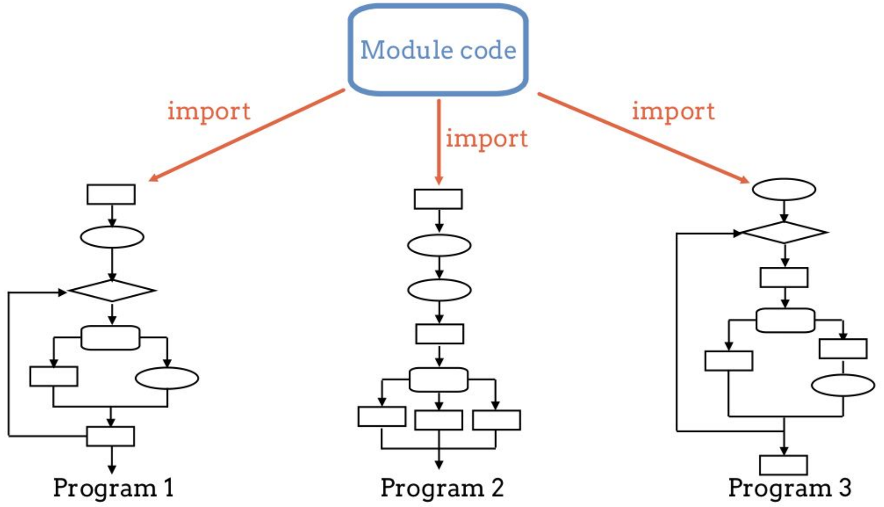
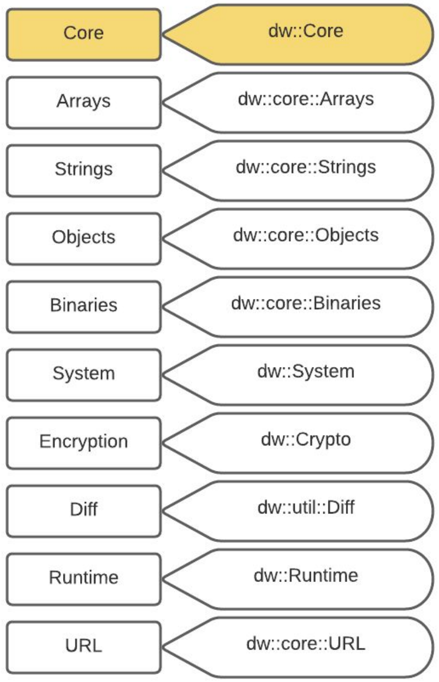
Dataweave Modules

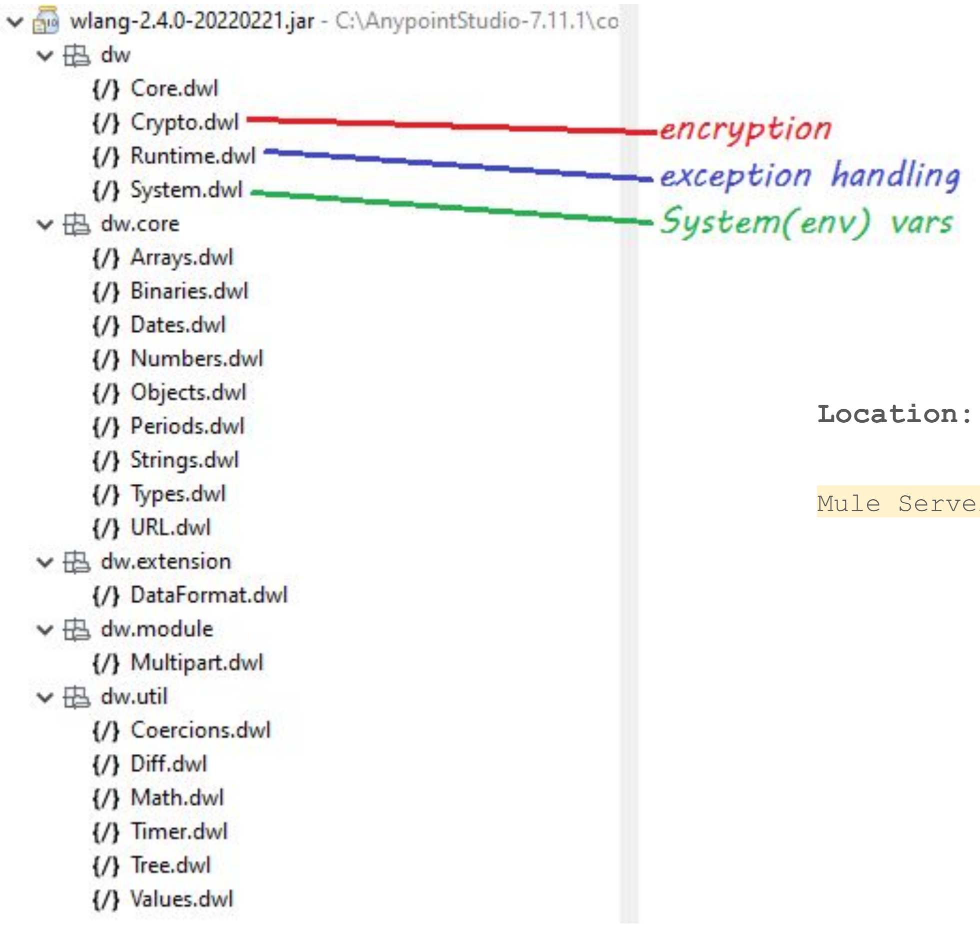
- Mehak Batra

Modules - Implementation of specific functionality





Core	dw::Core
Arrays	dw::core::Arrays
Strings	dw::core::Strings
Objects	dw::core::Objects
Binaries	dw::core::Binaries
System	dw::System
Encryption	dw::Crypto
Diff	dw::util::Diff
Runtime	dw::Runtime
URL	dw::core::URL



wlang-2.4.0-20220221.jar - C:\AnypointStudio-7.11.1\co
- dw
  - Core.dwl
  - Crypto.dwl
  - Runtime.dwl
  - System.dwl
- dw.core
  - Arrays.dwl
  - Binaries.dwl
  - Dates.dwl
  - Numbers.dwl
  - Objects.dwl
  - Periods.dwl
  - Strings.dwl
  - Types.dwl
  - URL.dwl
- dw.extension
  - DataFormat.dwl
- dw.module
  - Multipart.dwl
- dw.util
  - Coercions.dwl
  - Diff.dwl
  - Math.dwl
  - Timer.dwl
  - Tree.dwl
  - Values.dwl

Location:

`Mule Server 4.4.0 EE/wlang-2.4.0-20211228.jar`

Core

This module is imported by default

function	**description**
avg	Returns the average of numbers listed in an array.
contains	Returns true if an input contains a given value, false if not.
distinctBy	Iterates over an array and returns the unique elements in it.
endsWith	Returns true if a string ends with a provided substring, false if not.
filter	Iterates over an array and applies an expression that returns matching values.
filterObject	Iterates a list of key-value pairs in an object and applies an expression that returns only matching objects, filtering out the rest from the output.
find	Returns indices of an input that match a specified value.
flatten	Turns a set of subarrays (such as [[1,2,3], [4,5,[6]], [], [null]]) into a single, flattened array (such as [1, 2, 3, 4, 5, [6], null
isBlank	Returns true if the given string is empty or completely composed of whitespace, false if not.

