C3 Console Cheat Sheet

C3 Console is very important to interacting with the C3 Platform, both for Application Developers and Data Scientists. This cheat sheet summarizes useful C3 Console commands.

EXPLORE TYPES

Explore the types you have in your environment

```
> c3ShowType(MyType)
```

Shows type documentation on MyType

```
> c3Grid(MyType)
```

Shows MyType type definition

```
> c3Viz(MyType)
```

Shows MyType type definition

CREATE OBJECTS

Create new variables and upsert to environment

```
> var newObject = MyType.create({fields})
> newObject.field = "value"
```

Creates a new object and gives values to its fields

```
> newObject.merge()
```

Only updates the fields that are different in environment

```
> newObject.upsert()
```

Completely replaces the object in environment

QUERY TYPE

Run queries on types

```
> MyType.evaluate(spec)
> MyType.evaluate({projection: "field1,
  count()", group: "field1", order:
  "descending(count)m filter: "filter ==
  'condition', limit: -1})
```

Run a query on a type

FETCH, GET, & COUNT OBJECTS

Retrieve one or multiple instances of an object

```
> var newObject = MyType.get("some_id")
> newObject.get("another_field")
```

"Get" retrieves a single object

```
> MyType.fetch({include,filter,order,limit})
> MyType.fetchObjStream({include,ids,order,
  limit})
```

"Fetch" retrieves multiple instances

```
> MyType.fetchCount()
> c3Count(MyType)
```

Count number of instances

REMOVE OBJECTS

Remove one or more objects from environment

```
> MyType.remove("some_id")
```

Remove: delete one single object

```
> MyType.removeAll("some_filter")
```

RemoveAll: deleted all objects determined in filter: Careful !

REFRESH CALC FIELDS

Refresh calculated fields on a type

```
> MyType.refreshCalcFields(spec)
> MyType.refreshCalcFields()
```

Refresh calc fields defined in a spec or all calc fields

METRICS & EXPRESSIONS

Write complex expressions and handle metrics

```
> c3ShowType(MetricEvaluatable)
```

Shows type documentation on MetricEvauatable Type

```
> c3ShowType(ExpressionEngineFunction)
> c3ShowFunc(ExpressionEngineFunction)
```

Shows documentation on all expression functions

```
> MyType.evalMetrics({ids, expressions,
  start, end, interval})
```

Evaluate metrics defined in platform

```
> SimpleMetric.make({id, name, srcType, path,
  expression})
```

Make a simple metric on the fly

```
> CompoundMetric.make({id, name, expression})
```

Make a compound metric on the fly

```
> MyType.evalMetricsWithMetadata({spec,
  overrideMetrics})
```

Evaluate metrics defined on the fly

```
> MyType.listMetrics()
```

Lists all metrics that are available on MyType

MAP REDUCE JOB

Run and handle MapReduce jobs on the platform

```
> var job = MapReduceJobType.make({batchSize,
  limit, include, start, end, interval, id})
```

Make a MapReduce job

```
> job = MapReduceJobType.create(job)
```

Create the MapReduce job

```
> MapReduceJobType.start(job)
> job.start()
```

Start a MapReduce job

```
> MapReduceJobType.status(job)
> job.status()
```

Check the status of your MapReduce job

NORMALIZATION ENGINE

Handle Normalization on the platform

```
> c3ShowType(Ann.Ts)
> c3ShowType(Treatment)
> c3ShowType(Interval)
```

Shows doc on normalization annotation, treatment, interval, and timedDataFields options

```
> c3ShowType(TimedDataField)
```

Shows doc on timedDataFields options

```
> MySeriesHeader.normalizeTimeSeries({id}
> MySeriesHeader.refreshNormalization({ids:
  "array_of_ids", async: false})
```

Trigger normalization manually synchronously

```
> MySeriesHeader. refreshNormalization({ids:
  "array_of_ids"})

> Tag.rebuild(Tag.get("Your_Tag"),
   agRebuildSpec.make({normalizedtimeseries:
  true}))
```

Trigger normalization manually asynchronously

ANALYTICS

Handle Analytics on the platform

```
> c3ShowType(AnalyticsContaier)
```

Shows type documentation on AnalyticsContainer

```
> c3Grid(TenantConfied.fetch())
```

Shows current configuration

```
> AnalyticsContainer.fireAnalytics(args)
```

Invoke analytics manually synchronously, to use for testing

```
> AnalyticsContainer.invalidateSources(args)
```

Invoke analytics manually synchronously

```
> AnalyticsQueue.invalidateSources(args)
```

Invoke analytics manually asynchronously

MONITOR CLUSTER

Monitor what's going on in your cluster

```
> c3Grid(c3Context())
```

Print current tenant / tag

```
> c3Grid(Cluster.hosts())
```

Number of worker nodes

```
> c3Grid(Cluster.actionDump())
```

See actions in environment

MONITOR QUEUES 1/2

Monitor all queues in your environment

```
> c3Grid(InvalidationQueue.countAll())
```

Monitor al job statuses in your environment

```
> c3MonitorQueues()
```

Perform continuous monitoring of the queues

```
> c3StopMonitorQueues()
```

Stop continuous monitoring of the queues

MONITOR QUEUES 2/2

Monitor all queues in your environment

```
> MyQueue.pause()
```

Pause a queue

```
> MyQueue.resume()
```

Resume a queue

```
> c3QReport(MyQueue)
```

Report initialized, pending, computing, and failed actions in a queue

```
> c3QErrs(MyQueue)
```

Display error messages for failed tasks in a queue

```
> MyQueue.recoverFailed()
```

Recover error messages for failed tasks in a queue

```
> c3QErrorsClear(MyQueue)
```

Clear all errors

```
> MyQueue.recoverStuck()
```

If actions seem to be stuck

