Introduction to MLJS

What is MLJS?

What is MLJS?

- JavaScript wrapper for the REST API in MarkLogic V6 & 7
- Removes a lot of the repetitive, difficult work associated with learning the REST API and MarkLogic inner magic
- Only requires the skills of a JavaScript + HTML + CSS web developer
- Works in the browser, or within Node.js for server type applications (E.g. Instant Messaging integration)

MarkLogic

MLJS Benefits

- Quick and easy to learn
- Minimal code to create an application
- Not just search - you can also create and modify documents and triples
- Lots of samples you can copy/paste/customise
- Detailed API documentation
- No knowledge 'cliff' where things are easy until you want to do something slightly different, then it becomes very difficult
- Core underlying API with many widgets that can work together or independently
- Recent demo builds took 2-3 days rather than 2-3 weeks

MarkLogic

MLJS downsides

- JavaScript
- Development from the start - albeit easy
  - Although the 'Workplace' will solve this

- JavaScript

- Browser apps limited to being hosted within MarkLogic server only
  - Although W3C CORS support will solve this (Authenticated XSS)

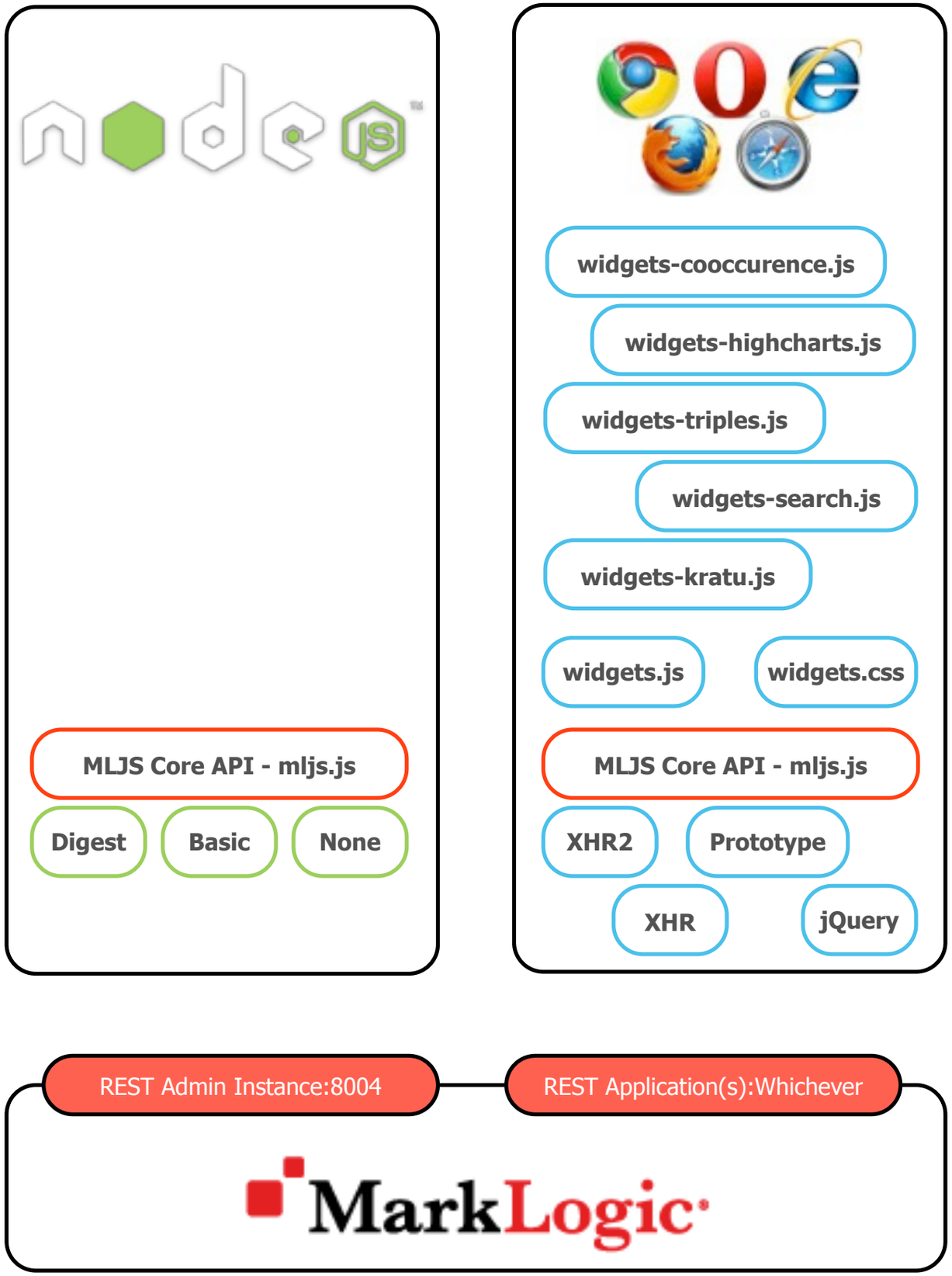
- JavaScript

MarkLogic

Little known facts

- Most JavaScript libraries do NOT provide a Digest authentication capability - MLJS does provide Digest Auth
- By choosing one library (E.g. jQuery) and needing another down the road (E.g. Prototype), the libraries can clash without special effort. MLJS does not cause API conflicts
- Most libraries can only be used in either Node.js or the browser. MLJS Core can be used in both Node.js and the browser
- Extending AppBuilder requires often knowing XSLT + XQuery + HTML + CSS + JavaScript, plus its own foldering layout. Extending Roxy requires often knowing XQuery + HTML + CSS + occasionally JavaScript, plus its own foldering layout. Extending MLJS requires only knowing JavaScript + HTML + CSS, and a couple of JavaScipt API hooks - like any web developer can
- Learning the REST API directly is hard, and using even jQuery to communicate with it brings up bugs with each new call. Using MLJS requires just knowing one or two JavaScript API calls.

MarkLogic



Widget Libraries
- potentially multiple widgets per file

Widgets baseline

MLJS Core

Communication
& Security Wrappers

MarkLogic Server
Versions 6 & 7

MLJS Core

What the MLJS object provides

REST API Instance Administration

create()
destroy()
exists() (aka test())

Document Operations

get()
metadata()
properties()
saveProperties()
save()
merge()
replaceProperty()
delete() (aka remove())

Search

collect()
list()
keyvalue()
search()
searchCollection()
structuredSearch()
  (aka structuredQuery())
saveSearchOptions()
values()
valuesCombined()
subcollections()
indexes()

Extension

do()

Semantics

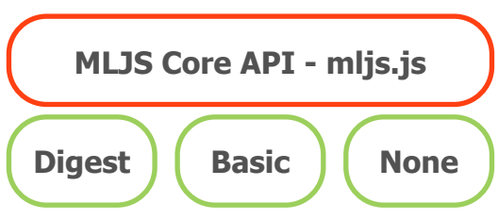
saveGraph()
mergeGraph()
graph()
deleteGraph()
sparql()

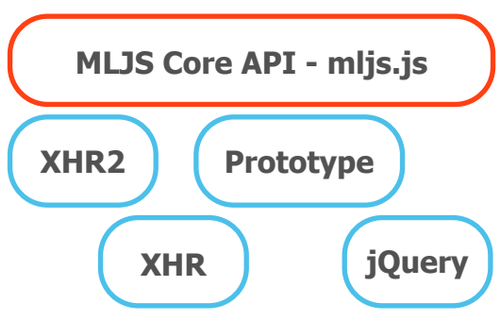
Transactions

begin()
commit()
rollback()