function	**description**
isEven	Returns true if the number or numeric result of a mathematical operation is even, false if not.
isOdd	Returns true if the number or numeric result of a mathematical operation is odd, false if not.
joinBy	Merges an array into a single string value and uses the provided string as a separator between each item in the list.
log	Without changing the value of the input, log returns the input as a system log.
lower	Returns the provided string in lowercase characters.
map	Iterates over items in an array and outputs the results into a new array.
mapObject	Iterates over an object using a mapper that acts on keys, values, or indices of that object.
match	Uses a Java regular expression (regex) to match a string and then separates it into capture groups. Returns the results in an array.
matches	Checks if an expression matches the entire input string.
random	Returns a pseudo-random number greater than or equal to 0.0 and less than 1.0.
read	Reads a string or binary and returns parsed content.
reduce	Applies a reduction expression to the elements in an array.
scan	Returns an array with all of the matches found in an input string.
write	Writes a value as a string or binary in a supported format.
zip	Merges elements from two arrays into an array of arrays.

code:

```
%dw 2.0
output application/json
---

{
   "avg": avg(1 to 15),
   "contains": [1,4,5,6]  contains(5),
   "distinctBy": [1,2,2,4,1] distinctBy $,
   "endsWith": "mehak" endsWith("m"),
   "flatten": flatten([[1,8,9],["a","b","c"]]),
   "find": [8,9,0] find 9,
   "isBlank": isBlank("    "),
   "isEmpty": isEmpty("    "),
   "log" : log("WARNING", "we are learning core"),
   "match": "mehak@mulesoft.com" match(/([a-z]*)@([a-z]*).com/),
   "matches": "mehak@mulesoft.com" matches(/([a-z]*)@([a-z]*).in/),
   "read": read("<?xml version='1.0' encoding='UTF-8'?><hello>world</hello>" ,'application/xml'),
   "trim": trim("   core module     "),
   "zip": [0,1] zip ["a","b"]
}
```

Arrays (dw::core::Arrays)

function	description
countBy	Counts the elements in an array that match the results of a function.
drop	Drops the first n elements. It returns the original array when n <= 0 and an empty array when n > sizeOf(array).
dropWhile	Drops elements from the array while the condition is met but stops the selection process when it reaches an element that fails to satisfy the condition.
leftJoin	Joins two arrays of objects by a given ID criteria.
splitAt	Splits an array into two at a given position.
take	Selects the first n elements. It returns an empty array when n <= 0 and the original array when n > sizeOf(array).
takeWhile	Selects elements from the array while the condition is met but stops the selection process when it reaches an element that fails to satisfy the condition.

```
%dw 2.0
import * from dw::core::Arrays
output application/json
var arr1= [{"a": 9,"b": 4},{"a": 5,"b": 94},{"a":
29,"b": 74}]
var arr2= [{"a": 9,"c": 0},{"a": 7,"b": 54}]
---
{
  "countBy": [1, 2, 3, 7] countBy (isEven($)),
 "drop": [1, 2, 3, 7] drop 2,
 "indexWhere" : ["mehak","test","try"] indexWhere
()->$ startsWith "t",
 "splitAt": [1,3,7,9,3,5] splitAt 2,
 "take":[1, 2, 3, 7] take 2,
 "takeWhile": [1, 9, 3, 7,1] takeWhile $ <= 2,
 "every": [1, 8, 3, 7,1] every (isEven($)),
 "some" : [1, 8, 3, 7,1] some (isEven($)),
 "join": join(arr1,arr2,(abc) ->abc.a,(abc)->abc.a),
 "leftJoin":
leftJoin(arr1,arr2,(abc) ->abc.a,(abc)->abc.a),
 "outerJoin":
outerJoin(arr1,arr2,(abc) ->abc.a,(abc)->abc.a)
}
```

Binaries (dw::core::Binaries)

Function	description
fromBase64	Transforms a Base64 string into a binary value.
fromHex	Transforms a hexadecimal string into a binary.
readLinesWith	Splits the specified binary content into lines and returns the results in an array.
toBase64	Transforms a binary value into a Base64 string.
toHex	Transforms a binary value into a hexadecimal string.
writeLinesWith	Writes the specified lines and returns the binary content.

```
%dw 2.0
import * from dw::core::Binaries
output application/json
---

{
  "toBase64": toBase64("hello World" as Binary),
  "fromBase64": fromBase64("aGVsbG8gV29ybGQ="),
  "toHex": toHex("hello World" as Binary),
  "fromHex": fromHex("68656C6C6F20576F726C64")
}
```

Objects (dw::core::Objects)

function	description
divideBy	Breaks up an object into sub-objects that contain the specified number of key-value pairs.
entrySet	Returns an array of key-value pairs that describe the key, value, and any attributes in the input object.
everyEntry	Returns true if every entry in the object matches the condition.
mergeWith	Appends any key-value pairs from a source object to a target object.
someEntry	Returns true if at least one entry in the object matches the specified condition.
takeWhile	Selects key-value pairs from the object while the condition is met.
valueSet	Returns an array of the values from key-value pairs in an object.

```
%dw 2.0
import * from dw::core::Objects
output application/json
---

{
  "divideBy" : {"a": 1, "b" : true, "a" : 2, "b" : false, "c" : 3} divideBy 2,
  //keySet replaced by keysOf(core module)
  "keysOf" : keysOf({ "a" : true, "b" : 1}),
  "mergeWith" : { "a" : true, "b" : 1} mergeWith { "a" : false, "c" : "Test"},
  "takeWhile": {"a": 1, "b" : 5, "a" : 2, "b" : 6, "c" : 3} takeWhile ((value, key) ->  value < 3),
  //valueSet replaced by valuesOf(core module)
  "valuesOf" : valuesOf({ "a" : true, "b" : 1}),

}
```

URL (dw::core::URL)

function	**description**
decodeURI	Decodes the escape sequences (such as %20) in a URI.
encodeURI	Encodes a URI with UTF-8 escape sequences.
parseURI	Parses a URL and returns a URI object.

