Advanced Velocity Workshop
Locator & Query Tools

Locator Tool	

Invoked with *$_.locate()* or *$_.locate* followed by the asset type in camelcase.
 	
There is currently a known bug with .locateReference(). Workaround is to use the non-explicit method and pass in the data type as a parameter instead.
	`$_.locate()`
	`$_.locate`**`Page`**`()`
	`$_.locate`**`File`**`()`
	`$_.locate`**`Block`**`()`
	`$_.locate`**`Folder`**`()`
	`$_.locate`**`Symlink`**`()`
	`$_.locate`**`Reference`**`()*`

Parameters
The first parameter is always the path to the asset. When using the invocation method where type isn't explicit, the asset type must be passed in as the second argument. The site name is always the last parameter and is optional. When a sitename is not provided, the site is presumed to be the site in which the current page is located.
	`$_.locate[Asset Type](PATH, SITENAME)`
	`$_.locate[Asset Type](PATH)`
	
	Example
	`$_.locatePage("path/to/page", "sitename" )`
	
	Non-explicit method
	`$_.locate(PATH, TYPE, SITENAME)`

Built-in Variables	As of 7.4
	`$currentPage`
	`$currentPagePath`
	
	As of 7.10.1
	`$currentPageSiteName`

Property Tool	

The outputProperties() method displays the properties and methods of an object.
 	
The result will be displayed in a list in which properties and methods are on the left and the expected returned output of each are listed on the right. For example, the property "link" will be returned as a string.
	Example
	$_PropertyTool.outputProperties($currentPage)
	
	Output Snippet
	`Properties:`
	`- link: String`
	`- parentFolder: Folder`
	`- metadata: Metadata`

isNull() is a method that checks for nullity and returns a boolean value.	Example `$_PropertyTool.isNull($currentPage.metadata.title)`

Query Tool
To begin a query, create a query object with the *query()* constructor and assign it to a variable.
 	
After all filtering and search options have been specified, you must execute the query to capture the results.
	Creating a Query Object
	`#set($queryObject = $_.query())`
	
	Executing a Query
	`#set($results = $queryObject.execute())`

Queries are done either by metadata set or content type and the respective path must be specified.	`$queryObject.byMetadataSet("Default")`
	
	`$queryObject.byContentType("News Article")`

Queries can include non-publishable and/or non-indexed assets.
 	
You can explicitly specify what type of assets should be included in the search by using the respective include method. The method is invoked with *.include* followed by the asset type in plural.
 	
Limit the number of returned assets with the maxResults() method which takes in an integer value. If not specified, 100 is the default value. There is a maximum limit of 500. Results will only return the first 500 assets even if there are more.
 	
To specify another site to search in, use the siteName() method.
 	
To search across all sites, use the searchAcrossAllSites() function.
	`$queryObject.pubishableOnly(BOOLEAN)`
	`$queryObject.indexableOnly(BOOLEAN)`
	
	`$queryObject.include Pages(true)`
	`$queryObject.include {Asset Type}s(BOOLEAN)`
	
	`$queryObject.maxResults(25)`
	
	`$queryObject.siteName("Advanced Velocity Workshops")`
	
	`$queryObject.searchAcrossAllSites()`

Sorting can be done using the sortBy() method with one of the following criteria: a static metadata field (title, startDate, etc),last modified date, creation date, name, or path.
 	
The sortDirection() method takes in either "asc" or "desc" as string arguments.
	`$queryObject.sortBy("displayName")`
	
	`$queryObject.sortDirection("desc")`

* Note that filter and sort methods can be chained for cleaner or more concise coding using the Java dot notation.

HashMaps	
HashMaps are associative arrays that contains pairs of keys and values. HashMaps cannot contain duplicate keys and values can be accessed by their respective key. A HashMap can be initialized with empty curly brackets. Data can be added by using the put() method that takes in a key and a value as parameters.	`#set($map = {})` `#set($temp = $map.put(KEY, VALUE))`
A value of a pair can be retrieved using its key with the get() method. Alternatively, dot notation can be used.	`$map.get(KEY)` `$map.KEY`
When the key is arbitrary and captured in a variable, use square brackets to retrieve its counterpart.	`$map.[$varToKey]`
keySet() returns all keys in a map and values() returns all values of a map.	`$map.keySet()` `$map.values()`

When iterating through a HashMap using the foreach directive, the looping variable is the value of the current iteration of the loop.	`#foreach($value in $map)`
	`  $value`
	`#end`
	
	`$value → The value at the current index.`

It is possible to check if a key or value is in a HashMap by using the respective contains method. These two methods are invoked with *contains* followed by the term *Key* or *Value*. These methods return a boolean value.	`$map.containsKey(KEY)` `$map.containsValue(VALUE)`