Context Management

createOptions()
createQuery()
createSemanticContext()
createTripleConfig()
createSearchContext()
createDocumentContext()





MarkLogic

MLJS Core provides

- HTTP Connection Management
  - Abstracts Digest, Basic and no authentication. E.g. handles MD5 sums and next request id increments (Digest auth to prevent replay attacks)
- Request wrappers
  - One or more function calls per REST endpoint. E.g. different uses of /v1/search - basic grammar query, keyvalue query, all docs in a collection
- Extensibility
  - do() function provides a generic way to perform your own request wrapper call without having to edit the mljs.js library
- Context objects
  - Co-ordination objects so you don't manually have to keep other objects in sync
- Utility Objects
  - Query builder, Options builder and Triple Config objects
- Helper functions
  - Converting text to/from XML/JSON, logging & event handlers

MarkLogic

HTTP Connection Management

- In a Node.js context
  - Plug in either No authentication, HTTP Basic or HTTP Digest
  - Maintenance of connection handled transparently between requests
  - No need to consider this for any REST API calls in your code
- In a browser context
  - Pluggable REST AJAX connection objects
  - XMLHttpRequest2 (ECMAScript 3), XMLHttpRequest, jQuery and Prototype
  - XMLHttpRequest2 is most tested, and faster and less buggy than jQuery. Supported by all modern browsers

MarkLogic

Context Objects in MLJS Core

- Provides co-ordination and management of state between calls to MLJS and acting on information returned from MLJS calls
  - Many objects/widgets can be linked to a single context object. Multiple context objects can be live in a single page (they are NOT Singletons)
- Search Context
  - Listens for search term or facet selection changes, and calls search() or structuredSearch(). Sends results to updateResults() and updateFacets() etc in listening objects
- Document Context
  - Maintains information on a single document. Fetches and set document properties, updating dependant widgets when an edit occurs
- Semantic Context
  - Provides Sparql executing utility functions. Finds all facts on a subject, all subjects that match a query, all documents related to subjects (via linked search context) - wraps the sparql() call. Calls updateFacts() and updateSubjects() and updateSubject() in listening widgets

MarkLogic

Context Object Example

MLJS Core API - mljs.js

```
// create db connection
var db = new mljs();

// create common search context
var ctx = db.createSearchContext();

// create widgets
var bar = new com.marklogic.widgets.searchbar();
var results = new com.marklogic.widgets.searchresults();
// ... and so on ...

// register
ctx.register(bar);
ctx.register(results);
```

Widgets API - widgets-search.js

The Core API contains the searchcontext object. This is linked to a particular MLJS connection instance. Creating new widgets and linking them to this class means each widget becomes interdependent, and doesn't need to know how to use mljs.search() itself.

Multiple searchcontext instances can be used in a single page, allowing for sophisticated dashboard-like pages to be created.

```
bar.setSearchContext(this);
results.setSearchContext(this);

// also introspects widgets to see what events they respond to
// - checks for updateResults, updateFacets, updateSort,
//    updatePage, updateSimpleQuery, updateOptions

// Also listens to events generated by widgets
// - calls addSortListener, addFacetSelectionListener, addPageListener
```

```
// search (can be done manually via search bar)
ctx.doSimpleQuery("some query text");
```

```
results.updateResults(true); // show 'loading...'

bar.updateResults(resultsJson);
results.updateResults(resultsJson);
```

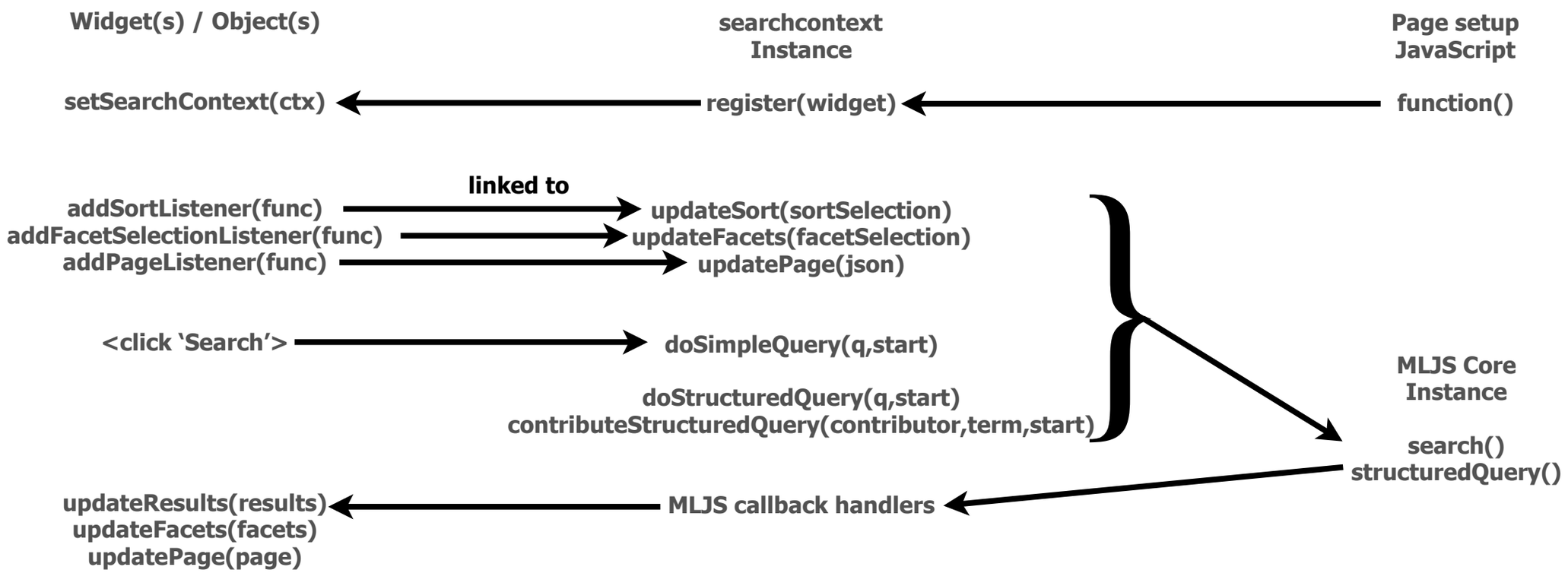
```
ctx.updateSimpleQuery("wibble");
```

User types 'wibble' and clicks search in search bar widget

```
ctx.doSimpleQuery("wibble");
```

```
results.updateResults(true); // show 'loading...'

bar.updateResults(resultsJson);
results.updateResults(resultsJson);
```

How Search Context works



MarkLogic

Utility objects

- Options Builder - db.createOptions()
  - Chained query options builder - abstracts REST API's JSON format, replacing it with simple function calls
- (Structured) Query Builder - db.createQuery()
  - For creating complex queries. E.g. and(), or(), geospatial, word, element value, json key. Abstracts the REST API's JSON structured query format
- Triple Config - db.createTripleConfig()
  - Understands programmatic ontology. E.g. valid triples between a FOAF Person and Organisation, what properties each of these can have, and what type they are (E.g. integer). Used for drawing sparqlbar query builder and determining the 'name' of an Entity. (E.g. use #name for a FOAF Person as the display name)
  - No support for importing OWL or similar definitions... yet... but you can plug your own in via JavaScript