The function *does not encode these characters* with UTF-8 escape sequences:

Type (not escaped)	**Examples**
Reserved characters	; , / ? : @ & = $
Unescaped characters	alphabetic, decimal digits, - _ . ! ~ * ' ( )
Number sign	#

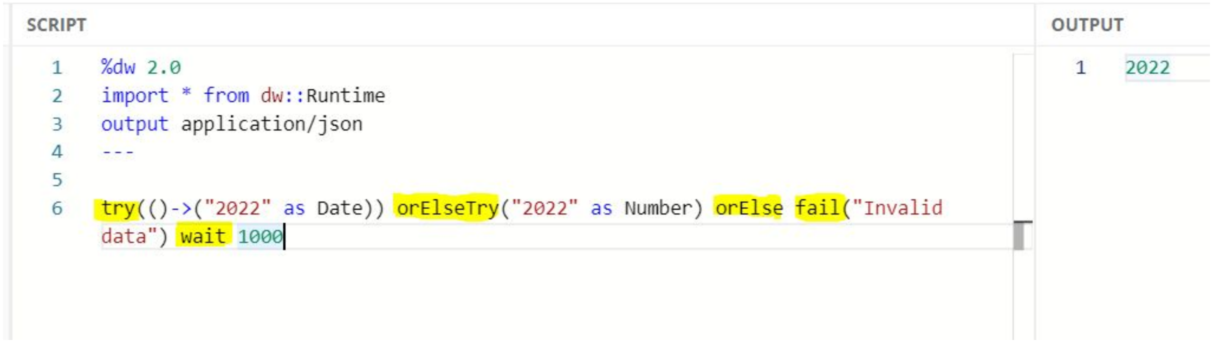
```
%dw 2.0
import * from dw::core::URL
output application/json

---
{
"decodeURI" :
decodeURI('http://test/%20text%20to%20decode
%20/text'),
"encodeURI" : encodeURI("http://test/ text
to decode /text"),
"not_encoded":
encodeURI("http://:;,/?:@&=\$_-_.!~*'()"),
"parseURI": parseURI("http://test/ text to
decode /text"),
"parseURI_encoded":
parseURI('http://test/%20text%20to%20decode%
20/text')
}
```

Runtime (dw::Runtime)

This module contains functions for interacting with the DataWeave runtime/engine

- **fail**
- **failIf**
- **try**
- **orElse**
- **orElseTry**
- **wait**
- **location**
- **prop**
- **props**

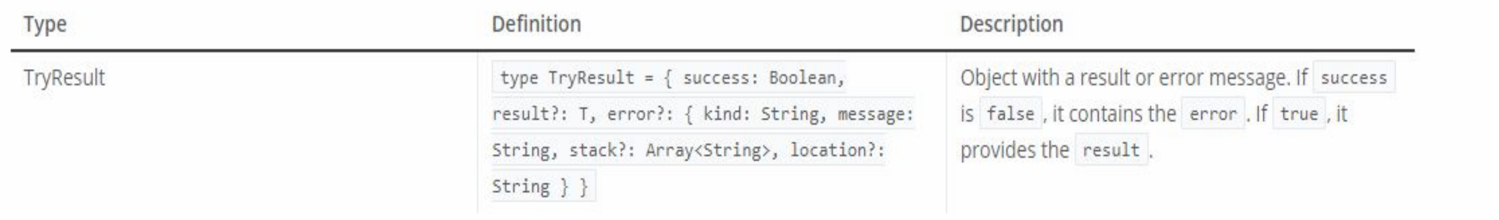

SCRIPT

```
%dw 2.0
import * from dw::Runtime
output application/json
---

try(()->("2022" as Date)) orElseTry("2022" as Number) orElse fail("Invalid
data") wait 1000
```

OUTPUT

```
2022
```



Type	Definition	Description
TryResult	`type TryResult = { success: Boolean, result?: T, error?: { kind: String, message: String, stack?: Array<String>, location?: String } }`	Object with a result or error message. If `success` is `false`, it contains the `error`. If `true`, it provides the `result`.

Example:

```
Code:

%dw 2.0
import * from dw::Runtime
output application/json
---

try(()->(payload.key as Number)) orElseTry (payload.key as Date) orElse fail("oops wrong key! please try
again")
//location(fail)
```

Crypto

functions that perform encryptions through common algorithms, such as MD5, SHA1, and so on.

```
import * from dw::Crypto
```

HMAC

Hash-based Message Authentication Code (HMAC) : Hash +Cryptographic key

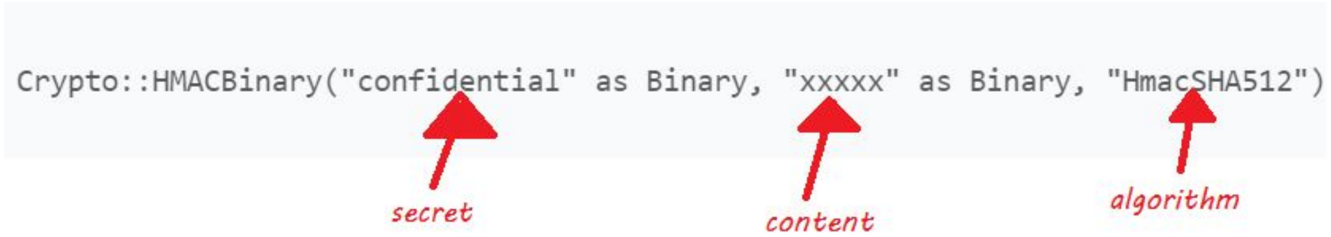
Hash-based message authentication code (HMAC) provides the server and the client each with a private key

The client creates a unique HMAC, or hash, per request to the server by hashing the request data with the private keys and sending it as part of a request

HMACBinary

`Syntax: HMACBinary(Binary, Binary, String): Binary`

Computes an HMAC hash (with a secret cryptographic key) on input content

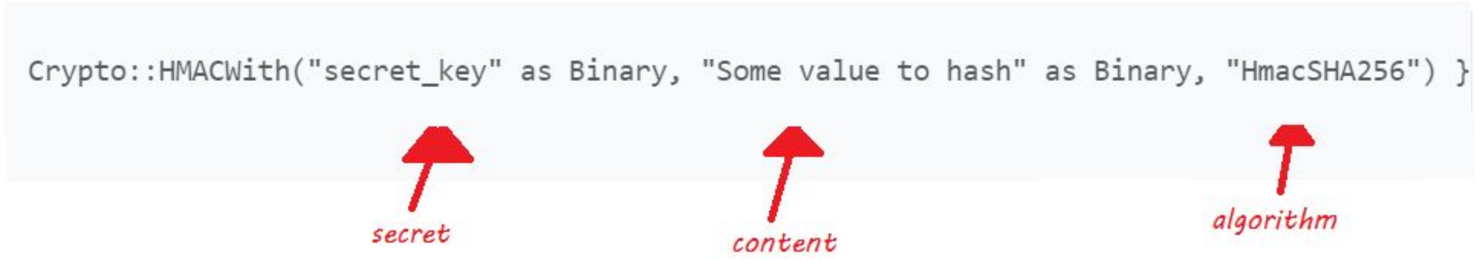


```
Crypto::HMACBinary("confidential" as Binary, "xxxxx" as Binary, "HmacSHA512")
```

HMACWith

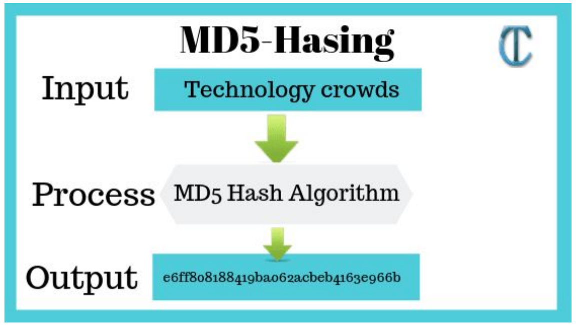
`Syntax: HMACWith(Binary, Binary, String): String`

Computes an HMAC hash (with a secret cryptographic key) on input content, then transforms the result into a lowercase, hexadecimal string.



```
Crypto::HMACWith("secret_key" as Binary, "Some value to hash" as Binary, "HmacSHA256") }
```

MD5 - primarily used for authenticating files



```
{ "md5" : Crypto::MD5("asd" as Binary) }
```

SHA1

```
Syntax: SHA1(Binary): String
```

Computes the SHA1 hash and transforms the result into a hexadecimal, lowercase string.

