entrinsicon

JavaScript and Saved Functions

Massaging Your Data

Cameron Blashka

Director, Informer Support

1. JavaScript Introduction
2. JavaScript - Strings
3. JavaScript - Arrays
4. JavaScript - Dates
5. Saved Functions

JavaScript Introduction

entrinsicon

Calculated Field Flow Steps

- Written in JavaScript
- Creates a new column in a query
- No return statement needed

Common JavaScript Types:

- String ("Hello World")
- Number (123)
- Date (09-01-2023)
- Array ([2, 3, 4, 0])
- Boolean (true, false)

Embedded Helper Libraries

- Can access them from within Calculated Fields or Power Script Flow Steps

- Lodash 3.10.1 - lots of utilities

```
var sortedArray = _.sortBy(unsortedArray);
```

- Moment.js - makes dates easier by converting them into moment data types

```
var momentDate = moment(regularDate);
```

JavaScript Introduction

entrinsicon

Variables

- Can store data for later use
- Simplifies the code with keywords
- All the data from the result set is stored in variables

```
var myNewVariable = "This is a variable";
myNewVariable;
```

If/then/else

- Conditional logic, like Excel
- If a condition is true, do something

```
var output = '';
if(someVariable > 5000){
    output = "Big Order";
}
else if(someVariable < 100){
    output = 'Small Order';
}

output;
```

JavaScript - Strings

Concatenation

Stitch multiple strings together

```
//Input: Two strings – one string says "Hello" and the other says "World"

//Output: A single string that says "Hello World"


"Hello" + " " + "World";
```

substring()

Extract a part of a string

string.substring(startPosition, endPosition)

```
//Input : A string with value "1234567890"

//Output: A new string with value "123-456-7890"


var numericString = $record['fieldAliasOne'];

var formattedString = numericString.substring(0,3) + "-" + numericString.substring(3,5) + "-" + numericString.substring(5,9);


formattedString;
```

Note: Double-check your data types

JavaScript - Strings

entrinsicon

_.padLeft / _.padRight

Pad a string to a certain number of characters

_.padLeft(string, desiredLength, padCharacter)

```
//Input : A string with value "12345"
//Output: A new string with value "0012345"

var plainString = $record['fieldAliasOne']
var paddedString = _.padLeft(plainString, 7, "0")

paddedString;
```

replace()

Replace a substring with another substring

string.replace(regex, replacement)

```
//Input : A string with value "123-456-7890"
//Output: A string with value "1234567890"

var stringWithHyphens = $record['fieldAliasOne'];
var stringWithoutHyphens =
stringWithHyphens.replace(/-/g, "");

stringWithoutHyphens;
```

JavaScript - Arrays

entrinsicon

_.max() / _.min()

Find the min/max value of an array

_.max(array)

```
//Input : An array of Dates
//Output: The most recent date

var dates = [02/22/2023, 01/12/2023, 01/13/2023,
03/24/2023, 03/25/2023, 02/27/2023];
_.max(dates);    //returns 03/25/2023
```

Note: Double-check your data types!

indexOf()

Return the index of a value in an array

array.indexOf(value)

```
//Input : An array of Dates
//Output: The index of the most recent date

var dates = [02/22/2023, 01/12/2023, 01/13/2023,
03/24/2023, 03/25/2023, 02/27/2023];
dates.indexOf(_.max(dates));    //returns 4
```

Note: It works on strings too!

JavaScript - Arrays

entrinsicon

.sum()

Sum the values in an array

_.sum(array)

```
//Input : An array of numbers
//Output: The sum of those numbers

var numbers = [4, 3, 7, 8, 5];
_.sum(numbers);    //returns 27
```

_.sortBy()

Sort an array in ascending order

_.sortBy(array)

```
//Input : An array of numbers
//Output: An array of numbers in ascending order

var numbers = [4, 3, 7, 8, 5];
_.sortBy(numbers);    //returns [3, 4, 5, 7, 8]
```

Note: Works with dates!