MarkLogic

Utility object example

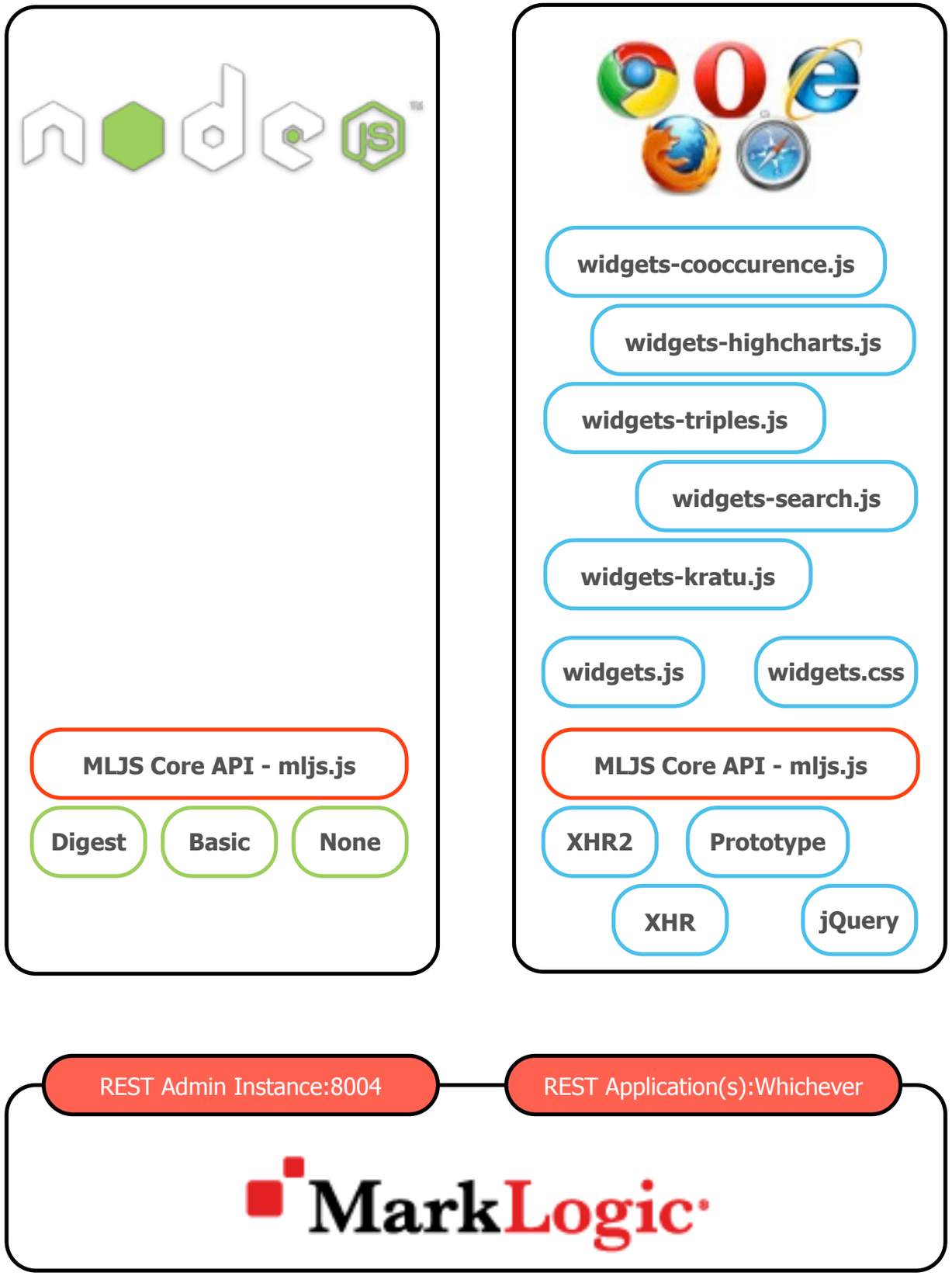
Options Builder supports ALL configuration settings of a JSON Search Options object

```
var ob = db.createOptions();

ob.defaultCollation("http://marklogic.com/collation/en")
  .collectionConstraint() // default constraint name of 'collection'
  .rangeConstraint("animal",["item-order"]) // constraint name defaults to that of the range element name
  .rangeConstraint("family",["item-frequency"]) // constraint name defaults to that of the range element name
  .rangeConstraint("actor",["item-frequency"],"http://marklogic.com/collation/")
  .rangeConstraint("year",["item-order"],"http://marklogic.com/collation/")
  .rangeConstraint("city",["item-order"],"http://marklogic.com/collation/")
  .rangeConstraint("month",["item-order"],"http://marklogic.com/collation/")
  .rangeConstraint("Title","title","http://www.w3.org/1999/xhtml","xs:string","http://marklogic.com/collation/",true)
  .rangeConstraint("Heading","h1","http://www.w3.org/1999/xhtml","xs:string","http://marklogic.com/collation/",true);
```

(Structured) Query Builder enables easy creation of a search

```
var qb = db.createQuery();

qb.query(
  qb.and(
    [qb.collection("testdata"),qb.collection("temperatures")]
  )
);

var query = qb.toJson();
```


