Level-Up your Power Query game with Custom Functions

„Make your data shine!“

2023

SPEAKER

ŠTĚPÁN REŠL

LINKEDIN

TWITTER

Data Brothers

MVP Microsoft® Most Valuable Professional

JAK NA POWER BI

Power BI kafíčko

Data Meerkat

Functions

Basics of functions

We use them all the time and for everything. Only because of them does Power Query work.

Functions

Text.From()

Table.SelectRows()

Date.Year()

...

PRE-DEFINED

EASY TO USE

WELL DOCUMENTED

PREPARED ERROR HANDLING

So what's different?

Custom functions are not so straightforward but very useful.

Custom Functions

- DEFINED BY YOU AND OTHERS
- CAN BE MORE COMPLICATED
- MOSTLY UNDOCUMENTED
- NOT PART OF YOUR NEXT SOLUTION

Why to use Custom functions?

or… where they can help?

In short, so many complexities have to pay off somehow.

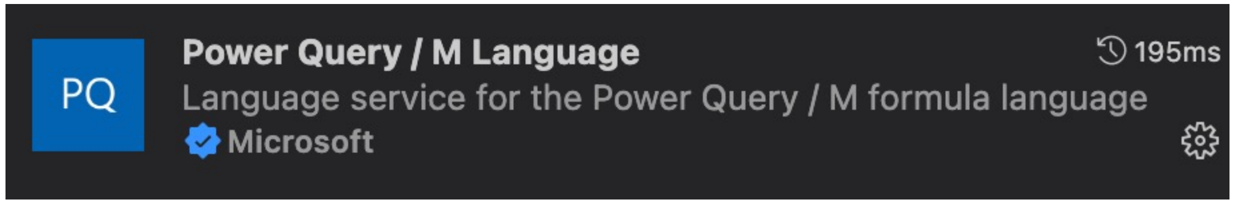
REUSABILITY

READABILITY

SCOPED

REUSABILITY

You can use them whenever you need them. You only need to create once!



Power Query / M Language

195ms

Language service for the Power Query / M formula language

Microsoft

READABILITY

In short, so many complexities have to pay off somehow.

```
= Table.AddColumn(
        filteredRowsOnlyToActive,
        "DurationInYears",
        Number.Round( List.Sum(
                List.Generate(
                        ()=> [initDate = [Date], counter = 1],
                        each _[initDate] <> today,
                        each [initDate = Date.AddMonths([initDate],1), counter = 1],
                        each [counter]
        ) ) /12, 2 )
)
```

READABILITY

In short, so many complexities have to pay off somehow.

```
= Table.AddColumn(
        filteredRowsOnlyToActive,
        “DurationInYears”,
        #”get-DurationInYears”([Date])
)
```

SCOPED

It is only in our hands what problem the function will solve and whether it will be applicable within the entire PQ or only in one query.

SHARED

PRIVATE

Anything else that is good to know?

The marking is different

The designation of custom functions is directly subordinate to its author in the given model

Like a library

Text.From()

get-ColoredExcelCells()

Inner function

A few useful insights

Little things that can have a big impact!

The return of functions can be **anything**…

You can reference functions without “()”

fctName()() can be **valid** function

Some native functions also ask custom!!!

How to define custom functions?

Basic defining

Creating a custom function is not hard. You just need ()=> and that’s it!

```
(<parameters>) => <function-body>
```

```
(x, y) => x * y
```

Type supporting

Is great to support users in understanding and preventing the usage of wrong data types!

```
(<parameters> as <type-definition>)
<return-data-type> =>
<function-body>
```

```
(x as number, y as number) as number => x * y
```

Optional parameters

The designation of custom functions is directly subordinate to its author in the given model

```
([<required-parameters>],[<optional-parameters>])
<return-data-type> =>
<function-body>
```

```
(x as number, y as number, optional z as number)
as number =>
        x * y * ( if z <> null then z else 1 )
```