JavaScript - Dates

entrinsicon

moment().format()

Format a moment date using tokens

```
//Date: August 6, 2022
//Result: "08-10-2022"

var date = $record['fieldAliasOne'];
var formattedDate = moment(date).format("MM-DD-YYYY");

formattedDate;
```

moment().add() / moment().subtract()

Add/subtract values from a moment date

```
//Date: August 6, 2022
//Result: August 10, 2022

var date = $record['fieldAliasOne'];
var adjustedDate = moment(date).add(4, 'days');

adjustedDate;
```

JavaScript - Dates

entrinsicon

moment().isSameOrBefore / moment().isSameOrAfter

Compare two moment dates

```
//Date One: August 6, 2022
//Date Two: August 12, 2022
//Result: TRUE

var dateOne = moment($record['fieldAliasOne']);
var dateTwo = moment($record['fieldAliasTwo']);

dateOne.isSameOrBefore(dateTwo);
```

moment().diff()

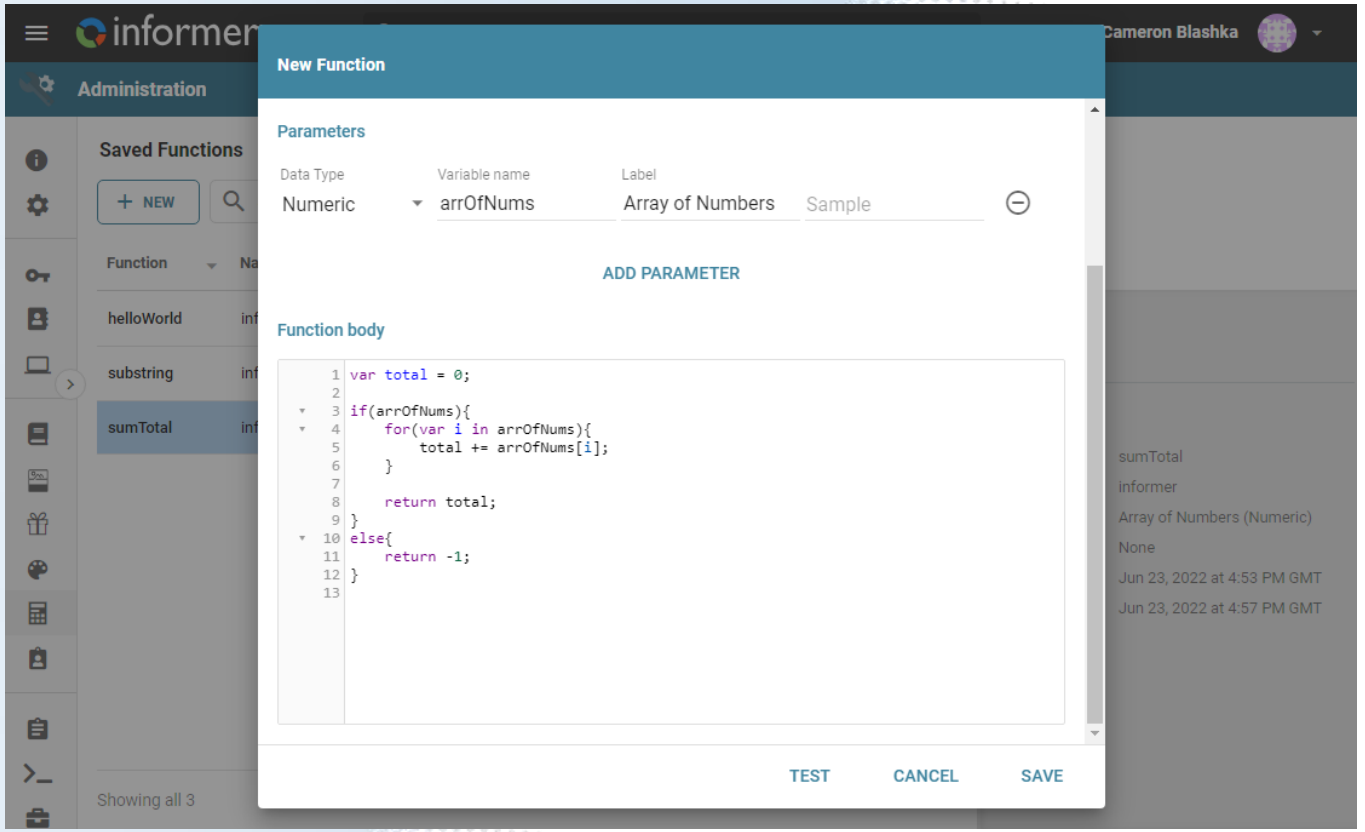
Get the time between two moment dates

```
//Date One: July 24, 2022
//Date Two: August 12, 2022
//Result: TRUE

var dateOne = moment($record['fieldAliasOne']);
var dateTwo = moment($record['fieldAliasTwo']);

dateTwo.diff(dateOne, 'days'); //returns 19
```

Note: Order matters!