MarkLogic



Widget Libraries
- potentially multiple widgets per file

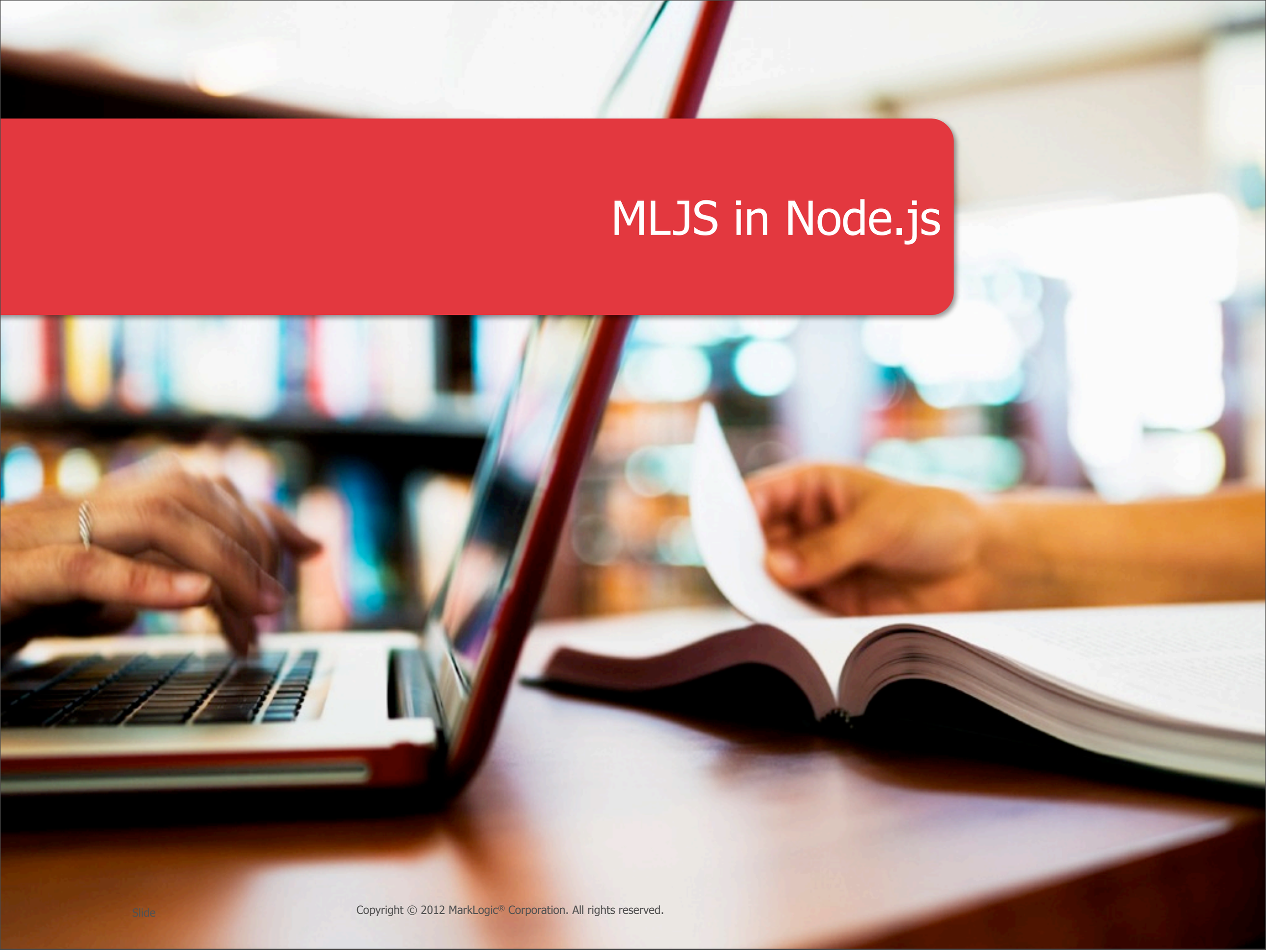
Widgets baseline

MLJS Core

Communication
& Security Wrappers

MarkLogic Server
Versions 6 & 7

MLJS in Node.js



Why MarkLogic and NodeJS?

- I had a need for an alerting application
  - Speed very important
  - Variety of portable hardware

- Wanted to use HTML5 and WebSockets
  - WebSockets is an open connection between a browser and a HTTP server that messages can be sent on in either direction at any time
  - Perfect for alerting applications (instead of polling every x seconds)
  - MarkLogic HTTP server doesn't have WebSockets

- Decided to use NodeJS as an intermediary server
- For non alerting part, decided to use REST API - E.g. Search, document display

MarkLogic

Why NodeJS?

- Very light weight architecture for building server processes
- Event driven - non blocking IO
- Write everything in JavaScript

```
1  var http = require('http');
2  http.createServer(function (req, res) {
3    res.writeHead(200, {'Content-Type': 'text/plain'});
4    res.end('Hello World\n');
5  }).listen(1337, '127.0.0.1');
6  console.log('Server running at http://127.0.0.1:1337/');
```


MarkLogic

MLJS and Node.js

- Found using the REST API directly was repetitive, with lots of complex duplicated code, and typo type bugs for each new call
- MLJS originally designed as a wrapper for the REST API within Node.js - browser functionality only added months later
- Used in demo applications
  - Situational awareness - performing searches for a HTML5 + WebSockets SA application - e.g. showing previously found IEDs in an area, intel reports, or weapons/vehicle field manuals
  - Instant Messaging - used to provide text interaction to a MarkLogic 'bot' on a Jabber IM network. Type commands to perform a search, and have the results IM'ed to you

MarkLogic

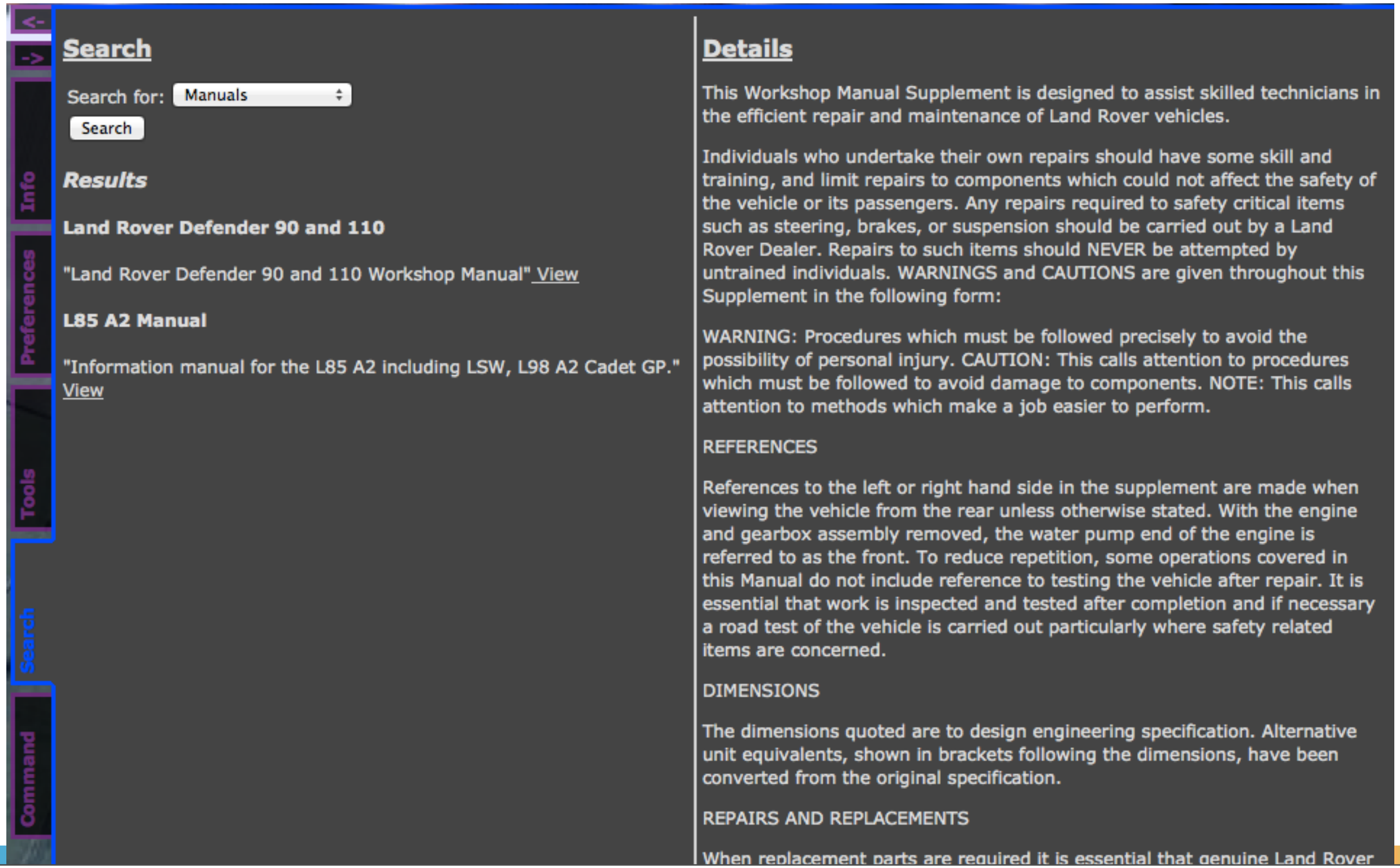
What can be built with MLJS?

