# DataWeave Tips and Tricks

 

# 1 Language Tips

## 1.1 MEL for a Dynamic DW Script

#[dw(<script>, [<output type>]) ]  example: #[dw(flowVars.script,'application/xml')]

## 1.2 Streaming Big CSV Files Using DW

Check Mulesoft Support KB.

## 1.3 Lambda Contexts

Many OOTB functions and operators use lambdas ($,$$) as unnamed parameters. Depending on the function/operator these lambdas have different meanings.

**map**:  $ = value, $$ = index

**mapObject**, pluck: $ = value, $$ = key

**reduce**: $ = value, $$ = accumulator

**match**: $ = input expression

**replace**: Not documented! -> $$ -> index of the regex match, $ -> all matches of the given expression (match string + capture groups)

## 1.4 When the output is XML

There is certain syntax that you have to adhere to when your output is **XML**:

- You should define a container element to hold the content: (e.g. `root: payload`)
- When you are mapping incoming array elements (JSON or JAVA) to XML, you will also have to wrap the **map** operation within `{(` and `)}`

## 1.5 Custom Functions

Check the Github repo. Please contact MuleSoft Professional Services to access the GitHub repo.

### 1.5.1 Remove Non-numerical Characters Function

```
%function removeNonNum(str) str splitBy "" filter ($ matches /\d/) joinBy ""
```

 

### 1.5.2 Pad String With 'n' Zeros Function

```
%function pad(str) str as :number as :string {format: "00000"}
```

## 1.6 Cheat Sheets

### 1.6.1 What DataWeave Expects in Any Operator/Function

```
== :any
=== typeOf (:any)
=== as (:any, :type)
=== + (:any, :array)
== :array
=== flatten (:array)
=== unzip (:array)
=== avg (:array)
=== min (:array)
=== max (:array)
=== sizeOf (:array)
=== sum (:array)
=== distinctBy (:array, :function)
=== zip (:array, :array)
=== map (:array, :function)
=== joinBy (:array, :string)
=== - (:array, :any)
=== filter (:array, :function)
=== ++ (:array, :array)
=== groupBy (:array, :function)
=== orderBy (:array, :function)
=== reduce (:array, :function)
=== + (:array, :any)
=== -- (:array, :array)
=== contains (:array, :any)
== :date
=== - (:date, :period)
=== - (:date, :date)
=== ++ (:date, :localtime)
=== ++ (:date, :time)
=== ++ (:date, :timezone)
=== + (:date, :period)
== :datetime
=== >> (:datetime, :timezone)
=== - (:datetime, :period)
=== - (:datetime, :datetime)
=== + (:datetime, :period)
== :localdatetime
```

 

```
=== - (:localdatetime, :period)
=== - (:localdatetime, :localdatetime)
=== ++ (:localdatetime, :timezone)
=== + (:localdatetime, :period)
== :localtime
=== - (:localtime, :period)
=== - (:localtime, :localtime)
=== ++ (:localtime, :date)
=== ++ (:localtime, :timezone)
=== + (:localtime, :period)
== :number
=== ceil (:number)
=== sqrt (:number)
=== floor (:number)
=== round (:number)
=== ordinalize (:number)
=== abs (:number)
=== * (:number, :number)
=== mod (:number, :number)
=== pow (:number, :number)
=== - (:number, :number)
=== + (:number, :number)
=== / (:number, :number)
== :object
=== sizeOf (:object)
=== - (:object, :name)
=== mapObject (:object, :function)
=== ++ (:object, :object)
=== orderBy (:object, :function)
=== -- (:object, :object)
=== pluck (:object, :function)
== :period
=== - (:period, :date)
=== - (:period, :localdatetime)
=== - (:period, :datetime)
=== - (:period, :time)
=== - (:period, :localtime)
=== + (:period, :date)
=== + (:period, :localdatetime)
=== + (:period, :datetime)
=== + (:period, :time)
=== + (:period, :localtime)
== :string
=== capitalize (:string)
=== lower (:string)
=== upper (:string)
=== singularize (:string)
=== dasherize (:string)
=== camelize (:string)
=== trim (:string)
=== sizeOf (:string)
=== pluralize (:string)
=== underscore (:string)
=== find (:string, :regex)
=== find (:string, :string)
```

 

```
=== splitBy (:string, :string)
=== splitBy (:string, :regex)
=== startsWith (:string, :string)
=== scan (:string, :regex)
=== match (:string, :regex)
=== match (:expre
=== endsWith (:string, :string)
=== ++ (:string, :string)
=== matches (:string, :regex)
=== contains (:string, :string)
=== contains (:string, :regex)
=== replace (:string, :regex, :function)
== :time
=== - (:time, :period)
=== - (:time, :time)
=== ++ (:time, :date)
=== + (:time, :period)
== :timezone
=== ++ (:timezone, :date)
=== ++ (:timezone, :localdatetime)
=== ++ (:timezone, :localtime)
```

 

# 2 DataWeave 2.0 Changes

## 2.1 Breaking Changes

```
- Removed coercion from object to array
- Updated version header to %dw 2.0
- Abbreviated and removed % from directives
  %namespace => ns
  %function => fun
- Changed type names from :object to Object, and so on
- Operators are now functions (is(), typeOf(), etc...)
- infix notation for binary functions (arg1 function arg2)
- overloaded filter, groupBy to operate on object
- Now functions are declared as:
  fun funName(args) = body
- Removed `..` operator (replaced by `to`)
- map/mapObject/filter are defined for null
- Changed pattern matching
  when/otherwise => if/else
- Namespace prefixes can't contain `-`

keywords: ["if", "else", "unless", "using", "---", "as", "is", "null", "true", "false", "default", "case",
"fun", "input", "output", "ns", "type", "import", "var", "and", "or"]
```

## 2.2 New Features

```
block comments
improved errors messages
  call values
  stacktrace
type inference (optional typing)
type checking
imports
  loaders
    java!
      MyClass::new(parameters)
array/object construct/deconstruct
  [head ~ tail] and {headKey:headValue ~ tail}
& selector (object filter)
# selector
Added index as 3rd param to mapObject, pluck, filter, groupBy
```

 