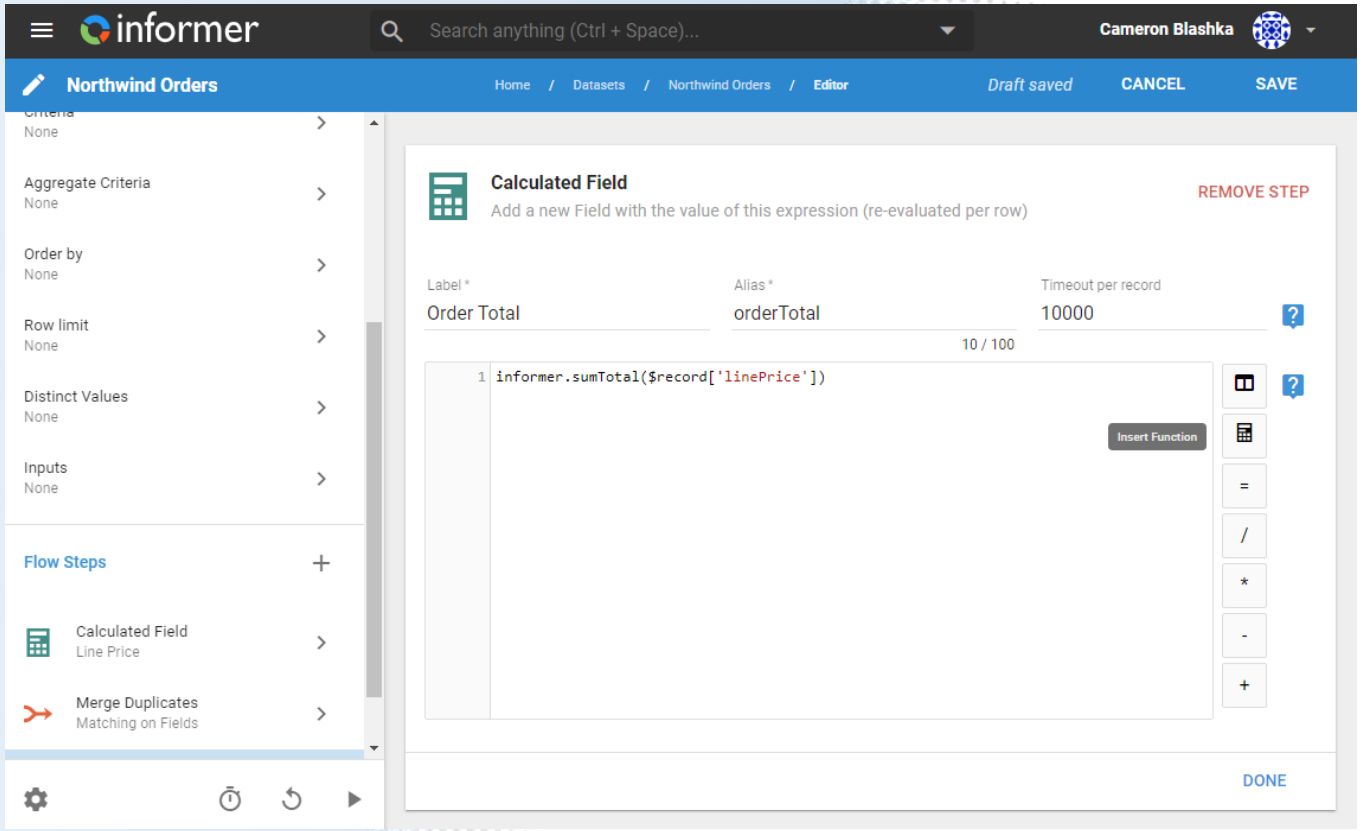
Saved Functions

- Saved Functions allow for reusable JavaScript code in Calculated Columns and Power Scripts.
- Can be updated later if made more efficient.
- Must be written as a proper JavaScript function
- Can be created and managed through the Administration page.



Saved Functions

- Add Saved Functions to Calculated Fields or Power Scripts
- "Fill in the blank" for the function's parameters
- Ease of use for end users not as familiar with JavaScript



Conclusion

- JavaScript and Calculated Fields offer many options for manipulating data
- Simple JavaScripts can accommodate advanced use cases when combined together
- Saved functions provide a flexbile solution for reusing code, helpful for beginner and advanced end users

Further Reading

- String Methods
- Array Methods
- Lodash
- Moment.js

entrinsicon

Q & A

JavaScript and Saved Functions

Cameron Blashka | cblashka@entrinsik.com