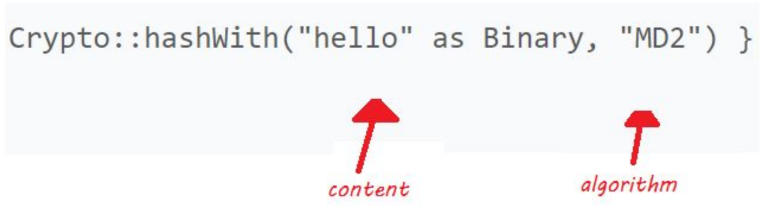
takes an input and produces a 160-bit hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long.

```
{ "sha1" : Crypto::SHA1("dsasd" as Binary) }
```

hashWith

```
Syntax: hashWith(Binary, String): Binary
```

Computes the hash value of binary content using a specified algorithm.



```
Crypto::hashWith("hello" as Binary, "MD2") }
```

System (dw::System)

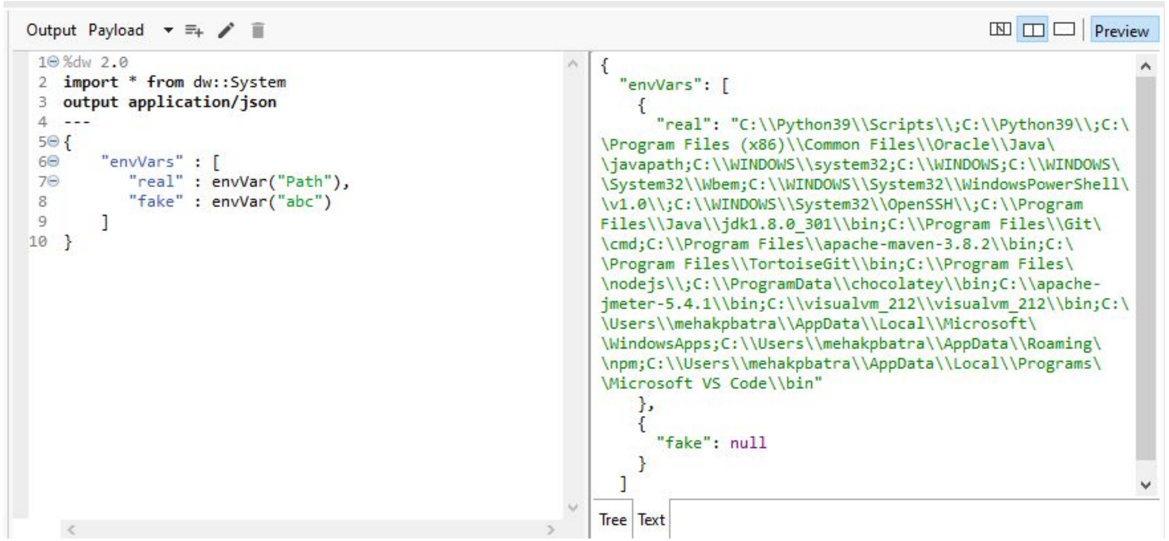
contains functions that allow you to interact with the underlying system.

envVar

Returns an environment variable with the specified name or `null` if the environment variable is not defined.

```
Syntax: envVar(String): String | Null
```

```
%dw 2.0
import * from dw::System
output application/json
---
{
    "envVars" : [
        "real" : envVar("Path"),
        "fake" : envVar("abc")
    ]
}
```



```
%dw 2.0
import * from dw::System
output application/json
---
{
    "envVars" : [
        "real" : envVar("Path"),
        "fake" : envVar("abc")
    ]
}
```

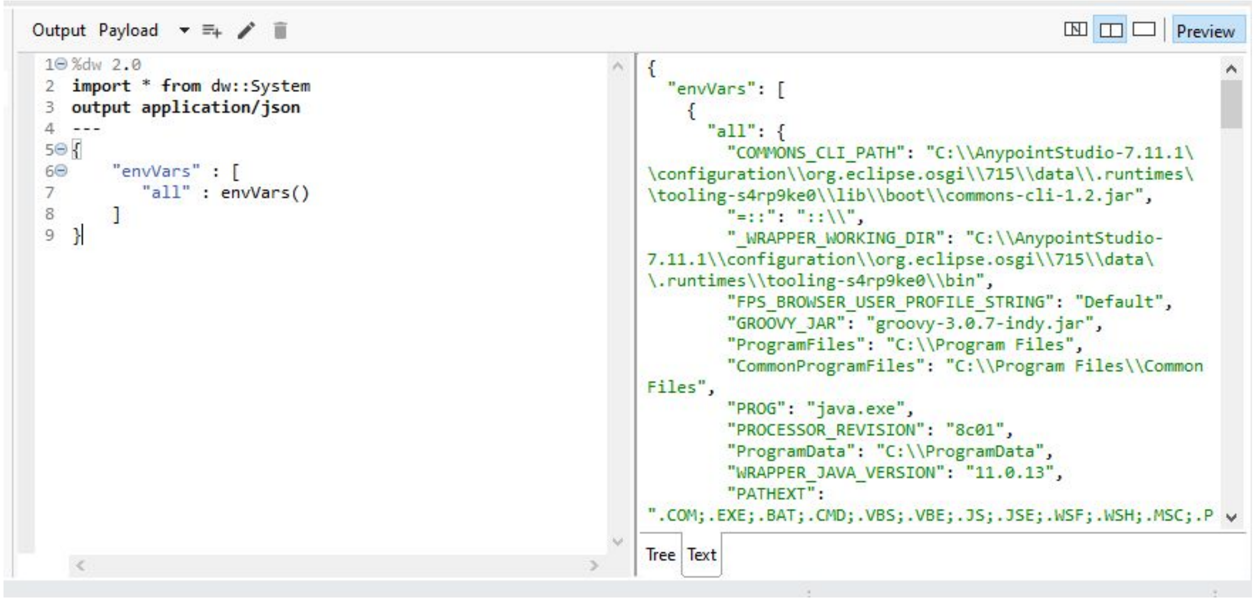
```
{
  "envVars": [
    {
      "real": "C:\\Python39\\Scripts\\;C:\\Python39\\;C:\
\Program Files (x86)\\Common Files\\Oracle\\Java\
\javapath;C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\
\System32\\Wbem;C:\\WINDOWS\\System32\\WindowsPowerShell\
\v1.0\\;C:\\WINDOWS\\System32\\OpenSSH\\;C:\\Program
Files\\Java\\jdk1.8.0_301\\bin;C:\\Program Files\\Git\
\cmd;C:\\Program Files\\apache-maven-3.8.2\\bin;C:\
\Program Files\\TortoiseGit\\bin;C:\\Program Files\
\nodejs\\;C:\\ProgramData\\chocolatey\\bin;C:\\apache-
jmeter-5.4.1\\bin;C:\\visualvm_212\\visualvm_212\\bin;C:\
\Users\\mehakpbatra\\AppData\\Local\\Microsoft\
\WindowsApps;C:\\Users\\mehakpbatra\\AppData\\Roaming\
\npm;C:\\Users\\mehakpbatra\\AppData\\Local\\Programs\
\Microsoft VS Code\\bin"
    },
    {
      "fake": null
    }
  ]
```