- Polygon Geospatial search for IEDs, alerts for troop movements

MarkLogic

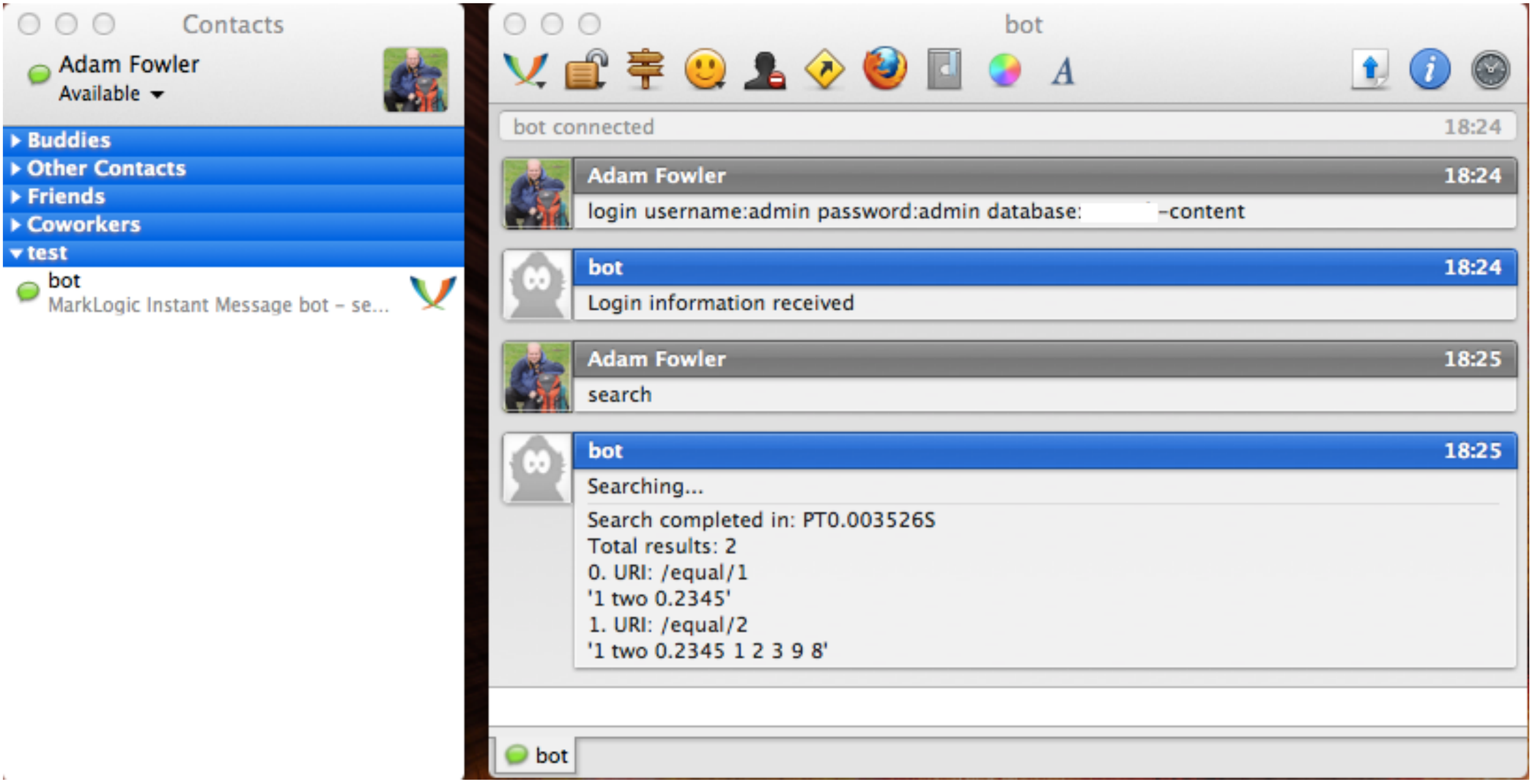
What can be built with MLJS?

- MLJS search functionality to find and display documents via JSON



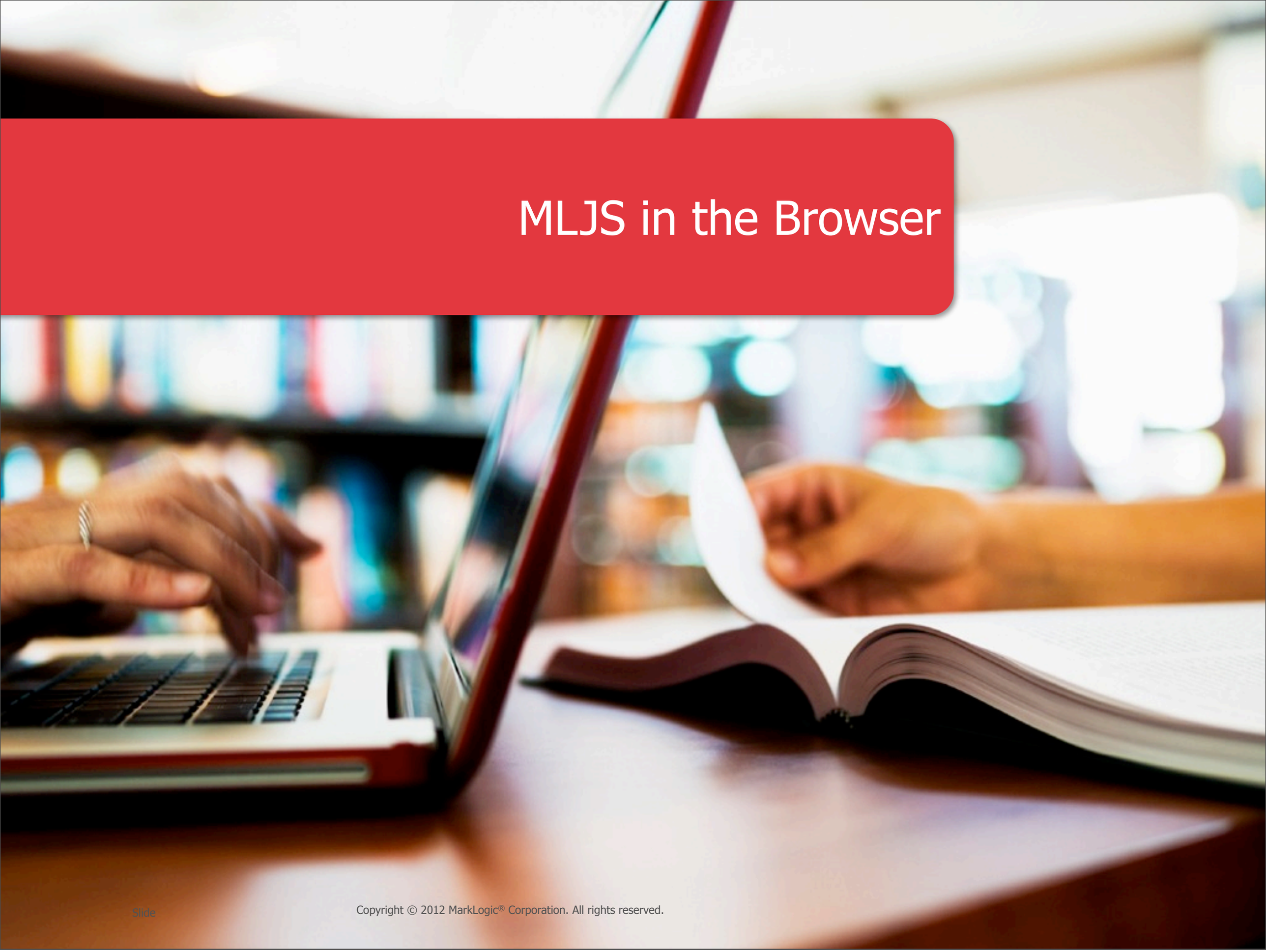
What can be built with MLJS?

