Planning Guide

Apparo Fast Edit

Version 3.3

Content

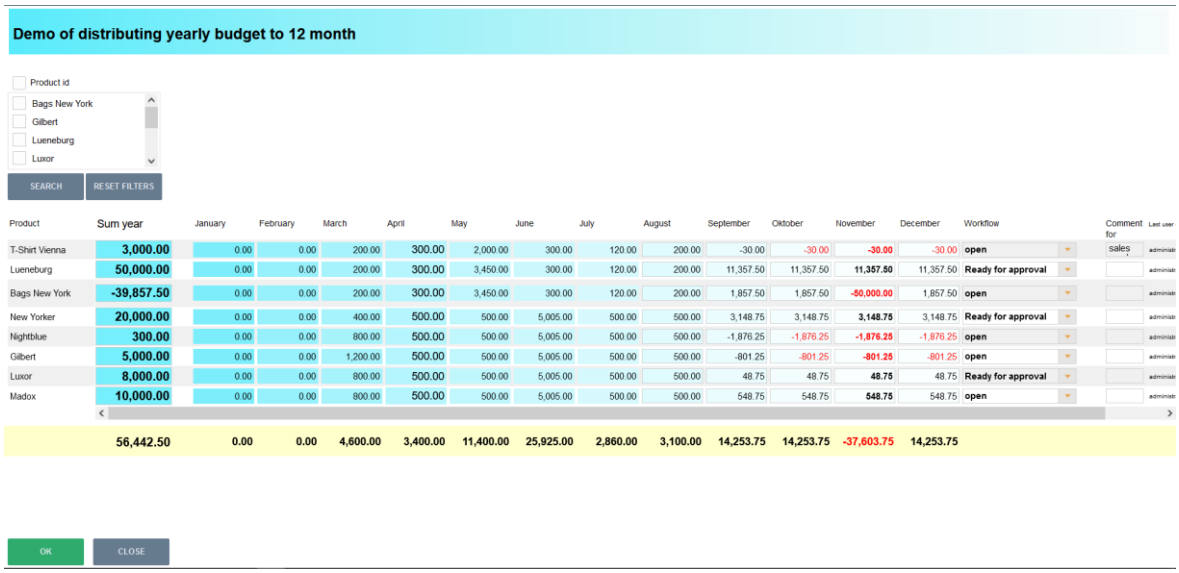
1 Intro

To map planning functionalities, we use Fast Edit's ability to **execute JavaScript on the client side**. For this, there are a number of additional JavaScript methods for reading/writing/in widgets in the edit and calculation area that allow us to define business logic.

Client-side means that JavaScript is executed in the web browser without interaction with the server.

Advantage: Distributing numbers to months and updating totals are done immediately after the user has adjusted a value with the Tab or Enter key. There are no waiting times because there is no communication with the server.

A small planning application is developed as an example for use.

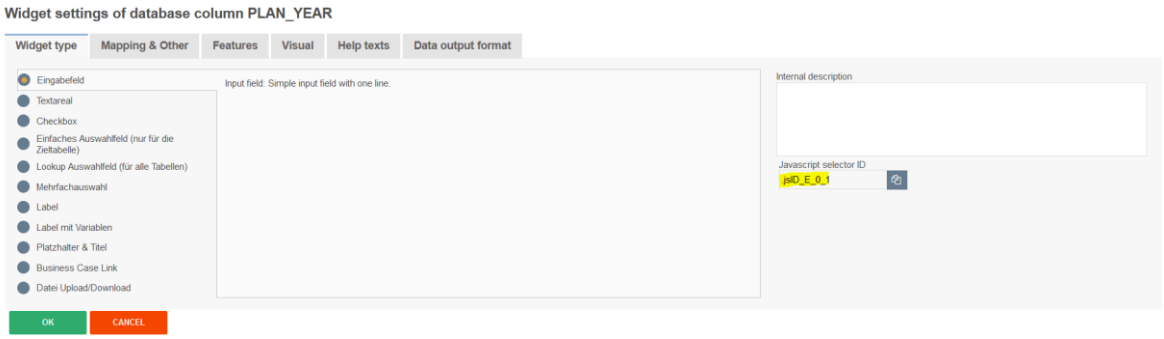


Product	Sum year	January	February	March	April	May	June	July	August	September	Oktober	November	December	Workflow	Comment for
T-Shirt Vienna	3,000.00	0.00	0.00	200.00	300.00	2,000.00	300.00	120.00	200.00	-30.00	-30.00	-30.00	-30.00	open	sales
Lueneburg	50,000.00	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	11,357.50	11,357.50	11,357.50	11,357.50	Ready for approval	
Bags New York	-39,857.50	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	1,857.50	1,857.50	-50,000.00	1,857.50	open	
New Yorker	20,000.00	0.00	0.00	400.00	500.00	500.00	5,005.00	500.00	500.00	3,148.75	3,148.75	3,148.75	3,148.75	Ready for approval	
Nightblue	300.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	-1,876.25	-1,876.25	-1,876.25	-1,876.25	open	
Gilbert	5,000.00	0.00	0.00	1,200.00	500.00	500.00	5,005.00	500.00	500.00	-801.25	-801.25	-801.25	-801.25	open	
Luxor	8,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	48.75	48.75	48.75	48.75	Ready for approval	
Madox	10,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	548.75	548.75	548.75	548.75	open	
	56,442.50	0.00	0.00	4,600.00	3,400.00	11,400.00	25,925.00	2,860.00	3,100.00	14,253.75	14,253.75	-37,603.75	14,253.75		

2 JavaScript Selektor ID

For unique assignment, each widget has a JavaScript selector ID.

