ArangoDB CHEAT SHEET

Starting & accessing	
arangod /path/to/my/db	start server
arangod --console --log error /path/to/my/db	start emergency console (do not use with a db which has a server attached to it!)
http://localhost:8529/_admin/html/index.html	access admin front end in browser
arangosh	start ArangoDB shell

arangod frequently used options	
--log	set log level: error, warning, info, debug, trace
--server.endpoint *protocol://host:port*	set address and port for HTTP clients (e.g. *tcp://localhost:8529*, *ssl://localhost:8529*)
--daemon	run as daemon/background process

Database management methods in arangosh	
db._createDatabase(*database-name*)	create database
db._dropDatabase(*database-name*)	drop a database
db._useDatabase(*database-name*)	change into an existing database
db._listDatabases()	list all databases

Collection management methods in arangosh	
db._create(*collection-name*, *properties*)	create collection (with optional *properties*)
db._createEdgeCollection(*collection-name*, *properties*)	create an edge collection
db._collection(*collection-name*|*collection-id*)	get collection
db._collections()	list all collections
db.*collection-name*	get a collection by name
db._drop(*collection-name*|*collection-id*)	drop collection with indexes
db._truncate(*collection-name*|*collection-id*)	remove collection, keep indexes

Collection methods in arangosh	
collection.drop()	drop collection with all data and indexes
collection.truncate()	remove all documents, keep indexes
collection.properties()	get collection properties
collection.properties(*properties*)	change collection properties
collection.figures()	get collection figures (disk space etc.)
collection.load()	load collection into memory
collection.unload()	start to unload a collection
collection.rename(*new-name*)	rename collection to new-name

Document methods in arangosh	
collection.document(*document*)	get document by identifier
collection.save(*data*)	create new document
collection.replace(*document*, *data*)	replace existing document
collection.update(*document*)	partially update
collection.remove(*document*)	remove document
db._document(*document*|*document-handle*)	get document by identifier|handle
db._replace(*document*|*document-handle*,*data*)	replace existing document
db._update(document)	partially update document
db._remove(*document*)	remove document

Edges in arangosh	
edge-collection.save(*from*, *to*, *document*)	save new edge
edge-collection.edges(*vertex*)	find edges from (outbound) to (inbound) vertex
edge-collection.inEdges(*vertices*)	find all edges ending in (inbound)
edge-collection.outEdges(*vertices*)	find all edges starting from (outbound)

Queries in arangosh	
db._query(*query*).toArray()	run an ad-hoc AQL query
collection.all()	select all documents and return cursor
collection.any()	select a random document
collection.byExample(*example*)	select all documents that matches the given *example*
collection.firstExample(*example*)	select the first document that matches the given *example*
collection.range(*attribute*, *left*, *right*)	select all documents with *attribute* >= *left* and < *right*
collection.removeByExample(*example*)	remove all documents that match the example
collection. replaceByExample (*example*, *newValue*)	remove all documents that match the example
collection. updateByExample (*example*, *newValue*)	remove all documents that match the example
collection.count()	return the number of documents
collection.toArray()	convert the collection into an array of documents (might be big, not for production!)

Geo Queries in arangosh	
collection.near(*latitude*, *longitude*)	get documents near the given coordinates
collection.within(*latitude*, *longitude*, *distance*)	get all documents within a radious of *distance* around the given coordinates
collection.geo(*location*)	the next near or within operator will use the specific geo-spatial index

Sequential Access And Cursors in arangosh	
cursor.hasNext()	returns true if the cursor still has documents
cursor.next()	advance cursor
cursor.dispose()	free resources associated with a cursor
cursor.count()	returns number of documents in the result set