envVars

envVars(): Dictionary<String>

Returns all of the environment variables defined in the host system.

```
%dw 2.0
import * from dw::System
output application/json
---
{
    "envVars" : [
        "all" : envVars()
    ]
}
```



```
%dw 2.0
import * from dw::System
output application/json
---
{
    "envVars" : [
        "all" : envVars()
    ]
}
```

```
{
  "envVars": [
    {
      "all": {
        "COMMONS_CLI_PATH": "C:\\AnypointStudio-7.11.1\
\configuration\\org.eclipse.osgi\\715\\data\\.runtimes\
\tooling-s4rp9ke0\\lib\\boot\\commons-cli-1.2.jar",
        "=::": "::\\",
        "_WRAPPER_WORKING_DIR": "C:\\AnypointStudio-
7.11.1\\configuration\\org.eclipse.osgi\\715\\data\
\.runtimes\\tooling-s4rp9ke0\\bin",
        "FPS_BROWSER_USER_PROFILE_STRING": "Default",
        "GROOVY_JAR": "groovy-3.0.7-indy.jar",
        "ProgramFiles": "C:\\Program Files",
        "CommonProgramFiles": "C:\\Program Files\\Common
Files",
        "PROG": "java.exe",
        "PROCESSOR_REVISION": "8c01",
        "ProgramData": "C:\\ProgramData",
        "WRAPPER_JAVA_VERSION": "11.0.13",
        "PATHEXT":
".COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.P
```

Multipart (dw::module::Multipart)

functions for creating MultiPart and formats and parts (including fields and boundaries) of MultiPart formats.

function	description
field	Creates a MultipartPart data structure using the specified part name, input content for the part, format (or mime type), and optionally, file name.
file	Creates a MultipartPart data structure from a resource file.
form	Creates a Multipart data structure using a specified array of parts.
generateBoundary	Helper function for generating boundaries in Multipart data structures.

```
%dw 2.0
import * from dw::module::Multipart
output multipart/form-data
var firstPart = "content for my first part"
var secondPart = "content for my second part"
---
{
 parts: {
   part1: field({name:"myFirstPart",value:
firstPart}),
   part2: field({name:"mySecondPart",value:
secondPart})
 }
}
```

Util (dw::util)

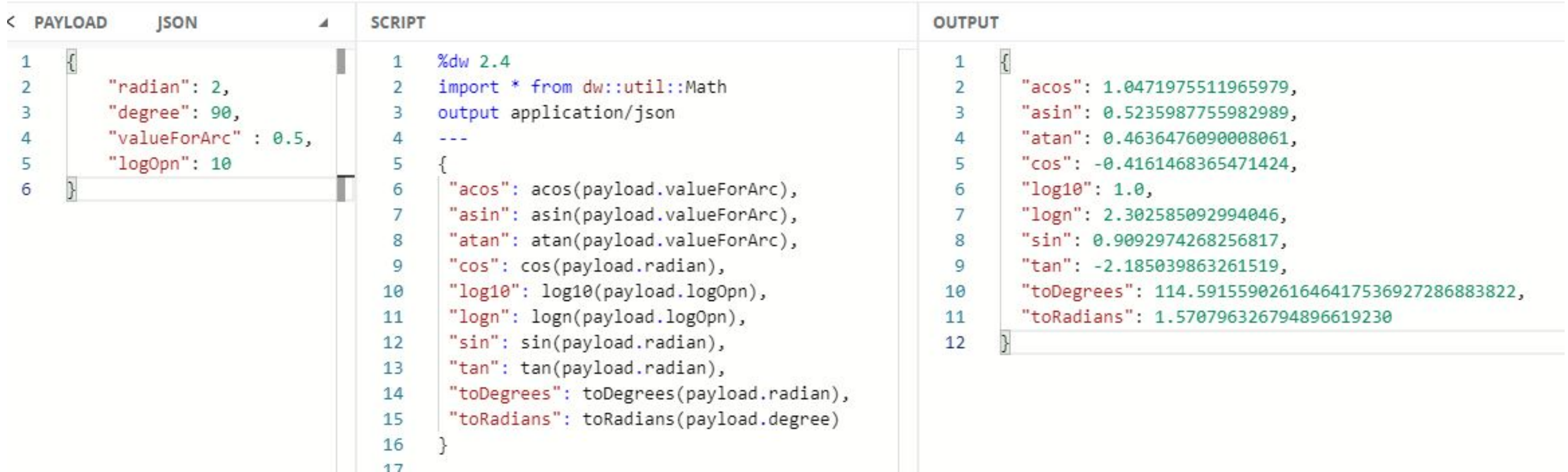
Class	Utility description	functions
Diff	calculates the difference between two values and returns a list of differences.	diff

```
%dw 2.0
import * from dw::util::Diff
output application/json

---
{
 "sameString": diff(payload.message,"hello"),
 "diffKey": diff({a:1}, {b:1}),
 "diffSize": diff({a: 1, b:1}, {b:1}),
 "diffType": diff([1,2], {b:1}),
 "restrictOrder": diff({a: 1, b:1}, {b: 1, a:1}, {unordered: false}),
 "noOrderMatch" : diff({a: 1, b:1}, {b: 1, a:1}, {unordered: true})
 }
```

Class	Utility description	functions	function description
Coercions	This utility is used for type conversion. Introduced in DataWeave version 2.4.0.	toArray	Splits a String value into an Array of characters.
		toBinary	Transform a String value into a Binary value using the specified encoding.
		toBoolean	Transform a String value into a Boolean value.
		toDate	Transforms a String value into a Date value and accepts a format and locale.
		toDateTime	Transforms a Number value into a DateTime value using milliseconds or seconds as the unit.
			Transforms a String value into a DateTime value and accepts a format and locale.
		toLocalDateTime	Transforms a String value into a LocalDateTime value and accepts a format and locale.
		toLocalTime	Transforms a String value into a LocalTime value and accepts a format and locale.
		toNumber	A variant of toNumber that transforms a DateTime value into a number of seconds or milliseconds, depending on the selected unit.
		toPeriod	Transform a String value into a Period value.
		toRegex	Transforms a String value into a Regex value.
		toString	A variant of toString that transforms a Number value (whole or decimal) into a String value and accepts a format, locale, and rounding mode value.
		toTime	Transforms a String value into a Time value and accepts a format and locale.
		toTimeZone	Transform a String value into a TimeZone value.
		toUri	Transforms a String value into a Uri value.