The JavaScript selector ID can be found in the widget settings under Widget Type:



To copy the ID, you can use the button to the right of the ID.

2.1 Structure of the ID

The selector ID looks like this:
.jsID_E_0_0

The first part of the ID is the abbreviation for JavaScript ID
.jsID_E_0_0

The second part of the ID describes the area in which the widget is used
.jsID_E_0_0

E stands for edit area and C stands for calculation area.

The two digits stand for the respective column and number in which the widget is used
.jsID_E_0_0

The counter starts at zero and is always incremented by 1.
The first digit identifies the column in which the widget is located and is used in Single Business Cases. In Table Business Cases, the counter is always 0, as no columns are used.
The second digit represents the consecutive numbering of the widgets.
If you change the order of the widgets, the ID also changes.

3 Business logic in the web browser

Own JavaScript business logic can be executed **automatically** when the user is in the

- **Single Business Case or**
- **Table Business Case**

in insert or editing mode:

- check and uncheck a **checkbox**
- exits an **input field** (or presses the Enter key)
- selects a value in a **lookup widget (for all tables)** without an input option (i.e. user can only select values but not limit the selection of values)

After that, a JavaScript routine can be automatically executed in the browser to change other widget values:

- Widget Label
- Widget Label with variables
- Input field widget

Attention: Only the widgets of the **current** data row can be changed, as well as all calculation widgets.

Example:

1. The user sets a checkbox or changes a number in an input field and exits this input field.
2. The self-defined JavaScript routine is started

The routine can now read values from other widgets and change the widget value of the type label, label with variables or input field without a submit.

In a table business case, the current values of a column can also be summed up and output in a calculation widget, for example.

3.1 Limitations

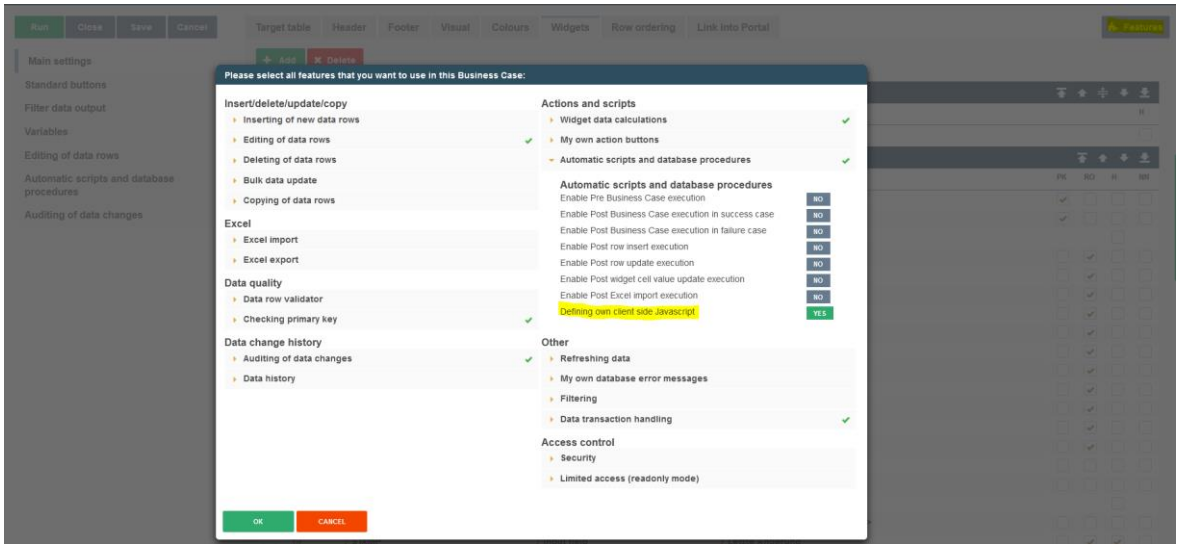
The execution in the web browser naturally results in some restrictions:

- Variables can only be used to a limited extent, they are calculated on the server side.
- You could start an ActionBC with window.open, but the (intermediate) results of the calculations cannot be saved in this way, as access to the widget references of the calculations is missing.

4 Use in a Table Business Case

4.1 Activating the feature

Activate the feature " Defining own client side JavaScript" under Features in the widget settings:



4.2 Available JavaScript methods

The general format for **get** and **set** methods:
getAfeWidgetValue(targetElementSelector)

must be extended in the table business case in the edit area as follows:
getAfeTableWidgetValue(sourceElement, targetElementSelector).

