DATAWEAVE 2.0 CHEAT SHEET

DataWeave is the MuleSoft expression language for accessing and transforming data that travels through a Mule app, which runs the scripts and expressions in your Mule app

DEFINE

FUNCTION(PARAM)

```
fun toUser(obj) = {  firstName: obj.fname }
```

FUNCTION(FUNCTION PARAM)

```
fun combined(function, msg="universe") = function(msg ++ " world") --- combined: combined(toUpper, "hello")
```

```
{"combined": "HELLO WORLD"}
```

FUNCTION

```
fun test(p: String) = do { var a = "Foo" ++ p --- a }
---
{ result: test(" Bar") }
```

```
{ "result": "Foo Bar"}
```

LAMBDA

```
var msg2 = (x = "ignore") -> "hello" ---  msg2: msg2()
```

```
{"msg2": "hello"}
```

LAMBDA (PARAM)

```
var toUpper = (aString) -> upper(aString)
--- toUpper: toUpper("hello")
```

```
{"toUpper": "HELLO"}
```

LAMBDA (FUNCTION PARAM)

```
var combined = (function, msg="universe") ->
function(msg ++ " WORLD")
-- combined: combined(toUpper, "hello")
```

```
{ "combined": "HELLO WORLD" }
```

NAMESPACE

```
ns ns0 http://www.pk.com
```

```
ns0#book : "works"
```

CASE STATEMENT

```
value match {
    case (<name>:) <condition> -> <routing expression>
    case (<name>:) <condition> -> <routing expression>
    else -> <when none of them matched>
  }
```

```
"hello world"  match {
    case word matches /(hello)\s+\w+/ -> word[1] as String ++ " was matched"
    case literalMatch: "hello world" -> upper(literalMatch)
    case hasOne if( hasOne is Object and hasOne.three? ) -> hasOne.three
    else -> $    /* case last if(true) -> last */
   }
```

GLOBAL VARIABLE

```
var myVar ="Hello"
```

LOCAL VARIABLE

```
var myAge = do { var age = "25" ---  age }
--- { Name : do { var name = "John Doe" ---  name }, Age : myAge}
```

```
{ "Name": "John Doe", "Age": "25"}
```

FORMATTING

CURRENCY

```
type Currency = String { format: "##"}
```

DATE

```
formattedDate: |2003-10-01T23:57:59|
 as String {format: "yyyy-MM-dd"}
```

ESCAPE CHAR

\

```
{"b": "dollar sign (\$)"}
```

Note: For escaping $$, use \$\$

REGEX

/\d+/

One or more digits from 0-9

/\s+/

One or more chars

```
["+1 (415) 123-7890"] map ($ match {
case phone matches /\+(\d+)\s\((\d+)\)\s(\d+\-\d+)/ -> {
country: phone[1], area: phone[2], number: phone[3]}})
```

```
[
  {
    "country": "1",
    "area": "415",
    "number": "123-7890"
  }
]
```

STRING INTERPOLATION

```
var name ="pk" ---
{ details: ""$name and age ${24 + 1}"}
```

```
{"details": "pk and age 25"}
```

Comments	// or /* */	
Object	Collection of name:value pairs	{ name: "John Doe" }
Array	[1,2]	

DYNAMIC ELEMENTS

var y = {e: "e"} --- { a: "a", (y)}	{"a": "a","e": "e"}

CONDITIONAL ASSIGNMENT

if (expression) "True value" else "False Value"	if (isOdd(3)) "value is odd" else "value is even"

IF ELSE

If Else	if (condition1) {} else if (condition2) {} else {}	if (payload.country =="USA") { currency: "USD" } else if (payload.country =="UK") { currency: "GBP" } else { currency: "EUR" }

JAVA

header (import function)	import java!utils::MyUtils::someFunction	imports class utils.MyUtils from src/main/java
header (import function directly)	import valueOf from java!java::lang::String	
body (creating a new instance)	java!java::lang::NullPointerException::new("foo")	

DYNAMIC KEY

var dynamicKey = "MyKey" --- (dynamicKey): " It works!"	{"MyKey": " It works!"}	

DATE

Date	|2003-10-01|
DateTime	|2003-10-01T23:57:59-03:00|
LocalDateTime	|2003-10-01T23:57:59|
LocalTime	A Time in the current TimeZone
Period	P<date>T<time> - P[n]Y[n]M[n]DT[n]H[n]M[n]S
Time	|23:59:56|
TimeZone	|-09:00|
Date Decomposition	<DATE>.year <DATE>.month

