Column Mapping

Block Group:	Table Operations
Icon:	

The Column Mapping block returns a new table that can include specified columns from the input table and can include new columns that contain calculated values.

For information on using dataflow blocks, see Dataflow.

For answers to some common questions about working with tables, see Tables.

Input/Output Properties

The following properties of the Column Mapping block can take input and give output.

- input *(table)*
- retainColumns *(boolean)*
- name *n (string)*
- from *n (string)*

input receives the input table.

retainColumns specifies whether to include the input table's columns in the output table.

name *n* specifies the new column name.

from *n* specifies the data that will appear in the new column, using JavaScript notation. For example:

- **Simple expressions**: =v1+v2 causes this column to hold the sum of the values in the **v1** column and the **v2** column.
- **DateTime functions**: `=dateFormat(v3,"y-MM-dd")` causes this column to to reformat and store the value from the **v3** column.
- **Number functions**: `=numberFormat(v4,"0.00")` causes this column to store the value in the **v4** column, formatted with the specified string.
- **Conditions**: `=(v5 > 1 ? true : false)` causes this column to check values in the **v5** column. This example returns a TRUE boolean value if the **v5** value is greater than one and a FALSE boolean value otherwise.
- **Nested conditions**: `=(v6 <= 1 ? false : (v6 > 2 ? false : true))` causes this column to check values in the **v6** column. If the **v6** value is less than or equal to one, this example returns a FALSE boolean value. Otherwise, this example returns the result of the nested condition.
- **Custom math functions**: `=(function($row) {return Math.round(Math.random() * $row * 100);})(v7)` causes this column to perform the math function contained in this **from *n*** property and return the result. In this example, the result is the rounded product of a random

number, the value in **v7** for this row, and 100.
- **Custom string functions**: `=(function($row) {return $row.length;})(v8)` causes this column to perform the string function contained in this **from *n*** property and return the result. In this example, the result is the length of the string in **v8** for this row.

Output Properties

The following properties of the Column Mapping block can give output but cannot take input.

- output *(table)*
- print *(string)*

output returns the output table with new columns.

print displays error messages and other notifications for debugging your dataflow.

Storing Temporary Values

You can use `$.<variable>` in Column Mapping and Filter to store any temporary variable between rows.

The following example returns the value of v1 from the previous row.

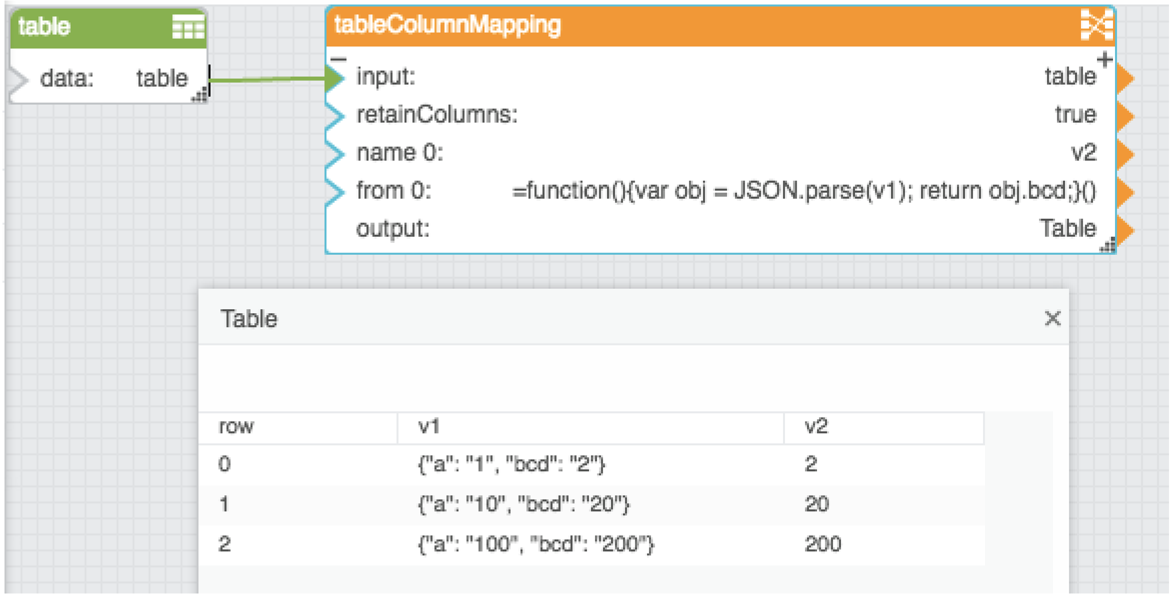
```
=function(){var prev = $.v1cache; $.v1cache = v1; return prev}()
```

The following example returns the difference between this row's v1 and the previous row's v1. The example converts both values to `millisecondsSinceEpoch`, then divides the difference by 60,000 to get minutes.

```
=function(){var diff = (dateParse(v1) - $.a)/1000/60; $.a = dateParse(v1);
return diff}()
```

Parsing a JSON Object

The following image shows an example. In this example, a column of the input table contains JSON. The output table contains the input JSON and also the value of the `bcd` item in the JSON for each row. See also: Scripting and Syntax.



