Collection Functions

Please send corrections and suggestions to cannon@synergfarmsolutions.com. (v3)

Mutated means the collections modifies itself. Keep in mind that you can effectively mutate a collection even if the collection returns a new collection by assigning it to itself like this: $cExample:=$cExample.reverse(). The colored bars are an attempt to visually identify function "pairs". Don't forget about the new For each syntax:

For each (Object ; Collection { ; begin { ; end } }) { Until | While } (Boolean_Expression) }
Can iterate through collections, entity selections, and object properties.

	Result Type	Copy Type	Parameter
Add/Remove Elements			
.push *(element {; element2 ; ... ; elementN}) -> Mutated* Appends one or more elements.	Mutated		
.pop *() -> Element* Removes the last element. The removed element is returned by the function.	Mutated (and element)		
.insert *(index ; element) -> Mutated* Inserts an element at the specified index.	Mutated		
.remove *(index {; howMany}) -> Mutated* Removing a range of elements.	Mutated		
.unshift *(value {; value2 ; ... ; valueN}) -> Mutated* Prepends one or more elements	Mutated		
.shift *() -> Element* Removes the first element. The removed element is returned by the function.	Mutated (and element)		
.resize *(size {; defaultValue}) -> Mutated* Resizes the collection, adding null elements if needed.	Mutated		
.clear *() -> Mutated* Removes all elements.	Mutated		
Combining Collections			
.combine *(col2 {; index}) -> Mutated* Inserts another collection into this collection, either at the end or a specific index.	Mutated		
.concat *(value {; value2 ; ... ; valueN}) -> New collection* Creates a collection from the original collection and with the passed in values appended. Note: if a value being appended is a collection, the *elements* of the collection will be appended.	New collection	Shallow	
Searching			
.query *(queryString {; value}{; value2 ; ... ; valueN}{; querySettings}) -> New collection* Returns a new collection with elements that match the query string.	New collection	Shallow	Query String
.filter *(methodName {; param}{; param2 ; ... ; paramN}) -> New collection* Returns a new collection with elements the method returns true for.	New collection	Shallow	Method
.indices *(queryString {; value}{; value2 ; ... ; valueN}) -> Collection of indices* Returns a collection of indices for elements that match the query string.	Collection of indices		Query String
.indexOf *(toSearch {; startFrom}) -> Index* Returns the index of the first element the toSearch value is found in.	Index		Value (text, number, boolean, date, null, object, or collection)
.lastIndexOf *(toSearch {; startFrom}) -> Index* Returns the index of the last element the toSearch value is found in.	Index		Value (same as indexOf)
.find *({startFrom ;} methodName {; param {; param2 ; ... ; paramN}}) -> Value* Returns the value of the first element the method returns true for.	Value		Method
.findIndex *({startFrom ;} methodName {; param {; param2 ; ... ; paramN}}) -> Index* Returns the index of the first element the method returns true for.	Index		Method

Query String Syntax *(from dataClass.query() documentation)*

attributePath comparator value {logicalOperator attributePath comparator value}

Comparator

Comparison	Symbol(s)	Comment
Equal to	=, ==	Gets matching data, supports the wildcard (@), neither case-sensitive nor diacritic.
	===, IS	Gets matching data, considers the @ as a standard character, neither case-sensitive nor diacritic
Not equal to	#, !=	Supports the wildcard (@)
	!==, IS NOT	Considers the @ as a standard character
Less than	<	
Greater than	>	
Less than or equal to	<=	
Greater than or equal to	>=	
Included in	IN	Gets data equal to at least one of the values in a collection or in a set of values
Not condition applied on a statement	NOT	Parenthesis are mandatory when NOT is used before a statement containing several operators
Contains keyword	%	Keywords can be used in attributes of string or picture type

Logical Operator

Conjunction	Symbol(s)
AND	&, &&, and
OR	|, ||, or

Attribute Path

Examples
"country = :1" ; "Canada"
"country.name = :1" ; "Canada"
"countries[].name = :1" ; "Canada"