Test the selectors using the following input

```
<users xmlns="www.pk.corp"><user name="pk">John Doe</user><user name="vik">Vikram</user></users>
```

SELECTORS (Static)

Single Value	payload.users.user	"John Doe"
Multi-value	payload.users.*user	["John Doe", "Vikram"]
Descendants	payload..users.user	["John Doe"]
Key&Value	payload.users.&user	{ "user": "John Doe", "user": "Vikram"}
Index	payload.users[0]	"John Doe"
Range	payload.users.*user[0 to 1]	["John Doe", "Vikram"]
XML Attribute	payload.users.user.@name	"pk"
NameSpace	payload.users.#	www.pk.corp

SELECTORS(Dynamic)

Single Value	payload[("users")]	{ "user": "John Doe", "user": "Vikram"}
Multi-value	payload[*("users")]	[{ "user": "John Doe", "user": "Vikram"}]
Atributes	payload.users.*user[@"name"]	["pk","vik"]
Key&Value	payload.users[&("user")]	{ "user": "John Doe", "user": "Vikram"}
Single Value with NS	payload.ns0#"users" where ns ns0 www.pk.corp	{ "user": "John Doe", "user": "Vikram"}

SELECTORS

Key Present	payload.user?	"false"
Attribute Present	payload.users.user.@name?	"true"
Asset Present	payload.user!	Error - "There is no key named 'user'"
Filter (Array or null)	payload.users.*user[?($=="Vikram")]	["Vikram"]

CONDITIONAL

Element	(hello: "world") if (true)	{ "hello": "world"}
Array	[(1) if true, (2) if false]	[1]
XML Attribute conditional	{name @((age: "25") if true): "John Doe"}	<name age="25">John Doe</name>
XML Attributes Dynamic	transform @((payload.users)): "That changed everything"	<transform xmlns:wstxns1="www.pk.corp" wstxns1:user="John Doe" wstxns1:user="Vikram">That changed everything</transform>

Dataweave Library and Functions

pluralize	pluralize("bar")	bars
upper	upper("bar")	BAR
lower	lower("BAR")	bar
camelize	camelize("BAR")	bAR
capitalize	capitalize("bar")	Bar
p (read property)	Mule::p('http:port')	8081
lookup (execute flow)	Mule::lookup('flow2', {test:'hello '})	
causedBy	Mule::causedBy('HTTP:FORBIDDEN')	
flatten	flatten([[3],[null],null])	[3,null,null]
filter	[9,2,3,4,5] filter (value, index) -> (value > 2)	[9,3,4,5]
filterObject	{"a" : "apple", "b" : "banana"} filterObject ((value) -> value == "apple")	{ "a": "apple" }
find	["Bond", "James", "Bond"] find "Bond"	[0,2]

HOW TO

Rename Keys	inside mapObject use, (newkey: value) if(key as String == 'oldkey')	
Output conditionally	inside map use, (insurance: $.insurance) if($.insurance?)	
Default values	(SomeField default "DefaultValue")	(payload.someField default "my default value")
Conditional Assignment	if (expression) "True value" else "False Value"	if (isOdd(3)) "value is odd" else "value is even"
zip (use INPUT2)	payload map (item, index) -> { screws: zip(item.screws.size, item.screws.quantity)}	[{"screws":[[4,15],[6,8]]}]
Exclude Fields (-)	personal_information: $.personal_information - "ssn"	
reduce	[{"channels":["ABN","Gemini","ETV","NDTV"]}] map() ->{(channels: reduceMapFor($.channels)) if(sizeOf($.channels) > 0)}	[{ "channels": "ABN,Gemini,ETV ,NDTV" }]
Read XML	read(XML)	var myInput = read('<bookstore> <book>...
Insert Attributes in XML element	"@(<attributeName> : <value>)"	title @(lang: "en", year: "2001"): "Da Vinci Code"
Merge Fields from different objects	[{ "bookId":"101", "title":"world history" }] map (firstInputValue) -> { theTitle: firstInputValue.title, ([{ "bookId":"101", "author":"john doe"}] filter ($.*bookId contains firstInputValue.bookId) map (secondInputValue) -> theAuthor : secondInputValue.author)}	[{ "theTitle": "world history", "theAuthor": "john doe" }]
DWL expression from file	<when expression="${file::someFile.dwl}" >	
Load DWL as Module	import modules::MyModule import myFunc as myFunction, myVar as myVarValue from modules::MyModule	MyModule.dwl resides in /src/main/resources/modules

groupBy	["a","b","c"] groupBy (item, index) -> index	{"2":["c"],"1":["b"],"0":["a"] }