Class	Utility description	functions	Description
Math	provides mathematical functions. Introduced in DataWeave version 2.4.0.	acos	Returns an arc cosine value that can range from 0.0 through pi.
		asin	Returns an arc sine value that can range from -pi/2 through pi/2.
		atan	Returns an arc tangent value that can range from -pi/2 through pi/2.
		cos	Returns the trigonometric cosine of an angle from a given number of radians.
		log10	Returns the logarithm base 10 of a number.
		logn	Returns the natural logarithm (base e) of a number.
		sin	Returns the trigonometric sine of an angle from a given number of radians.
		tan	Returns the trigonometric tangent of an angle from a given number of radians.
		toDegrees	Converts an angle measured in radians to an approximately equivalent number of degrees.
		toRadians	Converts a given number of degrees in an angle to an approximately equivalent number of radians.



```
{
    "radian": 2,
    "degree": 90,
    "valueForArc" : 0.5,
    "logOpn": 10
}
```

```
%dw 2.4
import * from dw::util::Math
output application/json
---
{
 "acos": acos(payload.valueForArc),
 "asin": asin(payload.valueForArc),
 "atan": atan(payload.valueForArc),
 "cos": cos(payload.radian),
 "log10": log10(payload.logOpn),
 "logn": logn(payload.logOpn),
 "sin": sin(payload.radian),
 "tan": tan(payload.radian),
 "toDegrees": toDegrees(payload.radian),
 "toRadians": toRadians(payload.degree)
}
```

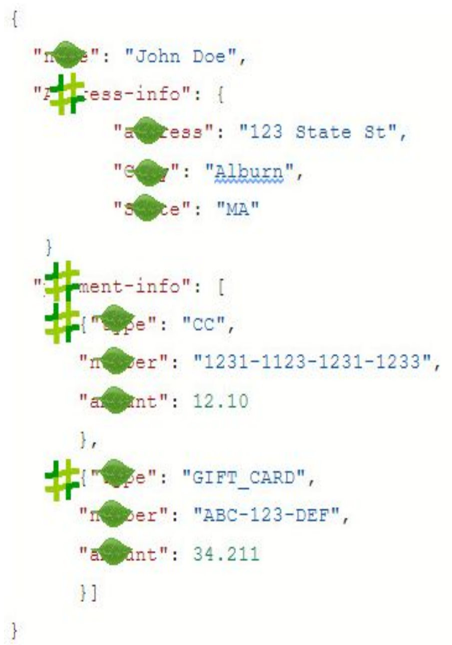
```
{
  "acos": 1.0471975511965979,
  "asin": 0.5235987755982989,
  "atan": 0.4636476090008061,
  "cos": -0.4161468365471424,
  "log10": 1.0,
  "logn": 2.302585092994046,
  "sin": 0.9092974268256817,
  "tan": -2.185039863261519,
  "toDegrees": 114.5915590261646417536927286883822,
  "toRadians": 1.570796326794896619230
}
```

Class	Utility description	functions	function description
Timer	contains functions for measuring time.	currentMilliseconds	Returns the current time in milliseconds.
		duration	Executes the input function and returns an object with execution time in milliseconds and result of that function.
		time	Executes the input function and returns a TimeMeasurement object that contains the start and end time for the execution of that function, as well the result of the function.
		toMilliseconds	Returns the representation of a specified date-time in milliseconds.

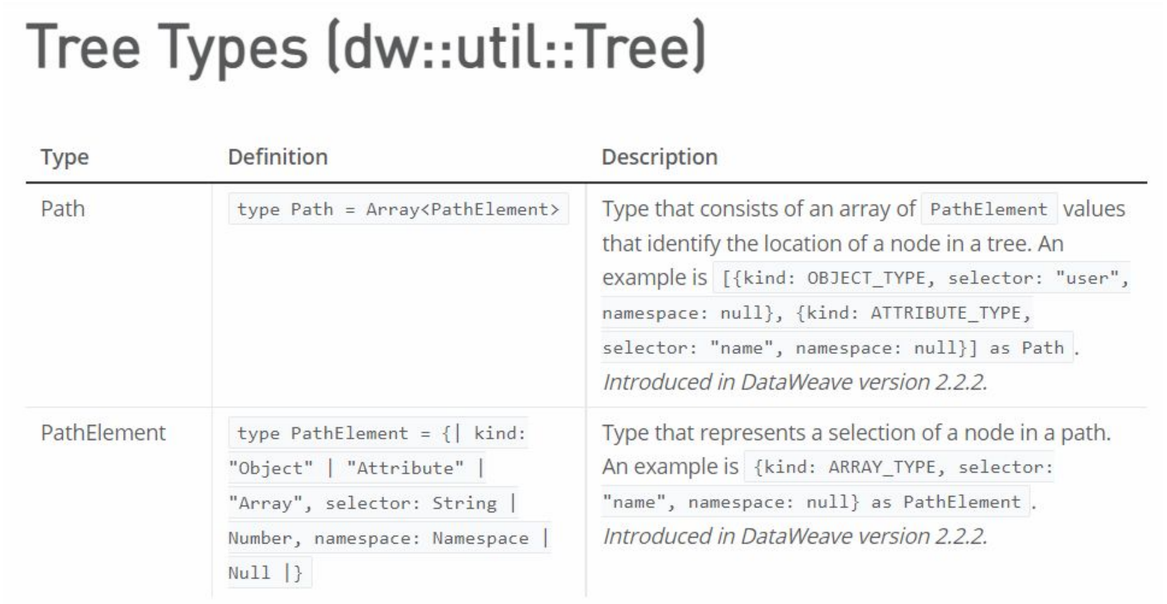
```
%dw 2.4
import * from dw::util::Timer
import * from dw::Runtime
output application/json
fun timeCheck() = (payload dw::Runtime::wait 1000)
---
{
"test" : "hello" wait 400,
"currentMilliseconds": currentMilliseconds(),
"duration":duration(()->timeCheck()),
"time": time(()->timeCheck()),
"toMilliseconds": toMilliseconds(now()),
}
```

Nodes that contain simple data types like integers, decimal values, dates and strings are like leaves on a tree

Tree:



```
{
  "name": "John Doe",
  "address-info": {
        "address": "123 State St",
        "city": "Alburn",
        "state": "MA"
  }
  "payment-info": [
    {"type": "CC",
     "number": "1231-1123-1231-1233",
     "amount": 12.10
     },
    {"type": "GIFT_CARD",
     "number": "ABC-123-DEF",
     "amount": 34.211
     }]
}
```



Tree Types (dw::util::Tree)

Type	Definition	Description
Path	`type Path = Array<PathElement>`	Type that consists of an array of `PathElement` values that identify the location of a node in a tree. An example is `[{kind: OBJECT_TYPE, selector: "user", namespace: null}, {kind: ATTRIBUTE_TYPE, selector: "name", namespace: null}] as Path`. *Introduced in DataWeave version 2.2.2.*
PathElement	`type PathElement = {| kind: "Object" | "Attribute" | "Array", selector: String | Number, namespace: Namespace | Null |}`	Type that represents a selection of a node in a path. An example is `{kind: ARRAY_TYPE, selector: "name", namespace: null} as PathElement`. *Introduced in DataWeave version 2.2.2.*