More complex function body

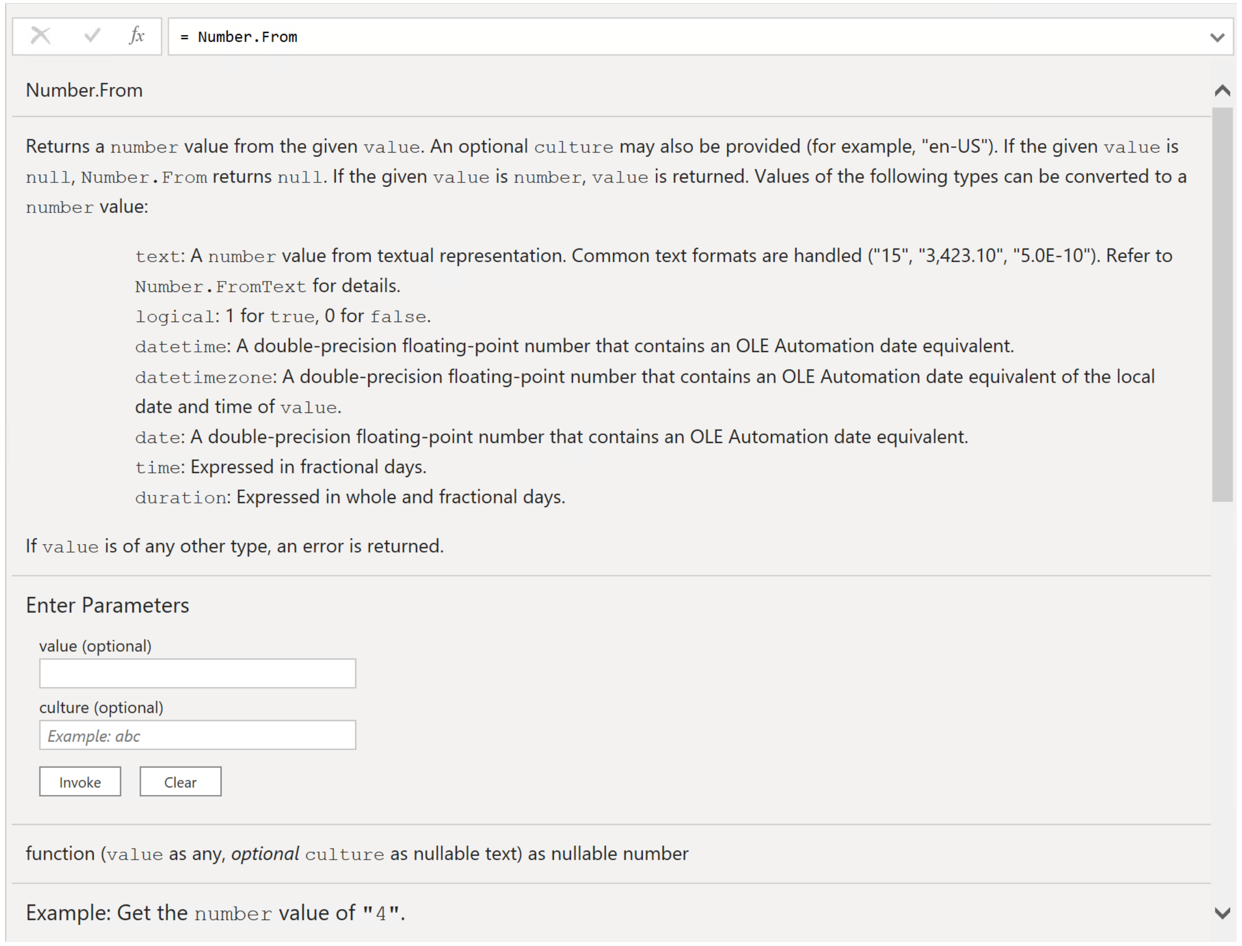
Not every time you will need just simple expression. More often, you will also need [let .. in]

```
(x as number, y as number, optional z as number)
as number =>
    let
        additional = ( if z <> null then z else 1 ),
        vl = x * y * additional
    in
        vl
```

Documentation of functions

Documentation

Documentation is essential to understand our features even a month from now.



```
= Number.From
```

Number.From

Returns a `number` value from the given `value`. An optional `culture` may also be provided (for example, "en-US"). If the given `value` is `null`, `Number.From` returns `null`. If the given `value` is `number`, `value` is returned. Values of the following types can be converted to a `number` value:

- `text`: A `number` value from textual representation. Common text formats are handled ("15", "3,423.10", "5.0E-10"). Refer to `Number.FromText` for details.
- `logical`: 1 for `true`, 0 for `false`.
- `datetime`: A double-precision floating-point number that contains an OLE Automation date equivalent.
- `datetimezone`: A double-precision floating-point number that contains an OLE Automation date equivalent of the local date and time of `value`.
- `date`: A double-precision floating-point number that contains an OLE Automation date equivalent.
- `time`: Expressed in fractional days.
- `duration`: Expressed in whole and fractional days.

If `value` is of any other type, an error is returned.