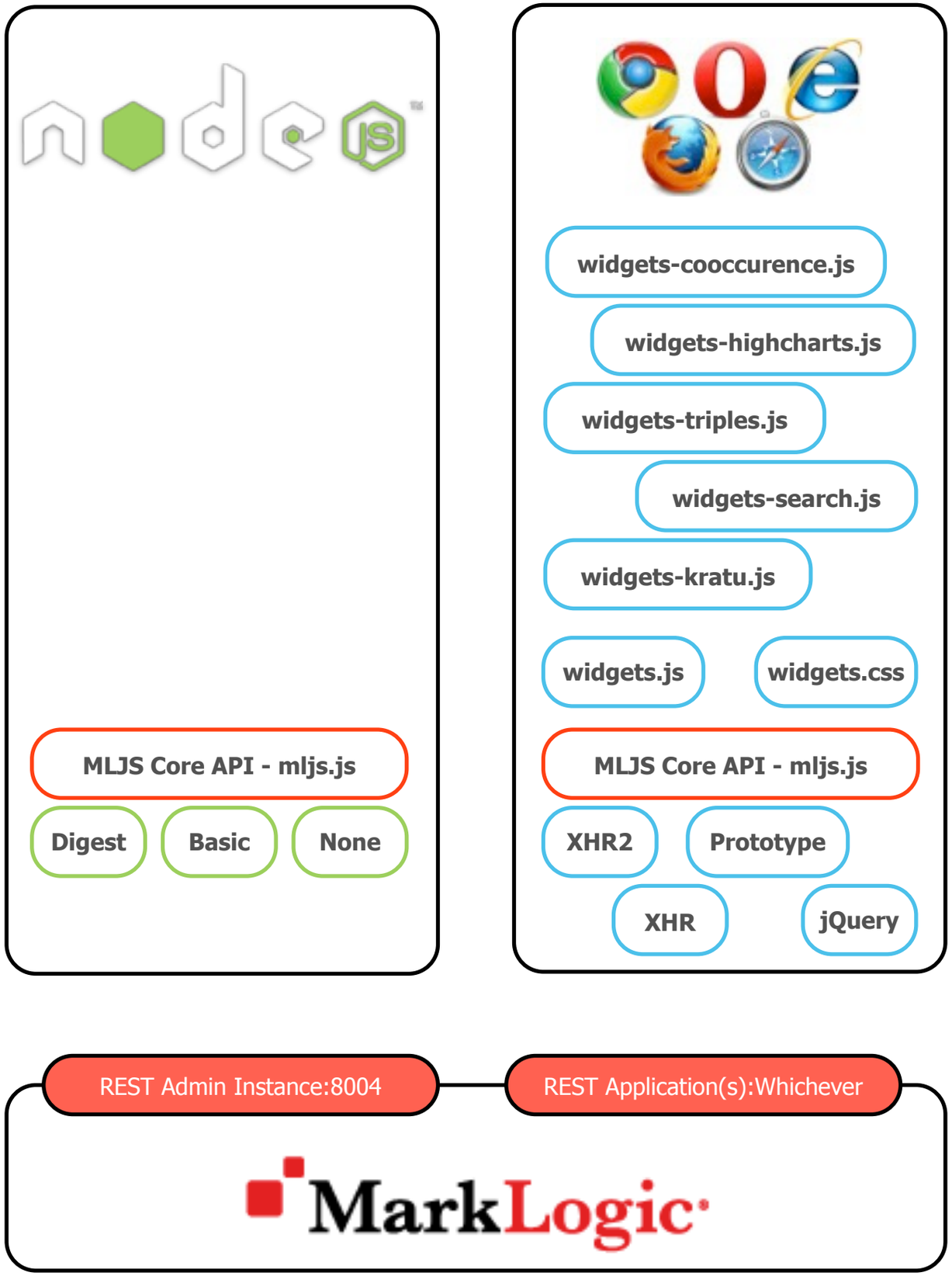
- Jabber based alerting and search application server



MarkLogic

MLJS in the Browser





Widget Libraries
- potentially multiple widgets per file

Widgets baseline

MLJS Core

Communication
& Security Wrappers

MarkLogic Server
Versions 6 & 7

22 Widgets currently available

Search	Triples	Data Exploration	High Charts	Co-occurence	Kratu
searchpage searchbar searchresults searchpages searchsort searchfacets	sparqlbar sparqlresults entityfacts	graphexplorer	highcharts	cooccurence	kratu

collections	rdb2rdf	DLS	Markings	Doc Builder	Documents
collectionuris	rdb2rdf	dlscollections dlscollection	markings	create	docproperties docheadviewer docviewer

MarkLogic

MLDB JavaScript Search

Browse

X Collection: Animals

Collection

Animals (14)

Animal

Dog (3)
Homosapien (3)
Hamster (2)
Cat (1)
Gerbil (1)
More...

Family

Pet (8)
Marklogician (3)
Bird (2)