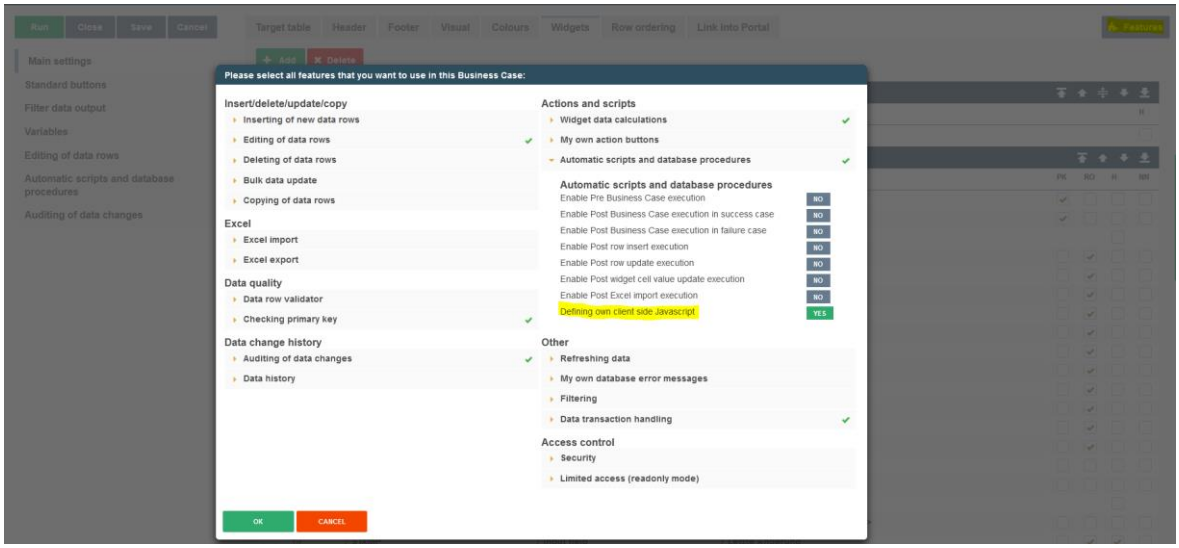
Table is a hint for the routine and sourceElement refers to the current row and **is always** 'this'.

Goal	Command
Read the value of a widget of the current row, Value is a **number**	getAfeTableWidgetNumValue(this, '.jsID_E_0_1');
Read the value of a widget of the current row, Value is a **string**	getAfeTableWidgetStringValue(this, '.jsID_E_0_1');
Write to a widget of the current row, Value is a **number**	setAfeTableWidgetNumValue(this,'.jsID_E_0_2', calcValueNum);
Write to a widget of the current row, Value is a **string**	setAfeTableWidgetStringValue(this,'.jsID_E_0_2', calcValueNum);
Read the value of a **calculation widget**, Value is a **number**	getAfeWidgetNumValue('.jsID_E_0_1');
Write a value to a calculation widget, Value is a **string**	setAfeWidgetStringValue('.jsID_C_0_1', 'CHANGED');
Reading a Lookup Widget Label	var myLabelValue = getAfeTableWidgetLookupLabel(this, '.jsID_E_0_3');
Read/aggregate all numeric values of a column of the current page	getAfeTableColumnFunction('.jsID_E_0_9', 'sum') ; getAfeTableColumnFunction('.jsID_E_0_9', 'min') ; getAfeTableColumnFunction('.jsID_E_0_9', 'max') ; getAfeTableColumnFunction('.jsID_E_0_9', 'avg') ; Null values are calculated with 0-values

5 Use in a Single Business Case

5.1 Activating the Feature

Activate the feature "Defining own client side JavaScript" under Features in the widget settings:



5.2 Available JavaScript methods

Goal	Command
Read the value of a widget, Value is a **number**	getAfeWidgetNumValue('.jsID_E_0_1');
Read the value of a widget, Value is a s**tring**	getAfeWidgetStringValue('.jsID_E_0_1');
Write to a widget, Value is a **number**	setAfeWidgetNumValue('.jsID_E_0_2', calcValueNum);
Write to a widget, Value is a **string**	setAfeWidgetStringValue('.jsID_E_0_2', calcValueNum);
Read the value of a **calculation widget**, Value is a **number**	getAfeWidgetNumValue('.jsID_E_0_1');
Write a value to a calculation widget, Value is a **string**	setAfeWidgetStringValue('.jsID_C_0_1', 'CHANGED');
Reading a Lookup Widget Label	var myLabelValue = getAfeWidgetLookupLabel('.jsID_E_0_3');

6 Read/Write Widget Values

6.1 Reading widget values

With the method ***getAfeWidgetValue(JavaScriptSelektorID)*** you can read any widget value.
The *JavaScriptSelektorID* identifies the widget whose value we want to read.

