試みとして、次の記述を追加します。

1. 変更を加える前に、bmRange の **start of content** を特定します。これはブックマークの **start of bookmark** と同一です。
2. bmRange の長さを測定します。つまり、挿入する文字列をカウント (**count**) します。
3. 測定した長さを **start of content** の結果に追加し、新しい **end of content** と新しい **end of bookmark** を取得します。

メモ **start of content** は挿入位置で、その数値は前にある文字数と同じです。文書の最初の挿入位置と **start of content** は 0 であって、1 ではありません。したがって、新しい **end of content** を取得するために、新しいテキストの長さを追加した後で 1 を差し引く必要はありません。

start of bookmark プロパティと **end of bookmark** プロパティの 2 つの数値を使用して、新しいブックマークを作成します。**text object** プロパティは読み取り専用ですが、これらのプロパティは値を割り当てるように設計されているので便利です。

この手順全体のスクリプトを次に示します。

```
tell application "Microsoft Word"
    set bmRange to text object of bookmark "myBookmark" of active document
    set bmStart to bmRange's start of content
    set content of bmRange to "Inserted Text"
    set bmEnd to bmStart + (count "Inserted Text")

    make new bookmark at active document with properties ↵
        {name:"myBookmark", start of bookmark:bmStart, ↵
            end of bookmark:bmEnd}
end tell
```

このスクリプトは問題なく動作します。ただし、AppleScript では範囲が動的でないため、最初の VBA を変換したものを編集する必要があります。別の方法として、多数のプロパティ、クラス、コマンドの中から選択して利用する方法があります。

解決策が見つからないということは、ほとんどありません。マクロを VBA から AppleScript に変換するときの問題が発生した場合、動作するしくみを把握していれば必ず解決策は見つかります。