Computers & the Humanities 281

Text Properties and Find Command

Text Units

Revolution recognizes the basic units of text: characters, words, lines, etc. However, Revolution has its own "definition" of these terms:

- **Character:** An individual letter, number, symbol, etc. Spaces count as characters, as well as the invisible Enter or "carriage return" at end of lines.
- **Word:** a sequence of characters delimited by Space or Return. Words are not delimited by punctuation, as punctuation marks are included with the word.
- **Line:** a sequence of characters delimited by Return.
- **Item:** a sequence of characters delimited by Comma.

Text Chunks

A "chunk" is Revolution's generic term for any unit of text: characters, words, lines, etc. You refer to chunks of text by specifying the sub-units as needed. Revolution goes by its natural hierarchy, not order, when referring to chunks, so something similar to `select line 3 of word 5` works just fine, even though it may read like nonsense.

Revolution recognizes the ordinals first through tenth, which means that `fourth word = word 4` (i.e., they are functional equivalents, the former being somewhat more readable). **NOTE:** It is not **`the`** `fourth word` ("the" **cannot** be used; it will result in a horrible error).

Other forms in the same class provide some extra capabilities:

- **last:** last unit specified
- **middle:** middle unit specified; this shifts up when there are an even number of units (e.g., `the middle word of "red yellow green mauve"` is "green")
- **any:** references a random unit specified

You can specify a range of text using the `to` keyword. For example, `word 5 to 7` or `char 14 to 29`. Note that you do **not** repeat "word" (e.g., you don't say `word 5 to word 7` and you do **not** use an "s" in the character range (e.g., you don't say **`chars`** `14 to 29` but you do say **`char`**).

Text Functions

Revolution provides some useful functions you can use to handle text

- **Length:** returns the length (number of characters) of a text chunk (e.g., `put the length of line 12 of field whatever`. **Note:** "the" is required before length in this construct.
- **Number:** used to obtain the number of a text unit in a larger chunk (e.g., the number of characters in a word, the number of words in a line, the number of lines in a field, the number of words in a field, the number of items in a line, etc.) You can use either `number of lines` **`in`** `field whatever` or `number of words` **`of`** `field whatever` in this construct. Both are completely interchangable. **Note:** The use of "the" is optional in conjunction with number (e.g., **`the`** `number of lines in field whatever` works just as well as the example given above).
- **Offset** locates a pattern string in a larger string (e.g., `offset("crown",field whatever)`
  - Returns beginning character number where first occurrence was found
  - Strictly character match, not word
  - Returns zero when pattern not found
  - Case is ignored
  - Does not ignore accent marks or punctuation

  There are three variants of this function:
  - itemOffset
  - lineOffset
  - wordOffset

  These all work the same as offset, except instead of returning the number of characters from the beginning of the search container, they return the number of items, lines, or words respectively.

Text Booleans

The text booleans `is in` and `contains` allow you to test for a pattern in a string. Both have the same purpose and they reflect the logic of English syntax. For example, `if "bob" is in fld 1 then beep` and `if fld 1 contains "bob" then beep` are identical. Case is ignored (it can be capitalized in the field and not capitalized in your script), but accents are not. Spaces and punctuation are not ignored, either. It handles `empty` in a reasonable fashion, e.g., a field with anything in it does not contain `empty`.

Find Command

The find command is a formal Transcript statement used to locate a certain string in the fields within a stack. This is the basic syntax:

```
find [form] <textToBeFound> [in field]
```

where

`[form]` is any of the form words described below.
`<textToBeFound>` is any expression that evaluates to a string.
`[in field]` a specfic field optionally specified in which the search will be confined.

When the command is executed, Revolution places a box around the first instance of the string that it locates. It keeps track of latest find and subsequent finds continue from most recent. Otherwise it begins with the first field on the current card. The command follows a certain hierarchy as it searches: first, grouped fields in order; second, card fields in order; then on to subsequent cards.

As stated earlier the search can be confined to a specific field. Without specifying the specific field, Revolution finds the first occurrence in search order. With a field specified, then the search is confined to that field alone. If not found and the field is part of a group, then it continues searching that field on subsequent cards.

The form `find empty` clears the current find operation, meaning a subsequent search will begin all over again in the search hierarchy, rather than beginning from the most recently found text. Consequently, the box drawn around the text found is removed.

There are five forms of the find command:

- **normal:** finds text at beginning of word; default if form not specified
- **chars:** finds text anywhere in word
- **word:** finds text as complete word
- **string:** finds exact text that contains partial words
- **whole:** finds exact text as complete word or words

Find uses a complex technique to speed searching through large amounts of data. However, its behavior with the different forms may seem somewhat peculiar when the search text consists of more than one word (i.e., it contains a space). With normal, chars, and word, Revolution breaks the search into words and performs a subordinate search as necessary: It looks for first word and then looks for the second word within same scope, etc. (if a field is specified, they must be in the same field; otherwise they can be present on the same card for the search to be successful). **Note:** the rectangle is placed only around the first word. The other two forms (string and whole) do not break this way (but also do not take full advantage of search speed).

Found Functions

The find command, if successful, results in a box being drawn around the text found. This is done to give access to the chunk of text that was found. Four functions exist to achieve that end:

- **foundText:** gives the text that was found (enclosed in rectangle)
- **foundChunk:** gives chunk expression of the text found
- **foundLine:** gives the line expression where the text was found
- **foundField:** gives the field where the text was found

Another useful function that is related to these is the **foundLoc**, which gives x and y coordinates of upper left hand corner of found rectangle.

If the search is not successful (i.e., the text was not found), then `not found` is returned in the `result` function.

Selection Functions

While the text chunk may have been found, it is not selected, though (i.e., you can't cut, copy, paste, etc.). In order for that to happen, you must use the `select` command. This command selects a chunk of text the same as if you clicked in the field and dragged the mouse across the text. You can then cut/copy/paste, etc. You can use the select command even if the field is locked.

Four functions give access to the selected chunk. They are similar to the found functions:

- **selectedText:** gives the text that was selected (hilighted)
- **selectedChunk:** gives chunk expression of the text selected
- **selectedLine:** gives the line where the text selected appears
- **selectedField:** gives the field where the text selected appears

You can select the entire contents of a field, but not with `select field`, as this selects the field as would happen with the pointer tool (to edit its properties). To accomplish the former you must use `select the text of field whatever`. You can also use `select before` or `select after` to exert greater control over where the insertion point is placed in the text. The selection functions can be used in conjunction with the found functions to find and copy a string or unit:

```
select the foundChunk
put the selectedText into thisVariable
```

Note that for all of these functions, **the** is required.

Click Functions

Click functions are a set of functions similar to found and selected functions. They return information from clicks on a locked text field:

- **clickText:** word or grouped words clicked
  Note: This uses a more liberal definition of word than Revolution usually uses (where characters are delimited by punctuation as well as space & return).
- **clickchar:** gives character clicked
- **clickChunk:** gives chunk expression of the word clicked

- **clickLine:** gives the line clicked in the field
- **clickField:** gives the field clicked
- **clickLoc:** gives the x,y coordinates of the location of the click

The values, of course, vary depending upon what is clicked. Note that some of the functions are empty in special cases: clicking on a space, clicking at the end of a line, clicking in an empty field, etc.

Course Schedule
Main Page