For numeric values (numbers) use *getAfeWidgetNumValue* and
For string values use *getAfeWidgetStringValue*.

6.2 Writing widget values

With the method ***setAfeWidgetValue(JavaScriptSelektorID, value)*** you can write values into widgets.
The *JavaScriptSelektorID* identifies the widget we want to write to and **value** identifies the value (e.g. a number) to write.

For numeric values (numbers) use *setAfeWidgetNumValue* and
For string values (strings) use *setAfeWidgetStringValue*.

6.3 Example function

In this example, the value of the widget with reference ID **.jsID_E_0_0** is read and
the value of the widget * 2 is stored back into the widget **.jsID_E_1_2** when the user exits the widget **.jsID_E_0_0**.

```
$(document).on('change', '.jsID_E_0_0', function(){
  var myValue = getAfeWidgetNumValue('.jsID_E_0_0');
  setAfeWidgetNumValue('.jsID_E_1_2', myValue * 2);
})
```

6.3.1 In detail

$(document).on('change', '.jsID_E_0_0', function()

Starts a JavaScript ***function()*** when a value in the widget ***'.jsID_E_0_0'*** in the web browser ***$(document)*** is changed ***on('change').***

The content of the JavaScript function is enclosed in the curly brackets.

{ var myValue = getAfeWidgetNumValue('.jsID_E_0_0');

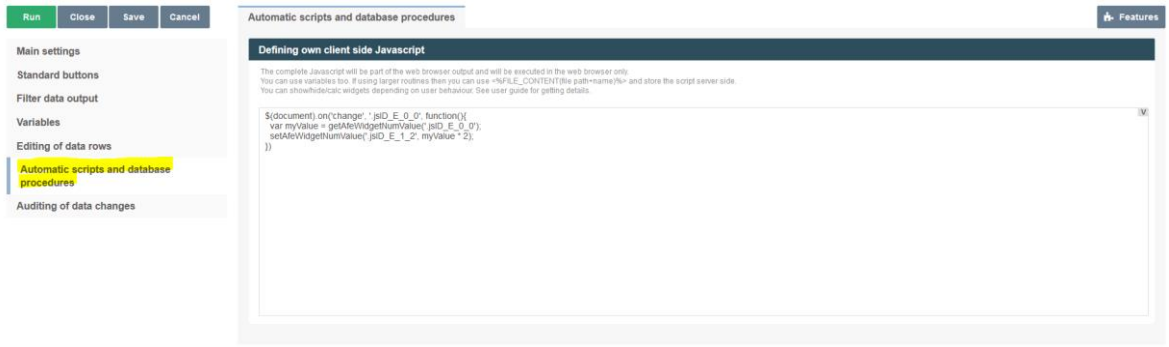
Defines ***var*** and fills the JavaScript variable ***myValue*** with the value of the widget ***'.jsID_E_0_0'***.

setAfeWidgetNumValue('.jsID_E_1_2', myValue * 2); }

Writes the content of the JavaScript variable ***myValue*** multiplied by 2 **** 2*** into the widget ***'.jsID_E_1_2'*** as numeric value ***setAfeWidgetNumValue***

Note
Numeric value is important here, the system can thus automatically use language-specific number formats, otherwise there would be problems e.g. when using different decimal separators (123.45 and 123,45)

6.3.2 Use in Apparo



6.4 Possibilities of a Checkbox

Checkboxes can be set independently of the used value (usually 0 and 1 or Y and N).
With true the checkbox is checked, with false the checkbox is unchecked.

setAfeWidgetStringValue('.jsID_E_0_3', true);

This call sets the checkmark (=true) in the checkbox widget with the reference ***'.jsID_E_0_3'***.
Since true is a string we use the method ***setAfeWidgetStringValue***.

Checkboxes can also be easily hidden.

In the following example, the checkbox is hidden depending on its value:

Example:

```
$(document).on('change', '.jsID_E_0_3', function(){
  var myValue = getAfeWidgetNumValue('.jsID_E_0_3');
  if(true == myValue) {


document.querySelector('.jsID_E_0_4').parentElement.parentElement.parentElement.parentElement.styl
e.display = "none";

  }
  else {

document.querySelector('.jsID_E_0_4').parentElement.parentElement.parentElement.parentElement.styl
e.display = "table-row";
  }
 });
```

If the checked checkbox is set, the checkbox widget ***.jsID_E_0_4*** is **hidden** or otherwise (re)**displayed**.

6.5 Possibilities of Lookup Widgets

Lookup widgets can only be read, but not set.