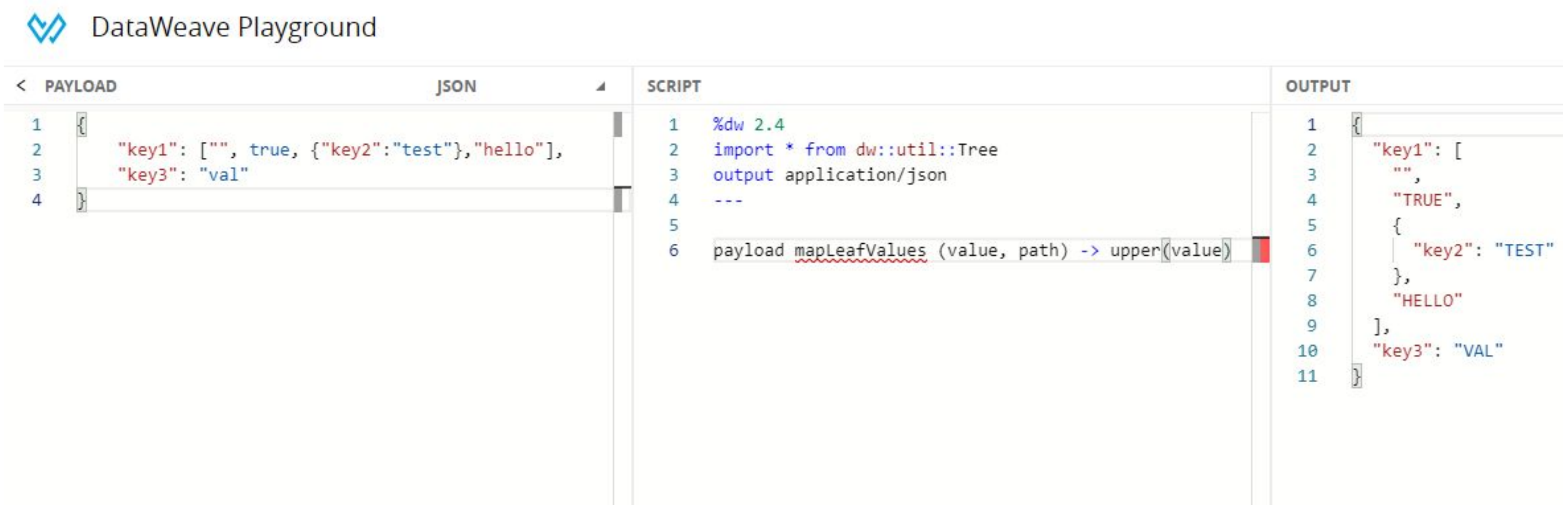
Class	Utility description	functions	function description
Tree	provides functions for handling values as tree-data structures. Introduced in DataWeave version 2.2.2.	asExpressionString	Transforms a Path value into a string representation of the path.
		filterArrayLeafs	Applies a filtering expression to leaf or Path values of an array.
		filterObjectLeafs	Applies a filtering expression to leaf or Path values of keys in an object.
		filterTree	Filters the value or path of nodes in an input based on a specified criteria.
		isArrayType	Returns true if the provided Path value is an ARRAY_TYPE expression.
		isAttributeType	Returns true if the provided Path value is an ATTRIBUTE_TYPE expression.
		isObjectType	Returns true if the provided Path value is an OBJECT_TYPE expression.
		mapLeafValues	Maps the terminal (leaf) nodes in the tree.
		nodeExists	Returns true if any node in a given tree validates against the specified criteria.

INPUT:

```
{
   "k1": ["", true,
{"k2":"test"},{"k3": 0} ,"hello"],
   "k4": "val",
   "k5": [3,8]
}
```

Code:

```
%dw 2.4
import * from dw::util::Tree
output application/json
---
{
   "conditionOnValue_Array":payload filterArrayLeafs ((value, path) ->  !(value is String)),
   "conditionOnPath_Array":payload filterArrayLeafs ((value, path) ->  isObjectType(path)),
    "conditionOnValue_Object":payload filterObjectLeafs ((value, path) ->  !(value is String)),
   "conditionOnPath_Object":payload filterObjectLeafs ((value, path) ->  isArrayType(path)),
    "conditionOnValue_tree":payload filterTree ((value, path) ->  !(value is String)),
   "conditionOnPath_tree":payload filterTree ((value, path) ->  isObjectType(path))
}
```



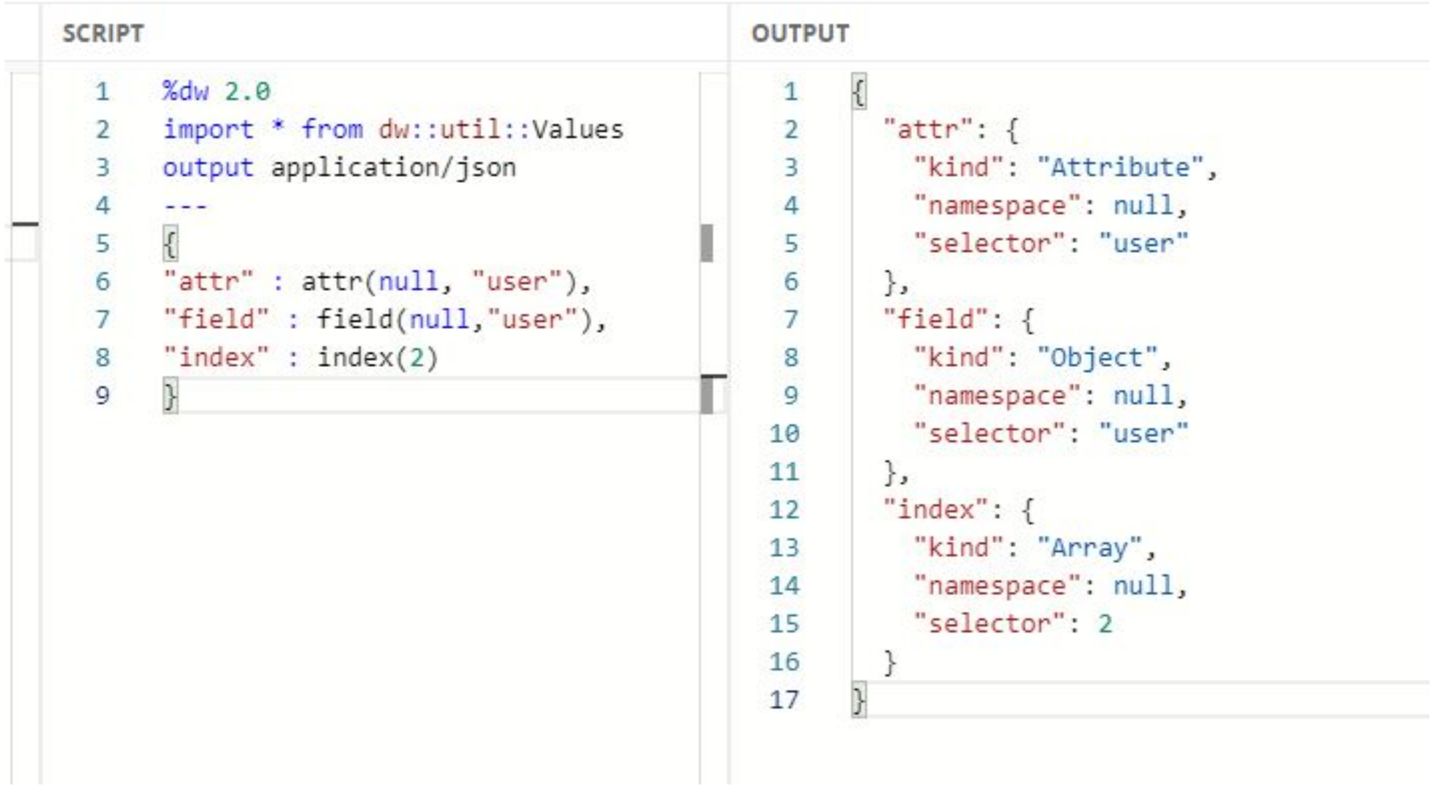
```json
{
    "key1": ["", true, {"key2":"test"},"hello"],
    "key3": "val"
}
```

```
%dw 2.4
import * from dw::util::Tree
output application/json
---

payload mapLeafValues (value, path) -> upper(value)
```

```json
{
  "key1": [
    "",
    "TRUE",
    {
      "key2": "TEST"
    },
    "HELLO"
  ],
  "key3": "VAL"
}
```

