jsCalc

an inline calculator for 4th Dimension

John Skinner

Version 1-6

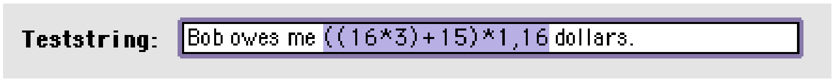
15.01.2003

Find the current version at
www.skinnerconsulting.com/downloads/4D/jsCalc/jsCalc.html

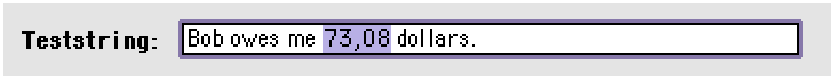
What is jsCalc?

jsCalc is an inline calculator: it reads a mathematical expression directly from the currently active field or variable in your existing input form, calculates the result, and pastes this back into the field. An example will show how this looks to the user:

Before …


Teststring: Bob owes me ((16*3)+15)*1,16 dollars.

… and after.

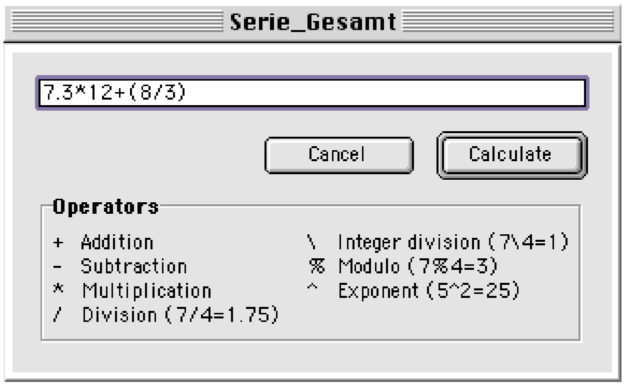

Teststring: Bob owes me 73,08 dollars.

The user types the expression directly into the field, selects it, runs **jsCalc** and sees the answer - right there where she needs it. No searching the desktop for a hand calculator, no remembering and mis-typing numbers: no trouble.

jsCalc is an end-user feature, to make your database easier to use and your customers happier. It's a (minor) selling point: your competitors' products cannot do this.

How does jsCalc work?

jsCalc can be started in two ways. When started normally, it uses the 4D **Last Object** Command to determine which field or variable in the frontmost window has the keyboard focus. If the field contains highlighted text, then only that substring is considered, otherwise the entire field content is considered. The original content is copied to the clip-board, in case it was valuable to the user. (The calculation cannot be undone with Command-Z, since no user action occurs.)


Serie_Gesamt

7.3*12+(8/3)

Operators

+ Addition	\ Integer division (7\4=1)
- Subtraction	% Modulo (7%4=3)
* Multiplication	^ Exponent (5^2=25)
/ Division (7/4=1.75)	

If the field is numeric, or if it is empty, then an input dialog is displayed where the user may type the expression. (Numeric fields cannot contain operator characters! You may type them at will, but 4D removes them as soon as the field loses the focus. Were the example here a numeric field, **jsCalc** would receive the string "7.31283".) The name of the referenced field or variable appears in the title bar.

You can also use the Developer Hook to specify a variable as source and destination for the calculation This lets you provide the functionality of **jsCalc** in places where Last Object does not work, e.g. within 4D Write areas. This is discussed on page 5 below.

The result of the expression is written back unformatted into the referenced field or variable, replacing highlighted content (or the whole string). jsCalc returns an integer to report error status (see page 6).

What can jsCalc do?

jsCalc supports all the 4th Dimension numerical operators, as listed in the Programming Language manual:

+	addition
-	subtraction
*	multiplication
/	floating-point division
\	long-integer division
%	modulo
^	exponentiation

(Most end-uers do not know "modulo", you may wish to include a note in your user documentation explaining this.) Internal calculations are performed using REALs (15-digit floating point). The result is rounded to 6 decimal places for display, but otherwise unformatted. (Actually, the rounding depends on the field length: the result 189,675 would be rounded to 0 decimal places = 190 for display in an Alpha(4) field.)

jsCalc supports multi-level nested parentheses ((1+((2*3)*4)*5)*6).