6.5.1 Lookup key values

Lookup key values can be read like this:

In the table business case

getAfeTableWidgetNumValue(sourceElement, targetElementSelector)

var myLabelValue = getAfeTableWidgetNumValue(this, '.jsID_E_0_4');

In the single business case

getAfeWidgetNumValue(targetElement)

var myLabelValue = getAfeWidgetNumValue('.jsID_E_0_4');

6.5.2 Lookup output values (Label)

The lookup output values can be read like this:

In the table business case

getAfeTableWidgetLookupLabel(sourceElement, targetElementSelector)

var myLabelValue = getAfeTableWidgetLookupLabel(this, '.jsID_E_0_4');

In the single business case

getAfeWidgetLookupLabel(targetElement)

var myLabelValue = getAfeWidgetLookupLabel('.jsID_E_0_4');

6.6 Aggregate all values of a column in a table Business Case

It is possible to perform calculations over all used rows of a widget (=column).
The column must be numeric.
All values of only the current page (=all visible data rows) are taken into account.
The sum can be output e.g. in a calculation widget.

Product	Sum year	January	February	March	April	May	June	July	August	September	Oktober	November	December	Workflow
T-Shirt Vienna	3,000.00	0.00	0.00	200.00	300.00	2,000.00	300.00	120.00	200.00	-30.00	-30.00	-30.00	-30.00	open
Lueneburg	50,000.00	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	11,357.50	11,357.50	11,357.50	11,357.50	Ready for approval
Bags New York	-39,857.50	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	1,857.50	1,857.50	-50,000.00	1,857.50	open
New Yorker	20,000.00	0.00	0.00	400.00	500.00	500.00	5,005.00	500.00	500.00	3,148.75	3,148.75	3,148.75	3,148.75	Ready for approval
Nightblue	300.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	-1,876.25	-1,876.25	-1,876.25	-1,876.25	open
Gilbert	5,000.00	0.00	0.00	1,200.00	500.00	500.00	5,005.00	500.00	500.00	-801.25	-801.25	-801.25	-801.25	open
Luxor	8,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	48.75	48.75	48.75	48.75	Ready for approval
Madox	10,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	548.75	548.75	548.75	548.75	open
	56,442.50	0.00	0.00	4,600.00	3,400.00	11,400.00	25,925.00	2,860.00	3,100.00	14,253.75	14,253.75	-37,603.75	14,253.75	

getAfeTableColumnFunction(targetColumnSelector, functionName)

targetColumnSelector refers to the widget for which the function is to be calculated over all rows.

functionName refers to the function and can be one of the following:

- **Sum**: getAfeTableColumnFunction('.jsID_E_0_3', 'sum')
- **Minimum**: getAfeTableColumnFunction('.jsID_E_0_3', 'min')
- **Maximum**: getAfeTableColumnFunction('.jsID_E_0_3', 'max')
- **Average**: getAfeTableColumnFunction('.jsID_E_0_3', 'avg')

6.6.1 Example for sum over one column

For this we need a calculation widget (without content) of type Label with variables for the output.
The output value is calculated and entered by the script.

var Write_Sum;

With this we define the JavaScript variable ***Write_Sum***

Write_Sum = getAfeTableColumnFunction('.jsID_E_0_3', 'sum');

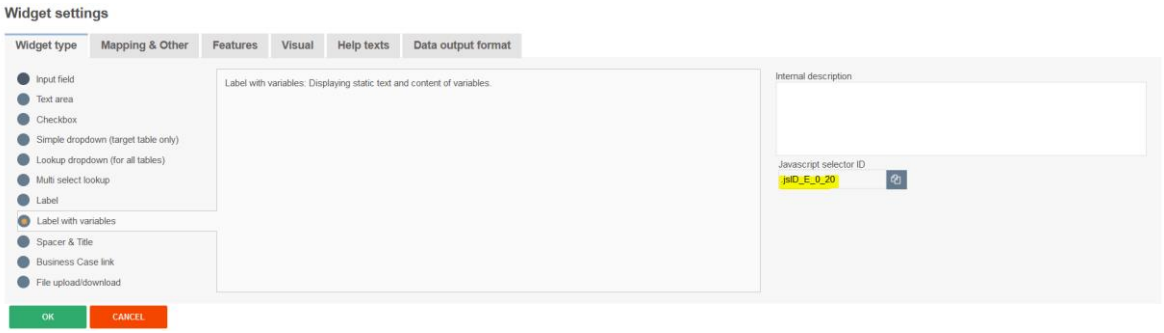
Herewith we fill the variable ***Write_Sum*** with the calculation of the sum ***getAfeTableColumnFunction('.jsID_E_0_3', 'sum')*** over all visible rows of the widget ***'.jsID_E_0_3'***.

setAfeWidgetNumValue('.jsID_C_0_0', Write_Sum);

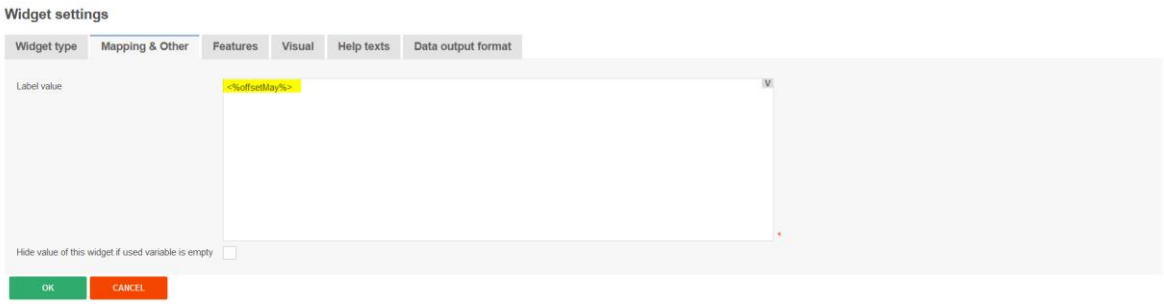
Here we write ***setAfeWidgetNumValue('.jsID_C_0_0', Write_Sum);*** the calculation stored in the variable ***Write_Sum*** to the calculation widget ***'.jsID_C_0_0'***.

