Advanced Velocity Cheatsheet

Tools and API's Unique to Cascade CMS

Locator Tool	

Invoked with *$_.locate()* or *$_.locate* followed by the asset type in camelcase. *Does not need to be bolded.*	`$_.locate()`
	`$_.locate`**`Page`**`()` `$_.locate`**`Folder`**`()`
	`$_.locate`**`File`**`()` `$_.locate`**`Symlink`**`()`
	`$_.locate`**`Block`**`()` `$_.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.
	`$_.locate[Asset Type](PATH)`
	`$_.locate[Asset Type](PATH, SITENAME)`
	
	Example
	`$_.locate`**`Page`**`("path/to/page", "sitename")`
	
	Non-explicit method
	`$_.locate(PATH, TYPE, SITENAME)`

Built-in Variables (as of 7.10.1)	`$currentPageSiteName - returns a string`
	`$currentPagePath - returns a string`
	`$currentPage - returns an object`

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.	`$_PropertyTool.isNull($currentPage.metadata.title)`
.isEmpty() takes a string and determines first if it is null and then if it's empty.	`$_PropertyTool.isEmpty(STRING)`

Advanced Velocity Cheatsheet

Tools and API's Unique to Cascade CMS

Query API	

To initiate 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. Since version 8.9, there is a hard maximum limit of 2000.
 	
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 uses the .sortBy() method with one of the following criteria: name, path or either a user- or system-metadata field (title, startDate, last modified date, creation date, etc.).
 	
The .sortDirection() method takes in either "asc" or "desc" as string arguments.
	`$queryObject.sortBy("displayName")`
	
	`$queryObject.sortDirection("desc")`

* Using the Java dot notation, filter and sort methods can be chained for cleaner, more concise, coding.

Advanced Velocity Cheatsheet

Tools and API's Unique to Cascade CMS

HashMaps (standard Velocity code)	

HashMaps are associative arrays that contain pairs of keys and values. HashMaps cannot contain duplicate keys. 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))`
	
	Keys and values can be strings, numbers or variables.
	
	Setting "$temp" is used to prevent "$map.put" from displaying on a page.

The value of a pair can be retrieved using its key with the .get() method or using dot notation.	`$map.get(KEY)` *-works with strings, numbers or variables.* `$map.KEY` *-works only with strings.*
When the key is arbitrary and captured in a variable, use square brackets to retrieve its counterpart value.	`$map[$varToKey]`
.keySet() returns all keys and .values() returns all values of a map as an array.	`$map.keySet()` `$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. Both methods return a boolean value.	`$map.containsKey(KEY)`
	
	`$map.containsValue(VALUE)`