Dataweave Library and Functions

isBlank	"empty" : isBlank("")	"empty": true
isEmpty	[isEmpty([]), isEmpty([1])]	[true, false]
map	payload.users.*user map ((item,index) -> {user : item})	[{"user":"John Doe"},{"user":"Vikram"}]
mapObject	payload mapObject (value, key) -> (key): value	{"users":{"user":"John Doe","user":"Vikram"}}
match	"me@mulesoft.com" match(/([a-z]*)@([a-z]*).com/)	["me@mulesoft.com", "me", "mulesoft"]
orderBy	[{ letter: "e" }, { letter: "d" }] orderBy($.letter)	[{"letter":"d"},{"letter":"e"}]
matches	("john Doe123" matches /j.*\d+/)	"true"
pluck	{"name":"john","age":"25"} pluck (value,key,index) -> field: { (index) : { (value):key} }	[{"column":{"0":{"john":"name"}}},{"column":{"1":{"25":"age"}}}]
concat (++)	{ "concat" : ["A", "B"] ++ ["C", "D"]}	{"concat":["A","B","C","D"] }
remove (--)	{ "remove": ["A", "B"] -- ["A"] }	{"remove":["B"] }
Math Operations (+,-,/,*)	{ "Answer" : 2 * (2 / 2) +2 - 2 }	{ "Answer": 2.0}
Relational (>, >=, <,<=, ==, ~=) and Logical Operators (and, or, not)	var x=8 var strTrue="true" --- { "IsIt8?" : if((x>7 and x<9) or (x>=7 and x<=8) or (x==8)) "Yes" else "No","IsItReallyTrue?" : if(strTrue ~= true) "Yes" else "No" }	{ "IsIt8?": "Yes", "IsItReallyTrue?": "Yes" }
unzip	unzip([["silver","C"], ["Gold","B"], ["Platinum","A"]])	[["silver","Gold","Platinum"],["C","B","A"]]
scan	"(510)-456-1234" scan(/([(,)0-9]*)-([0-9]*)-([0-9]*)/)	[["(510)-456-1234","(510)","456","1234"]]

uuid	uuid()	"cafaae62-3b57-4879-a2aa-cadefa6c652f"
splitBy	"192.168.1.1/24" splitBy(/[.\/]/)	["192", "168", "1", "1", "24"]
replace with	{ "ssn" : "987-65-4321" replace /[0-9]/ with("x") }	{ "ssn": "xxx-xx-xxxx" }
write	write(payload.users, "application/csv", {"header":true, "separator" : "|"})	"user|user\nJohn Doe|Vikram\n"
to (Range)	{ "customRange": 1 to 5 }	{ "customRange": [1, 2, 3, 4, 5] }
envVar	envVar("SHELL")	"/bin/bash"
read	{ person : read('<name>john</name>','application/xml') }	{ "person": {"name": "john"}}
contains	{ "1or2There?" : if(contains("12345", /[1-2]/)) "yes" else "no" ,"cotainsRam?" : contains("Vikram","ram")}	{"1or2There?":"yes" ,"RamThere?":true }

Timezone

Add/Subtract a period (Year/Month/Days/Time) (P[n]Y[n]M[n]DT[n]H[n]M[n]S)	{anYearTwoMonthsAnd3DaysAgo: (now() - |P1Y2M3D|) as LocalDateTime {format : "dd-MM-yyyy'T'HH:mm:ss"}}	{ "anYearTwoMonthsAnd3DaysAgo": "29-01-2019T04:04:57" }
Append Timezone	{ "DateTime" : (|2019-10-01T23:57:59| ++ |-03:00|) }	{ "DateTime": "2019-10-01T23:57:59-03:00 }
Shift Timezone	{shiftedTime: (now() as LocalDateTime >> |+05:00|) as LocalDateTime {format : "yyyy-MM-dd'T'HH:mm:ss.SSS'Z'"}}	{ "shiftedTime": "2020-04-01T09:12:53.784Z" }
Shift Timezone	output application/java --- { gmtTime: now() >> ("GMT" as TimeZone)}	{gmtTime=2020-04-01T04:31:15.416Z[GMT]}
Days Difference	{ Days : (now() - |2019-12-31|)/(24*3600), Days : daysBetween('2017-10-01', '2017-11-01') }	{ "Days": 92.0, "Days": 31 }