6.7 Use of variables

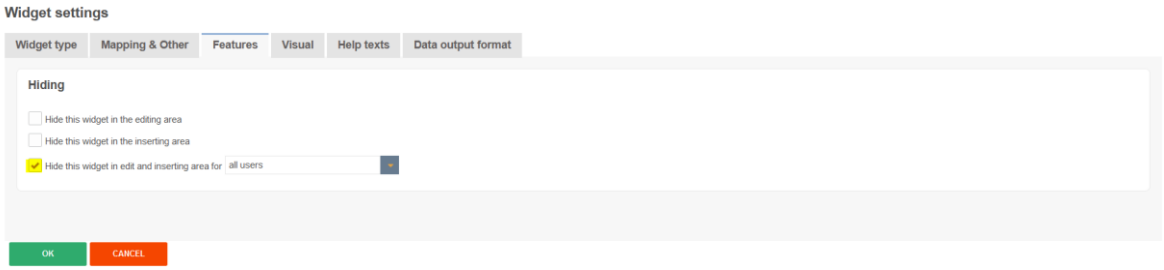
As already mentioned in chapter 3.1, the use of variables is only possible to a limited extent.
If these are to be included in calculations, they must first be output in a widget of the type label with variables and then read from there with the get method.



The JavaScript selector ID of the widget is required.



The variable is output as a label value.



Optionally, the widget can also be hidden as 'hidden'.

Example:

setAfeTableWidgetNumValue(this,'.jsID_E_0_7', calcValueNum+
getAfeTableWidgetNumValue(this,'.jsID_E_0_20')

With ***setAfe*** the value of the JavaScript variable ***calcValueNum*** is written into the widget ***'.jsID_E_0_7'***, added with the value of the Apparo variable ***<%offsetMay%>***, output in widget ***'.jsID_E_0_20'***.

Hint:

If you want to store a calculated value but the user must not see this value then just use an input widget **with column size of 0. That means it isn't visible but normal useable.**

7 Use of larger JavaScript programmes

With increasing complexity, it is advisable to outsource the JavaScript to an external file.

With the variable:

<%FILE_CONTENT(path+file)%>

you can import the contents of the file again.

Example call of the file myJsFunctions.txt:

<%FILE_CONTENT(D:\My Data\script\myJsFunctions.txt)%>.

The following replaces the variable with the contents of the file:

myAlert(); // <- Example content of the file.

8 Enter key for calling the JavaScript routine

The Enter key is normally used to simulate a click on the OK key.
However, if the user is to be able to start the calculations with the enter key, the function "ready" must be extended:

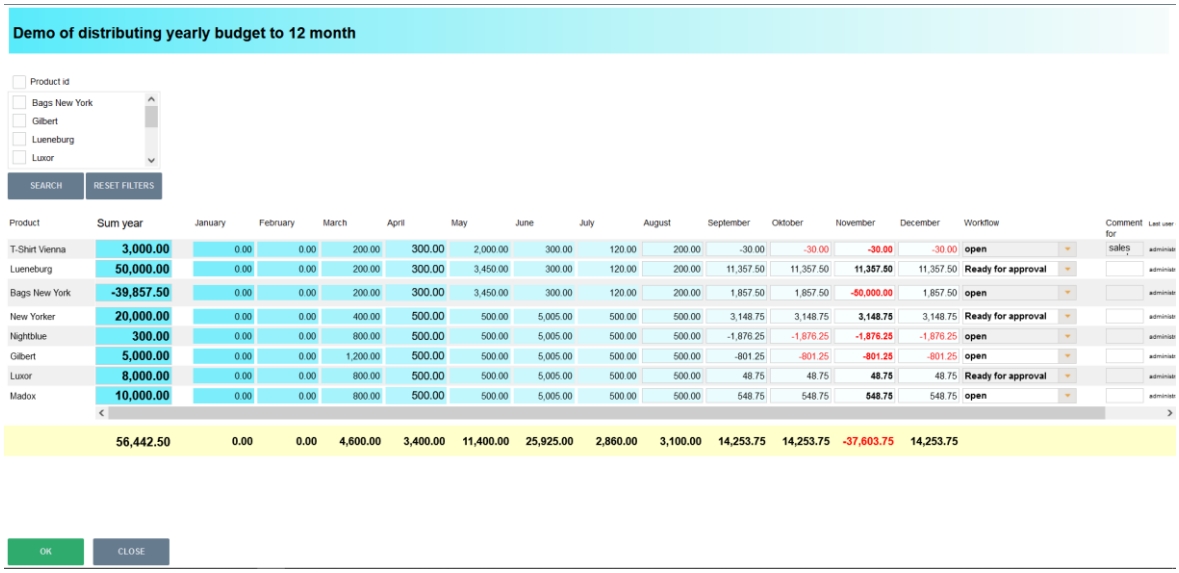
```
$(document).ready(function(){
// The business case has been started, the Calc widget is set to 0
setAfeWidgetNumValue('.jsID_C_0_0', 0);

  disableFormSubmitOnEnter();
})
```