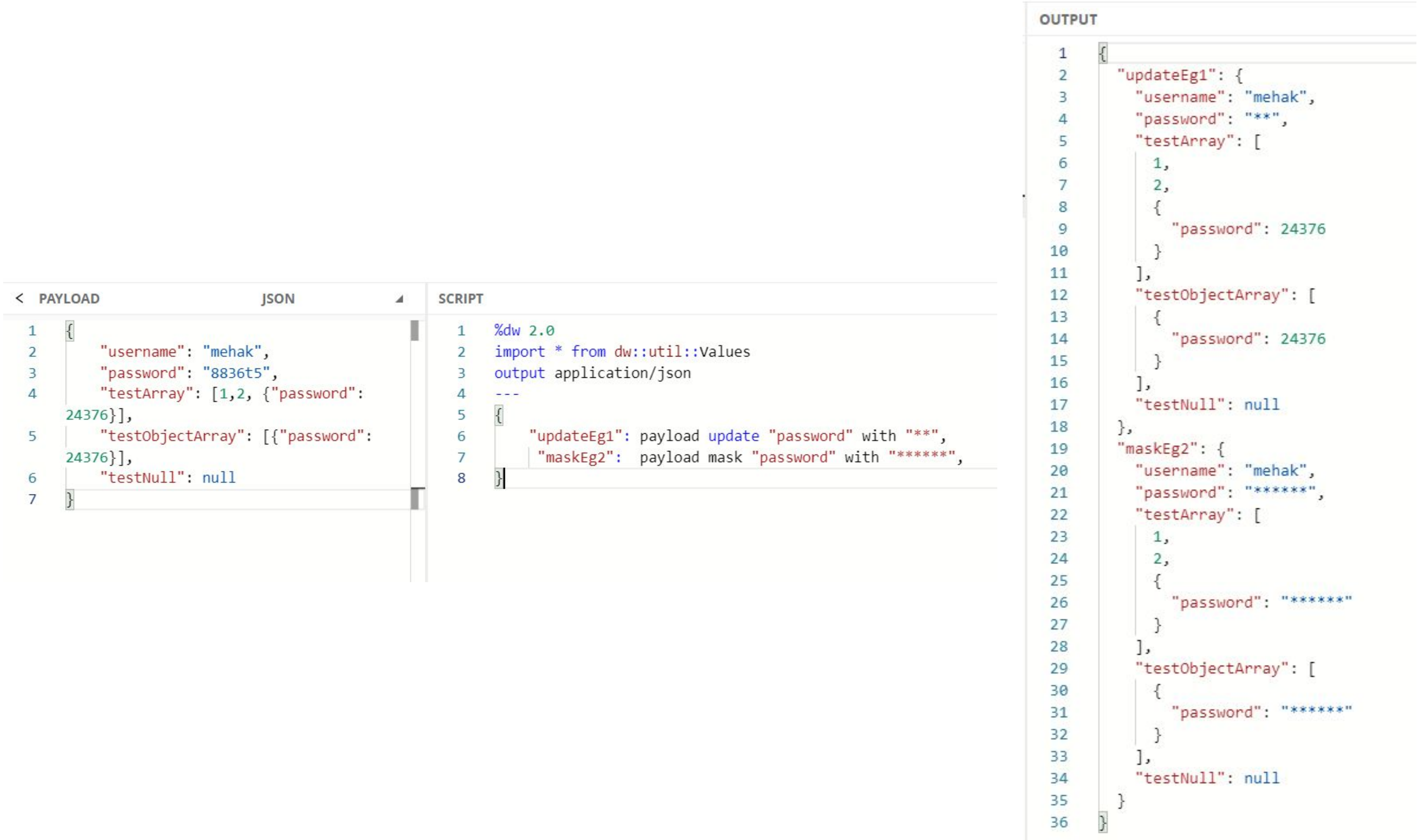
Class	Utility description	functions	function description
Values	This utility module simplifies changes to values. Introduced in DataWeave version 2.2.2.	attr	This function creates a PathElement to use for selecting an XML attribute and populates the type’s selector field with the given string.
		field	This function creates a PathElement data type to use for selecting an object field and populates the type’s selector field with the given string.
		index	This function creates a PathElement data type to use for selecting an array element and populates the type’s selector field with the specified index.
		mask	This mask function replaces all simple elements that match the specified criteria.
		update	This update function updates a field in an object/array with the specified string value.


SCRIPT

```
%dw 2.0
import * from dw::util::Values
output application/json
---
{
"attr" : attr(null, "user"),
"field" : field(null,"user"),
"index" : index(2)
}
```

OUTPUT

```
{
  "attr": {
    "kind": "Attribute",
    "namespace": null,
    "selector": "user"
  },
  "field": {
    "kind": "Object",
    "namespace": null,
    "selector": "user"
  },
  "index": {
    "kind": "Array",
    "namespace": null,
    "selector": 2
  }
}
```


PAYLOAD JSON

```
{
    "username": "mehak",
    "password": "8836t5",
    "testArray": [1,2, {"password":
24376}],
    "testObjectArray": [{"password":
24376}],
    "testNull": null
}
```

SCRIPT

```
%dw 2.0
import * from dw::util::Values
output application/json
---
{
    "updateEg1": payload update "password" with "**",
     "maskEg2":  payload mask "password" with "******",
}
```

OUTPUT

```
{
  "updateEg1": {
    "username": "mehak",
    "password": "**",
    "testArray": [
      1,
      2,
      {
        "password": 24376
      }
    ],
    "testObjectArray": [
      {
        "password": 24376
      }
    ],
    "testNull": null
  },
  "maskEg2": {
    "username": "mehak",
    "password": "******",
    "testArray": [
      1,
      2,
      {
        "password": "******"
      }
    ],
    "testObjectArray": [
      {
        "password": "******"
      }
    ],
    "testNull": null
  }
}
```

Custom Module

Custom modules are separate dwl files. You place them in **src/main/resources** catalog in for example **dw** folder. However, you can use the package name whatever you like for example modules/json/utils.

Your custom module may contain only:

- variable declaration – **var**
- functions – **fun**
- namespace declaration – **ns**
- custom types – **type**

In such a file, **output** directive is not permitted, as well as headers-body separator **—**. Because the custom module file contains only the body. Below you can see an example DataWeave file.

THANK YOU