Whitespace, CR/LF characters and leading equals "=" characters will be ignored. Equals characters within the expression are seen as delimiters: the string will be evaluated from left to right, stopping at the equals sign.

What can jsCalc not do?

jsCalc does not follow Order of Operations: 2+3*7=35. Users must set parentheses to clarify the processing sequence: 2+(3*7)=23. This is mathematically incorrect, but it is consistent with 4th Dimension's user-mode math processing (e.g. the formular editor), which is why I wrote it so.

jsCalc supports only decimal numbers: Hex, Octal and Log are not supported. Since **jsCalc** is an end-user feature not a programmer tool, this is unlikely to be a significant limitation.

jsCalc can not evaluate Plug-In areas or web forms directly (since they are not "owned" by 4D). However, the **Developer Hook** feature allows you access to its functionality even in these cases. This is discussed on page 5 below.

Known problems

jsCalc is affected by a minor bug in the 4D toolbox: fields in sub-forms (related tables) are handled incorrectly. Should the keyboard focus be on a field in a sub-form, **Last Object** returns a pointer to a random object in the main form table. This is undetectable, since the pointer and its target are both valid (but incorrect) objects.

This will remain a problem until 4D fixes **Last Object**.

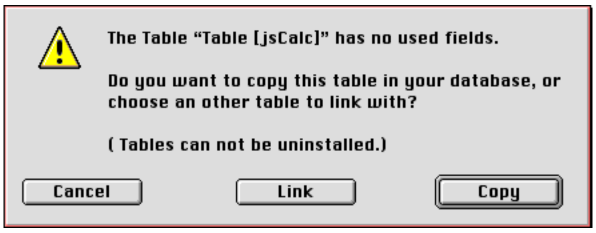
Installing jsCalc

jsCalc is a 4D Component, you may install it into any structure file using 4D Insider 6.7 or newer. The component was created with 4th Dimension 6.7.2 Always make a copy of your structure file before installing any component!

All Insider's *Moving…* settings should be either *Do not copy* (top half of the dialog) or *Ignore* (bottom half).

Warning: the component includes a table [jsCalc] which is never used. It's only there to contain the input dialog shown on page 2. **Do not let Insider create this table!**

During installation, Insider will ask how the table should be handled. You should choose **Link** to place the input dialog in an existing table (preferably one used for system tasks rather than user data). This allows you to uninstall the compo-onent cleanly should you not wish to use it.


The Table "Table [jsCalc]" has no used fields.

Do you want to copy this table in your database, or choose an other table to link with?

(Tables can not be uninstalled.)

Close the Insider window, and open your structure file with 4D.

Localising jsCalc

You may edit the public method **jsCalc_errorMsg** to adapt it to your language or to the "look and feel" of your database. The input dialog is also public, you may change its appearance and edit the button labels etc. Do not rename the variables, nor delete the boundary rectangle!

Uninstalling jsCalc

The **jsCalc** component can easily be removed from your structure files – provided that you read these notes before installing it. If not, the component can still be uninstalled but 4D Insider will have created a table which you are now stuck with.

Because the input dialog is in one of your own tables, Insider will not delete it when uninstalling the component. You must delete the dialog and its Style Sheets by hand.

Enabling jsCalc in your application

All that you need do to enable **jsCalc,** is to create a menu item or a button in your existing input form which calls **jsCalc_start** (and to give this a Tool Tip describing what it does). That's it, you've already finished! The component does everything else by itself, including calling the input dialog if necessary.

Please set the button's "Focusable" and "Tabable" attributes **off** (disabled). This is necessary for Windows users, otherwise Last Object finds the button instead of a text field.