With ***disableFormSubmitOnEnter();*** the enter key is converted and only calls the JavaScript routine.

9 Example of a Table Business Case for Planning

In this example, a small planning application is developed.

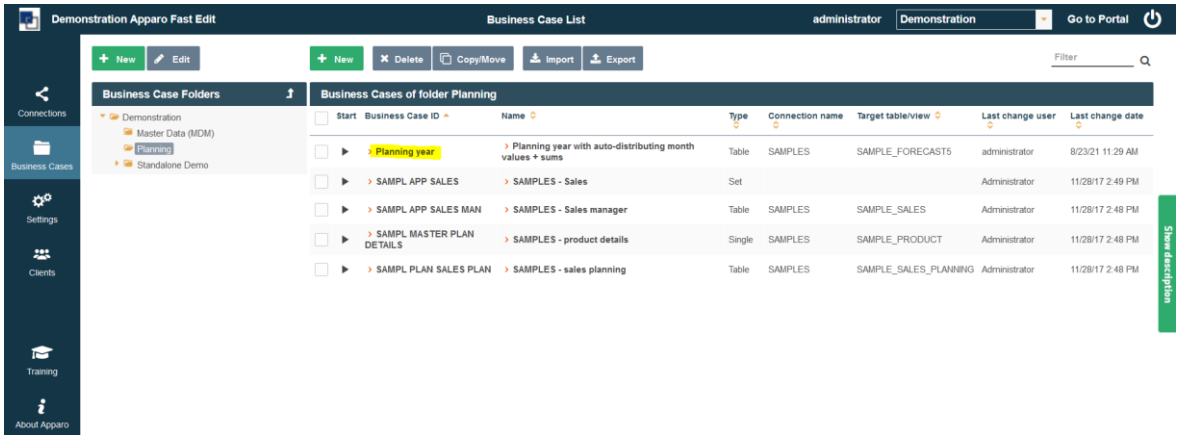
1. The entered annual total is distributed over the annual months, ignoring the previous months (these are "frozen").
2. If the user has entered a monthly plan value, the annual total is automatically updated.
3. For May, an additional value is automatically added from a hidden "Label with variables" widget.
4. The monthly totals for all selected products and the yearly total are automatically calculated.



Product	Sum year	January	February	March	April	May	June	July	August	September	Oktober	November	December	Workflow	Comment for
T-Shirt Vienna	3,000.00	0.00	0.00	200.00	300.00	2,000.00	300.00	120.00	200.00	-30.00	-30.00	-30.00	-30.00	open	sales
Lueneburg	50,000.00	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	11,357.50	11,357.50	11,357.50	11,357.50	Ready for approval	
Bags New York	-39,857.50	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	1,857.50	1,857.50	-50,000.00	1,857.50	open	
New Yorker	20,000.00	0.00	0.00	400.00	500.00	500.00	5,005.00	500.00	500.00	3,148.75	3,148.75	3,148.75	3,148.75	Ready for approval	
Nightblue	300.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	-1,876.25	-1,876.25	-1,876.25	-1,876.25	open	
Gilbert	5,000.00	0.00	0.00	1,200.00	500.00	500.00	5,005.00	500.00	500.00	-801.25	-801.25	-801.25	-801.25	open	
Luxor	8,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	48.75	48.75	48.75	48.75	Ready for approval	
Madox	10,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	548.75	548.75	548.75	548.75	open	
	56,442.50	0.00	0.00	4,600.00	3,400.00	11,400.00	25,925.00	2,860.00	3,100.00	14,253.75	14,253.75	-37,603.75	14,253.75		

You can find this example business case in the public demo

https://demo.apparo.services



Hint:

In the training menu of the Apparo Designer, you can find the making of videos for this Business Case.