Search: collection:"animals" Search

Showing 11 to 14 of 14 << < Page 2 of 2 > >> Sort: None

None
Animal
Family
Actor
Year
City
Month

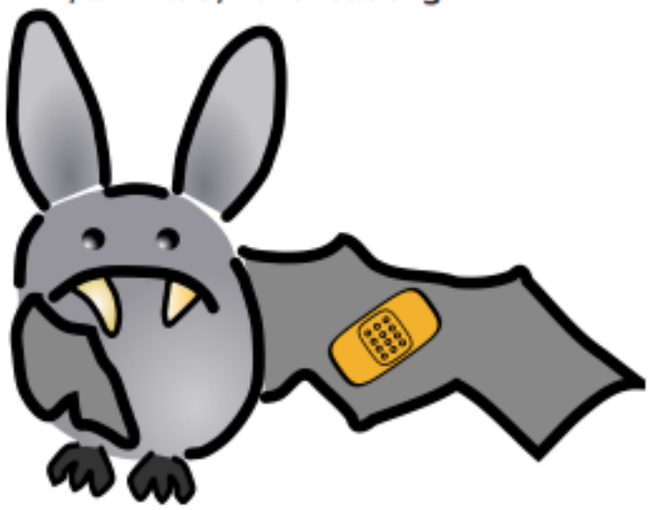
Results

11. Eric the MarkLogician
I am Eric the MarkLogician. A homosapien fact is: Sir Eric of the Community

12. Olly the Ostrich
I am Olly the Ostrich. A ostrich fact is: Tasty and lean

13. Polly the Penguin
I am Polly the Penguin. A penguin fact is: Penguins are awesome

14. /animals/hurt-bat.svg

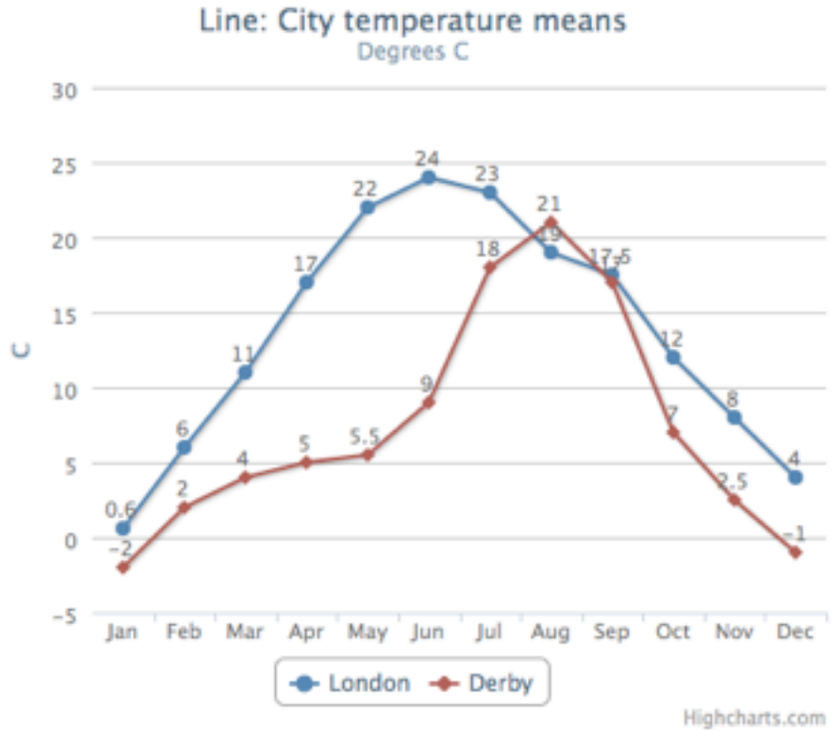


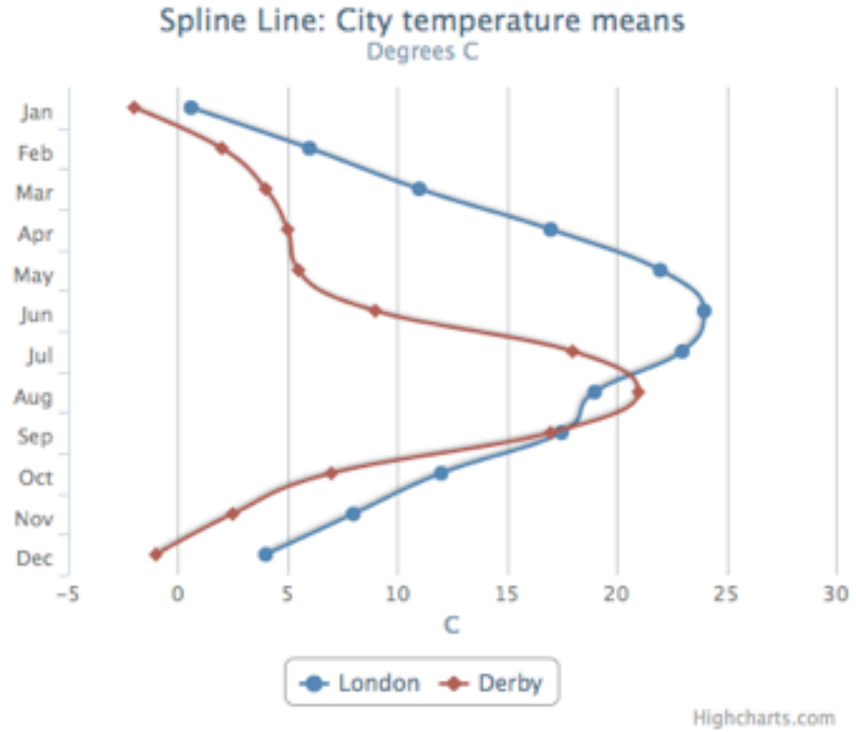
Code required for search page

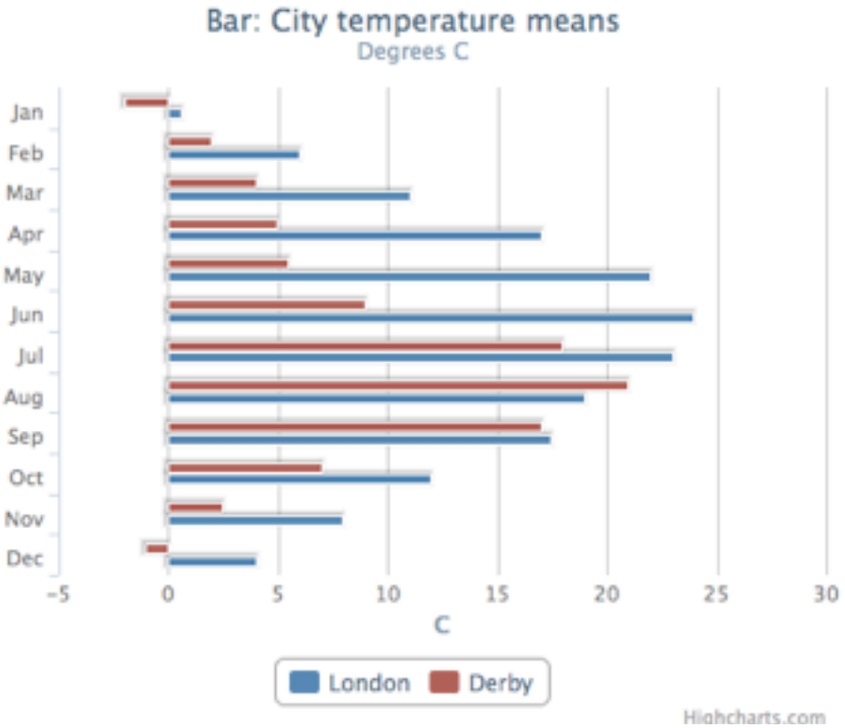
```
1  $(document).ready(function() {
2    var db = new mldb();
3
4    var wgt = new com.marklogic.widgets.searchpage("search-page");
5    wgt.results.addProcessor("animals", function(result) {...
11   wgt.results.addProcessor("movies", function(result) {...
20   wgt.results.addProcessor("svg", function(result) {...
42
43   var ob = new com.marklogic.widgets.options();
44   ob.defaultCollation("http://marklogic.com/collation/en")
45     .collectionConstraint() // default constraint name of 'collection'
46     .rangeConstraint("animal",["item-order"]) // constraint name defaults to that of the range element name
47     .rangeConstraint("family",["item-frequency"]) // constraint name defaults to that of the range element name
48     .rangeConstraint("actor",["item-frequency"],"http://marklogic.com/collation/")
49     .rangeConstraint("year",["item-order"],"http://marklogic.com/collation/")
50     .rangeConstraint("city",["item-order"],"http://marklogic.com/collation/")
51     .rangeConstraint("month",["item-order"],"http://marklogic.com/collation/");
52   var options = ob.toJson();
53
54   wgt.setOptions("mldbtest-page-search-options",options);
55   wgt.execute();
56
57 });
```