	Result Type	Copy Type	Parameter
Sorting			
.orderBy *({criteria}) -> New collection* Returns a new collection with elements ordered by the criteria. Criteria options: • Order by values at a property path (add "**asc**" or "**desc**" to specify sort order) • A collection of criteria objects • ck ascending or ck descending if ordering by scalar values	New collection	Shallow	Property Path (and other options)
.orderByMethod *(methodName {; extraParam}{; extraParam2 ; ... ; extraParamN}) -> New collection* Returns a new collection with elements ordered by the method.	New collection	Shallow	Method
.sort *({methodName {; extraParam}{; extraParam2 ; ... ; extraParamN}}) -> Mutated* Sorts the elements using the method.	Mutated		Method
Converting to New Collection			
.copy *({ck resolve pointers}) -> New collection* Returns a copy of the collection.	New collection	Deep	
.reverse *() -> New collection* Returns a copy of the original collection, but with all its elements order reversed.	New collection	Deep	
.slice *(startFrom {; end}) -> New collection* Returns a copy of a range of elements from the original collection. The end value can be negative to count backward from the end.	New collection	Shallow	
.distinct *({propertyPath}{;}{ck diacritical}) -> New collection* Creates a collection containing only distinct elements of the collection or at a specific property path. Note: Only non-null elements are considered. The new collection will be sorted.	New collection	Shallow	Property Path
.extract *(propertyPath {; targetPath}{; propertyPath2 ; targetPath2 ; ... ; propertyPathN ; targetPathN}{; ck keep null}) -> New collection* Creates a new collection with values from specific property paths extracted from the original collection. Optionally, you can change the property name during the extraction as well. Note: the targetPath can target a different property path level. Parent objects will automatically be created as needed.	New collection	Deep	Property Path
.map *(methodName ; param {; param2 ; ... ; paramN}) -> New collection* Creates a collection that contains the values returned by the method after processing each element in the original collection.	New collection	Depends on method ?	Method
.reduce *(methodName {; initValue}{; param}{; param2 ; ... ; paramN}) -> Accumulated value* Provides a way for a method to be called for each element in the collection, each time modifying the accumulated value. The accumulated value can be thought of as a local variable that is initialized with the initValue parameter and exists while the function is running. This value is then returned at the end. The accumulated value can be of type Text, Number, Object, Collection, Date, or Boolean.	Value		Method
Boolean Logic Inspection			
.count *({propertyPath}) -> Real value* Returns the number of non-null elements in the collection or at a given property path.	Real		Property Path
.countValues *(value {; propertyPath}) -> Long value* Returns the number of times a specific value exists in the collection or at the given property path. Note: when looking for an object or collection, it is doing so based on reference.	Long		Text, Number, Boolean, Date, Object, or Collection and Property Path
.every *({startFrom ;} methodName {; param {; param2 ; ... ; paramN}}) -> Boolean* Returns true if the called method returns true for **all** the elements in the collection. The test can be limited to a range if startFrom is used and if the method terminates the scan early.	Boolean		Method
.some *({startFrom ;} methodName {; param {; param2 ; ... ; paramN}}) -> Boolean* Returns true if the called method returns true for **any** the elements in the collection. The test can be limited to a range if startFrom is used and if the method terminates the scan early.	Boolean		Method
Math			
.sum *({propertyPath}) -> Real value* Returns the sum of any numeric values in the top level of the collection or at a given property path.	Real		Property Path
.average *({propertyPath}) -> Real value* Returns the average of any numeric values in the top level of the collection or at a given property path.	Real		Property Path
.min *({propertyPath}) -> Boolean, Text, Number, Collection, Object, or Date* Returns the lowest value in the top level of the collection or at a given property path.	Boolean, Text, Number, Collection, Object, or Date		Property Path
.max *({propertyPath}) -> Boolean, Text, Number, Collection, Object, or Date* Returns the highest value in the top level of the collection or at a given property path.	Boolean, Text, Number, Collection, Object, or Date		Property Path
Miscellaneous			
.equal *(collection2 {; ck diacritical}) -> Boolean* Returns true if the collections are equivalent using a deep comparison.	Boolean		
.fill *(value {; startFrom {; end}}) -> Mutated* Replaces a range of elements with the passed in value. The end value can be negative to count backward from the end.	Mutated		Value
.join *(delimiter {; ck ignore null or empty}) -> String* Creates a string that is the concatenation of all the elements of the collection, separated by the delimiter. Non-string values are converted to strings. Object are converted to "[object Object]".	String		