The button method is very simple:

```
Case of
  :(Form Event = on clicked)
     C_INTEGER ($err)
     $err:=jsCalc_start (True)  `  with error messages
End case
```

The Boolean parameter controls the display of error messages: if TRUE then displayed, else the method is silent. You may check the returned error status and do your own reporting if desired (set the parameter to False).

The public method **jsCalc_example** can be used "as delivered" as the button method, please ensure that only the "On Clicked" form event is enabled. You may alternatively wish to call **jsCalc** from a menu item, in this case you will need to "wrap" **jsCalc_start** to capture the return value.

Enabling jsCalc "outside" your application

With just a little more effort, you can offer your users the functionality of **jsCalc** even in Plug-In areas e.g. within 4D Write. The **Developer Hook** makes this possible.

The method **jsCalc_start** takes an optional second parameter: a pointer to a field or variable. You can copy the expression to this variable, send it to jsCalc, then copy the result to where it belongs. Here's an example of applying **jsCalc** in an on-screen 4D Write area:

```
v_myVar:=WR Get Selected Text (w_Area)
$err:=jsCalc_start (True; ->v_myVar)
If ($err = 0)  ` successful?
   WR Insert Text (w_Area; v_myVar)
End if
```

This would work equally well (the necessary changes being made) with other Plug-Ins like AreaList Pro or 4D View.

Error status from jsCalc

jsCalc returns an integer error codePossible values are:

0	Normal successful completion
-1	User cancelled the input dialog (not really an error)
-29001	Overflow: either longint value in integer field, or SANE error in a real field, or the result string is too long for the field (e.g. "752168" in alpha 4)
-29002	Invalid expression: the string contained alphabetical characters
-29003	Unbalanced expression: too many (
-29004	Unbalanced expression: too many)
-29005	Badly formed expression: wrong number of operators
-29006	Divide by zero
-29007	No field has the keyboard focus. This typically occurs when the focus was previously on a button and the user clicks into an empty variable. Type a character or two and erase them, then call **jsCalc** again. This error also occurs if the button that launches **jsCalc** is itself focusable and/or tabable. Please disable these attributes for this button.
-29008	Wrong data type: field is neither String nor Numeric
-29009	Expression string is empty
-29010	String is not an expression (no operators)
-29011	Modulo by zero or a negative number

The phrase "normal successful completion" is an echo from my time as an Embedded-SQL programmer at Intergraph. I always wondered how "abnormal successful completion" or "normal unsuccessful completion" would look.

The Price

The **jsCalc** component is free, use it with my blessings. The source code is available for 15 Euros (US$15). It's even tax-deductible: include your formal company name and address (or mail me at john@skinnerconsulting.com) and I will send you a sales receipt.

Please send a cheque or money order (not cash) to
John Skinner
Bebelstraße 36
70193 Stuttgart
Germany

In Deutschland basierte Entwickler dürfen auch per Überweisung bezahlen, ich schicke Ihnen meine Bankverbindungsinfos per E-Mail Antwort zu.

Credits

My heartfelt thanks go to:

Jef Raskin, creator of the classic Apple Mac interface. A page of wishful thinking in his book "The Humane Interface" (Addison Wesley, 2000) inspired me to write **jsCalc** after years of moaning about needing both a computer and a calculator on my physical desktop. The book is not just for Mac developers: it discusses the Person/Machine Interface on a high theoretical level, with examples that range from computer software to car radios.

Many **members** of the german-language **4D Support List**, in particular Dietmar Harms, Klaus Renoldi, Mark Sebastian and Thomas Maul of 4D Germany, for helpful advice (some of it inadvertent).

Dietmar Harms, **Angelika Maser, Ortwin Zillgen** and **Michael Zink** for beta-testing (some of it inadvertent).

The Developer Hook was **Ortwin Zillgen**'s idea. I had been wondering for weeks, how to enable **jsCalc** in plug-ins like 4D Write; his suggestion was simple but very effective.

Any remaining errors are of course all my own work.

The legal stuff