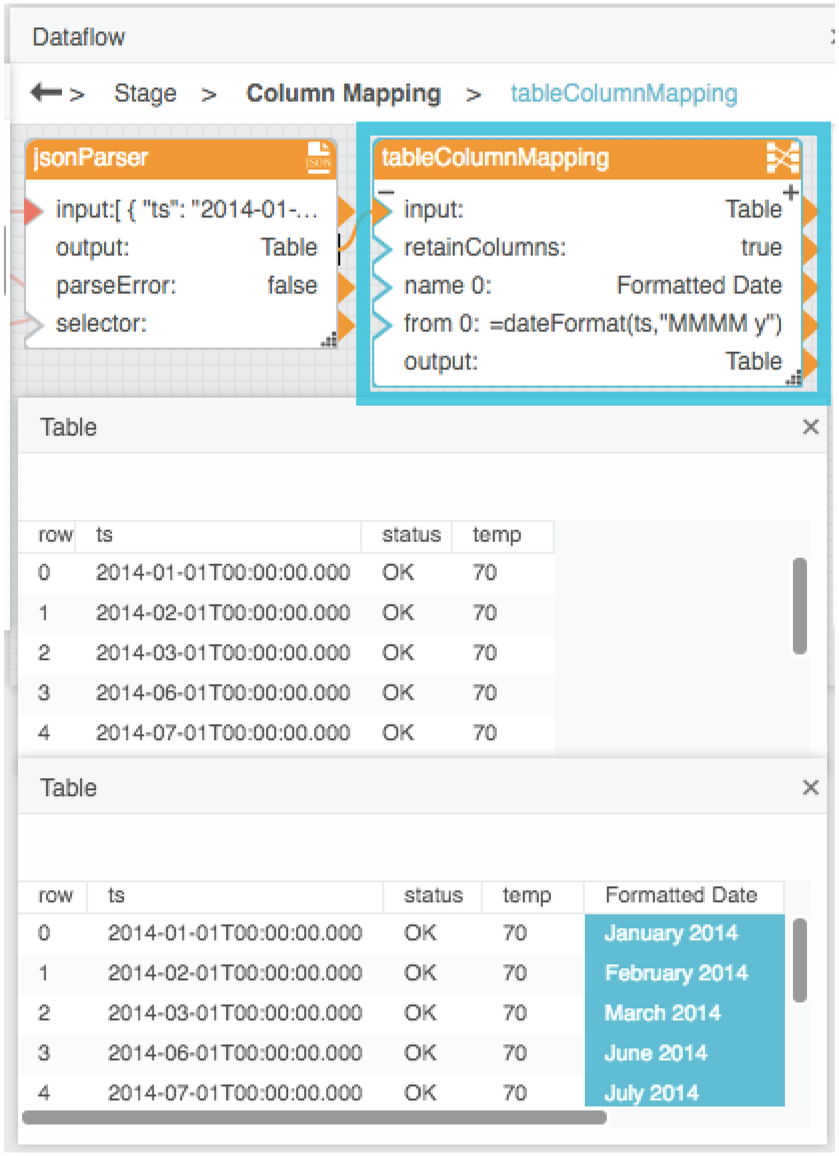
input: table
retainColumns: true
name 0: v2
from 0: =function(){var obj = JSON.parse(v1); return obj.bcd;}()
output: Table

Table

row	v1	v2
0	{"a": "1", "bcd": "2"}	2
1	{"a": "10", "bcd": "20"}	20
2	{"a": "100", "bcd": "200"}	200

Examples

The following images show examples of the Column Mapping block. In the first example, the new column displays a reformatted date string. In the second example, inches are converted to centimeters.



input:	[{ "ts": "2014-01-...
output:	Table
parseError:	false
selector:	