The Script:

```
// sum of a product was changed:
$(document).on('change', '.jsID_E_0_1', function(){

// get current value of the sum widget:
 var myValue = getAfeTableWidgetNumValue(this, '.jsID_E_0_1');

// Date calculations...
 var currentDate = new Date();
 var currentMonth = currentDate.getMonth() +2;
 var months = 13 - currentMonth;

 //calculate sum of values in the past
 var sumOfPast = 0;
 if (currentMonth >= 2 ) {  sumOfPast =                  getAfeTableWidgetNumValue(this,'.jsID_E_0_3'); };
 if (currentMonth >= 3 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_4');  };
 if (currentMonth >= 4 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_5');  };
 if (currentMonth >= 5 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_6');  };
 if (currentMonth >= 6 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_7');  };
 if (currentMonth >= 7 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_8');  };
 if (currentMonth >= 8 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_9');  };
 if (currentMonth >= 9 ) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_10');  };
 if (currentMonth >= 10) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_11'); };
 if (currentMonth >= 11) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_12'); };
 if (currentMonth == 12) {  sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this,'.jsID_E_0_13'); };


// change values of future months and current only
// toFixed(2) rounds the value to 2 decimal places, but the result is string. The '+' in the begin convert this string into English number
var calcValueNum = + (((myValue-sumOfPast) / months).toFixed(2));

 if (currentMonth == 1 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_3', calcValueNum); }
 if (currentMonth <= 2 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_4', calcValueNum); }
 if (currentMonth <= 3 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_5', calcValueNum); }
 if (currentMonth <= 4 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_6', calcValueNum); }

// for May additional the calculated value of E_0_17 must be added.
// this is an example how to use calculated values from the database, e.g. SQL. The value is stored in a hidden widget of type "Label with variables"
 if (currentMonth <= 5 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_7', calcValueNum+ getAfeTableWidgetNumValue(this,'.jsID_E_0_17')  ); }

 if (currentMonth <= 6 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_8', calcValueNum); }
 if (currentMonth <= 7 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_9', calcValueNum); }
 if (currentMonth <= 8 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_10', calcValueNum); }
 if (currentMonth <= 9 ) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_11', calcValueNum); }
 if (currentMonth <= 10) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_12', calcValueNum); }
 if (currentMonth <= 11) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_13', calcValueNum); }
 if (currentMonth <= 12) {  setAfeTableWidgetNumValue(this,'.jsID_E_0_14', calcValueNum); }

  // because the particular values may be rounded, recalculate the SUM in order to reflect the sum of rounded values
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_3', function(){
 // january value was changed
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_4', function(){
 // February value was changed
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_5', function(){
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_6', function(){
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_7', function(){
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_8', function(){
 calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_9', function(){
```

```
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_10', function(){
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_11', function(){
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_12', function(){
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_13', function(){
  calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_14', function(){
  calculateYearSum(this);
})

function calculateYearSum(elem) {
// make a sum of all months of the current product
  var yearSum =  getAfeTableWidgetNumValue(elem,'.jsID_E_0_3');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_4');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_5');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_6');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_7');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_8');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_9');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_10');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_11');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_12');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_13');
   yearSum = yearSum + getAfeTableWidgetNumValue(elem,'.jsID_E_0_14');

// recalculate all month sums of all products
   calculateColumnSums();

// set sum of year of current product
   setAfeTableWidgetNumValue(elem, '.jsID_E_0_1', yearSum);


}

function calculateColumnSums() {

// recalc all month sums

var m;
m = getAfeTableColumnFunction('.jsID_E_0_3', 'sum');
setAfeWidgetNumValue('.jsID_C_0_3', m);

m = getAfeTableColumnFunction('.jsID_E_0_4', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_4', m);

m = getAfeTableColumnFunction('.jsID_E_0_5', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_5', m);

m = getAfeTableColumnFunction('.jsID_E_0_6', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_6', m);

m = getAfeTableColumnFunction('.jsID_E_0_7', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_7', m);

m = getAfeTableColumnFunction('.jsID_E_0_8', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_8', m);

m = getAfeTableColumnFunction('.jsID_E_0_9', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_9', m);

m = getAfeTableColumnFunction('.jsID_E_0_10', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_10', m);

m = getAfeTableColumnFunction('.jsID_E_0_11', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_11', m);

m = getAfeTableColumnFunction('.jsID_E_0_12', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_12', m);
```

```
m = getAfeTableColumnFunction('.jsID_E_0_13', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_13', m);

m = getAfeTableColumnFunction('.jsID_E_0_14', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_14', m);

// calc total sum and display it, value is the sum of all products
setAfeWidgetNumValue( '.jsID_C_0_1',
  getAfeTableColumnFunction('.jsID_E_0_3', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_4', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_5', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_6', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_7', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_8', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_9', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_10', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_11', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_12', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_13', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_14', 'sum')  );
}



$(document).ready(function(){
// Business Case was started, this function will be called automatically, the calc widget are updated

// pressing enter key means new event and not making submit
  disableFormSubmitOnEnter();

// calc month sums:
  calculateColumnSums();

})


function onAfeFormReload() {
 $(document).ready(function(){
// Business Case after submit (e.g. pressing OK button) is calling this function automatically

// enter key means new event
  disableFormSubmitOnEnter();

// calc month sums:
  calculateColumnSums();
 })
}




$(document).on('focus', '.jsID_E_0_1', function(){

// the user has clicked into the sum widget. Now this function is called automatically.
// This is helpful if you want to make calculations directly after user clicked into a widget

// ... place for activities

} )
```