Enter Parameters

value (optional)

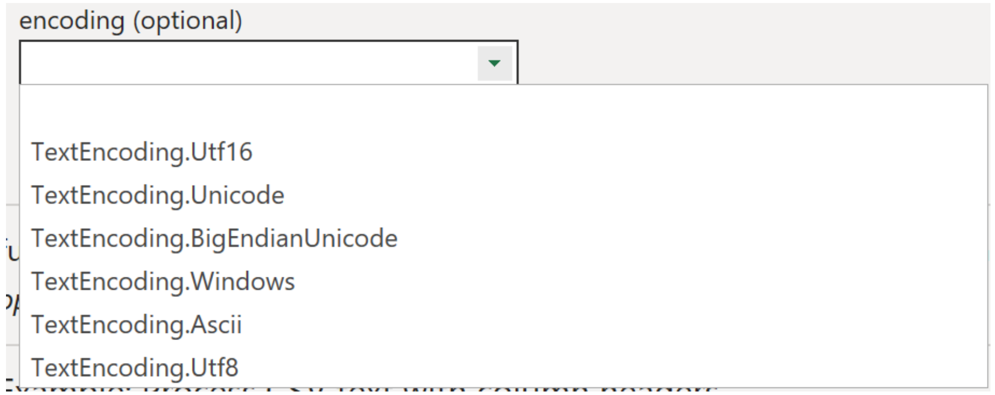
culture (optional)

Example: abc

Invoke Clear

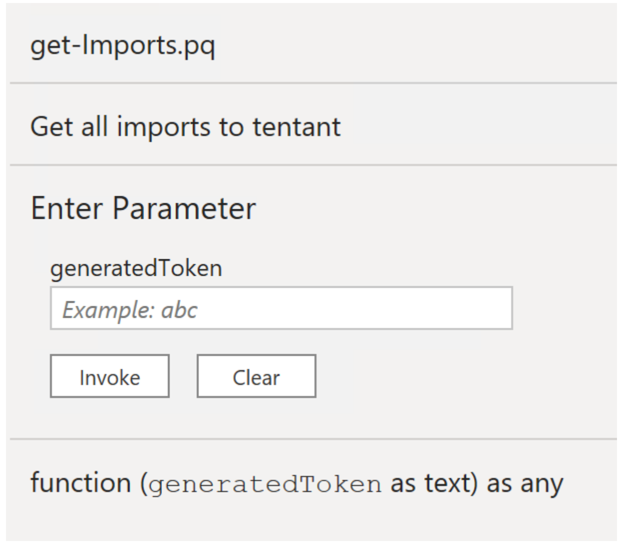
function (`value` as any, *optional* `culture` as nullable text) as nullable number

Example: Get the `number` value of `"4"`.



encoding (optional)

TextEncoding.Utf16
TextEncoding.Unicode
TextEncoding.BigEndianUnicode
TextEncoding.Windows
TextEncoding.Ascii
TextEncoding.Utf8



get-Imports.pq

Get all imports to tentant

Enter Parameter

generatedToken

Example: abc

Invoke Clear

function (`generatedToken` as text) as any

Documentation

Documentation is essential to understand our features even a month from now.

```
// --------------------------- Function ------------------------------
let
    // --------------------------- Fucntion segment -----------------------------------
    output =
        (/*parameter as text, optional opt_parameter as text*/) /*as text*/ =>      // Input definition + Function
output type definition
            let                                                                         // Inner function steps
declaration
                initStep = "",
                lastStep = ""
            in
                lastStep,                                                               // Output from inner steps
    // --------------------------- Documentation segment ------------------------------
    documentation = [
        Documentation.Name = " NAME OF FUNCTION ",                                      // Name of the function
        Documentation.Description = " DESCRIPTION ",                                    // Decription of the function
        Documentation.Source = " URL / SOURCE DESCRIPTION ",                            // Source of the function
        Documentation.Version = " VERSION ",                                            // Version of the function
        Documentation.Author = " AUTHOR ",                                              // Author of the function
        Documentation.Examples =                                                        // Examples of the functions
        {
            [
                Description = " EXAMPLE DESCRIPTION ",                                  // Description of the example
                Code = " EXAMPLE CODE ",                                                // Code of the example
                Result = " EXAMPLE RESULT "                                             // Result of the example
            ]
        }
    ]
    // --------------------------- Output --------------------------------------------
in
    Value.ReplaceType(                                                                  // Replace type of the value
        output,                                                                         // Function caller
        Value.ReplaceMetadata(                                                          // Replace metadata of the
function
            Value.Type(output),                                                         // Return output type of
function
            documentation                                                               // Documentation assigment
        )
    )
// ------------------------------------------------------------------------------------
```

Link for a Template

A reusable template can help you to have it always ready!

THANK YOU FOR THE ATTENTION

ŠTĚPÁN REŠL

LINKEDIN

TWITTER

Data Brothers

Microsoft® Most Valuable Professional

JAK NA POWER BI

Power BI kafíčko

Data Meerkat

PDF VERSION OF THIS PPT