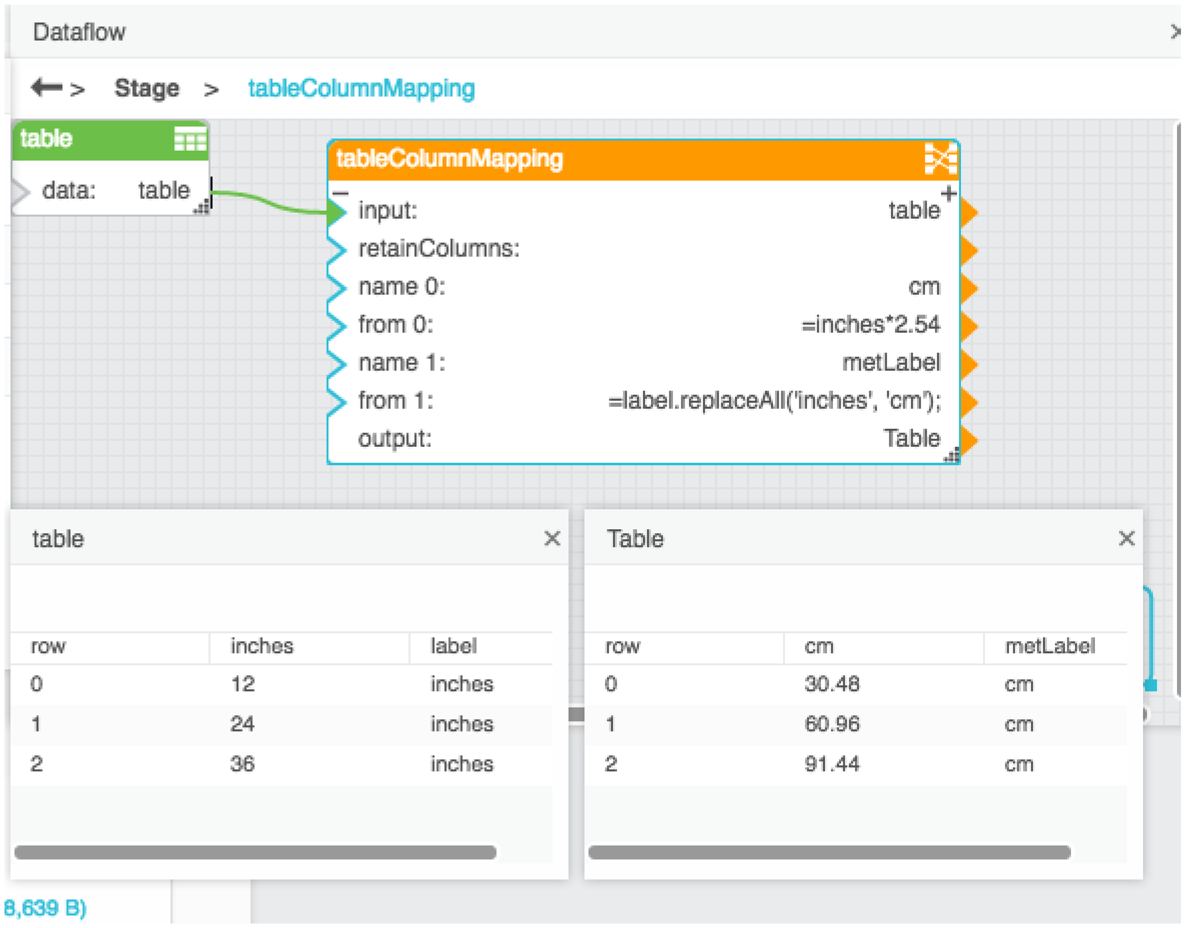
input:	Table
retainColumns:	true
name 0:	Formatted Date
from 0:	=dateFormat(ts,"MMMM y")
output:	Table

Table

row	ts	status	temp
0	2014-01-01T00:00:00.000	OK	70
1	2014-02-01T00:00:00.000	OK	70
2	2014-03-01T00:00:00.000	OK	70
3	2014-06-01T00:00:00.000	OK	70
4	2014-07-01T00:00:00.000	OK	70

Table

row	ts	status	temp	Formatted Date
0	2014-01-01T00:00:00.000	OK	70	January 2014
1	2014-02-01T00:00:00.000	OK	70	February 2014
2	2014-03-01T00:00:00.000	OK	70	March 2014
3	2014-06-01T00:00:00.000	OK	70	June 2014
4	2014-07-01T00:00:00.000	OK	70	July 2014



input: table
retainColumns:
name 0: cm
from 0: =inches*2.54
name 1: metLabel
from 1: =label.replaceAll('inches', 'cm');
output: Table

table

row	inches	label
0	12	inches
1	24	inches
2	36	inches

Table

row	cm	metLabel
0	30.48	cm
1	60.96	cm
2	91.44	cm

Use Cases

These threads in the DGLogik Community Forum show some use cases for the block:

- Calculating total of other columns
- Formatting date and time
- Rounding values
- Calculating using values from two tables (also uses Join block)
- Using dateFormat to create comparison charts

Previous: JSON Parser

Next: Sort