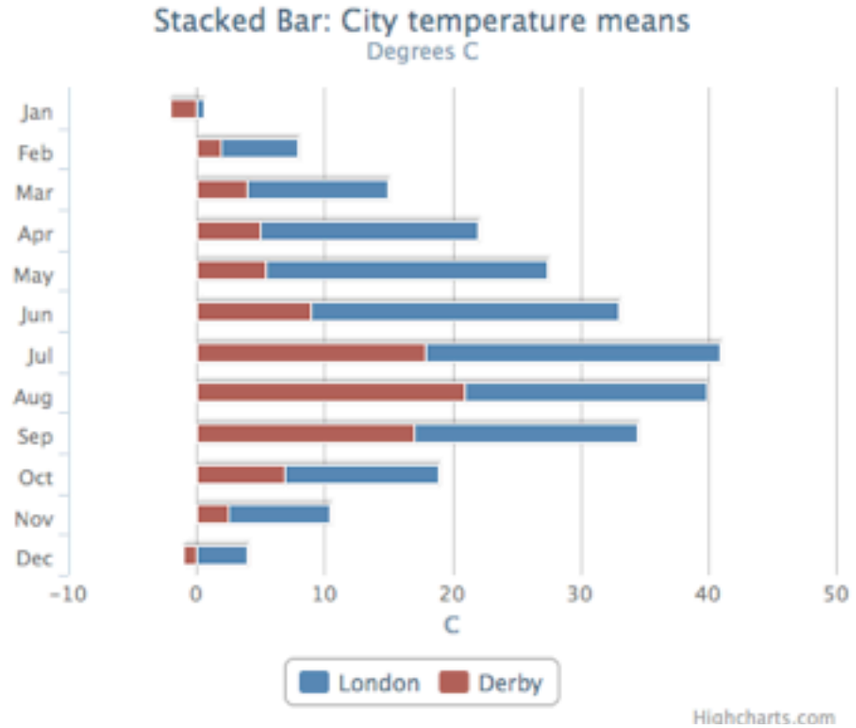

MarkLogic

HighCharts linked to MLJS



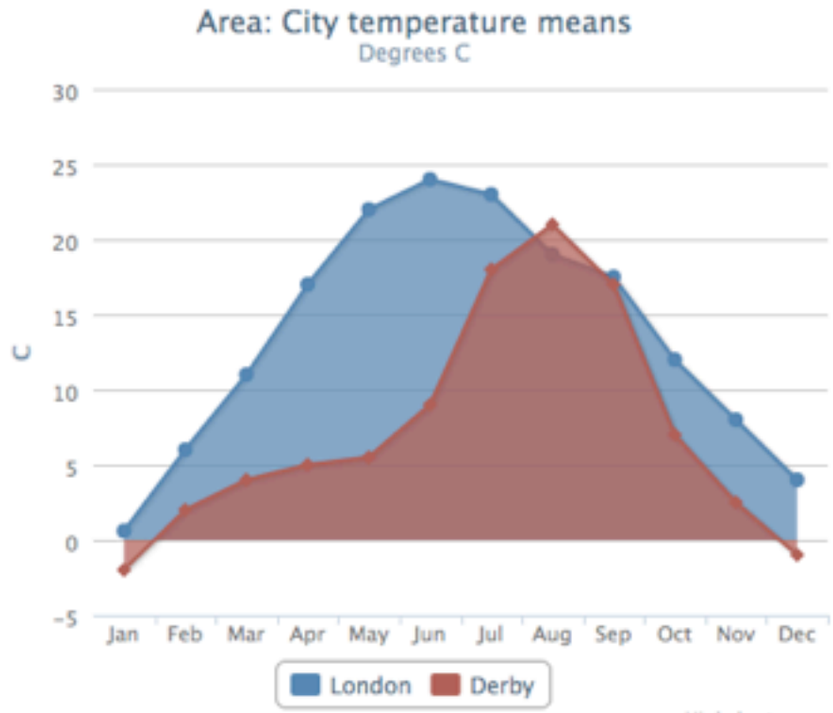


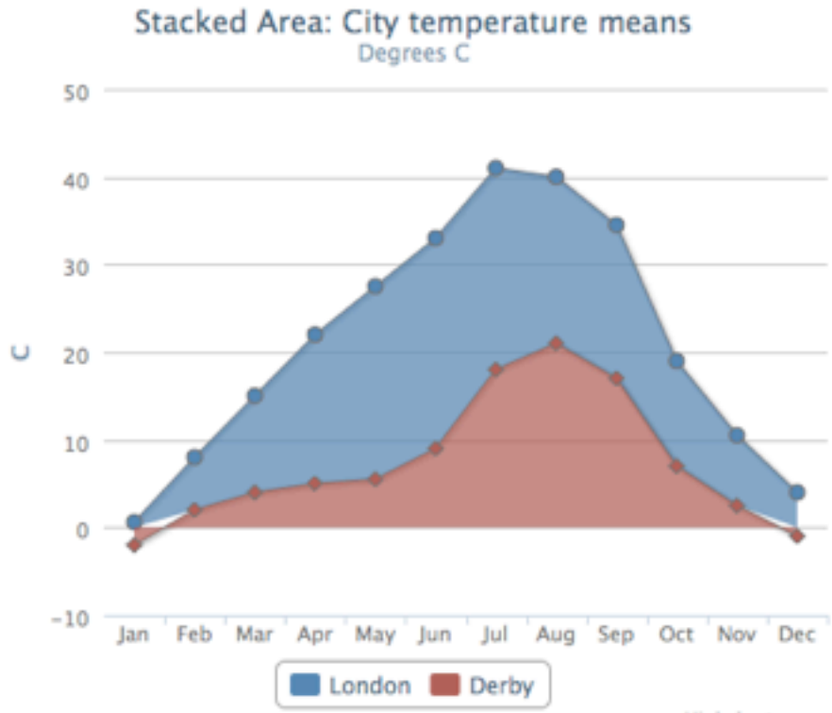


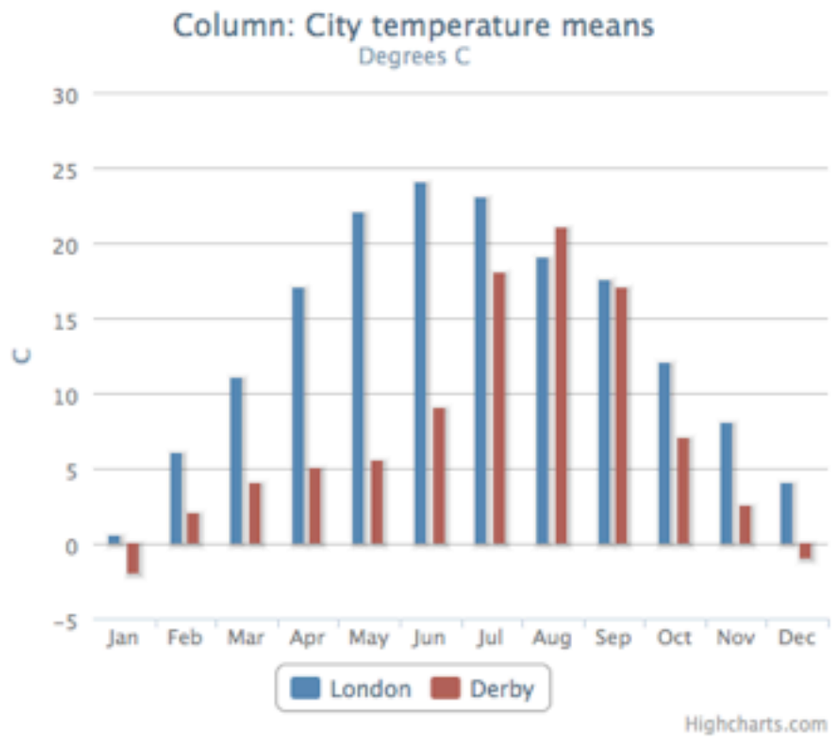


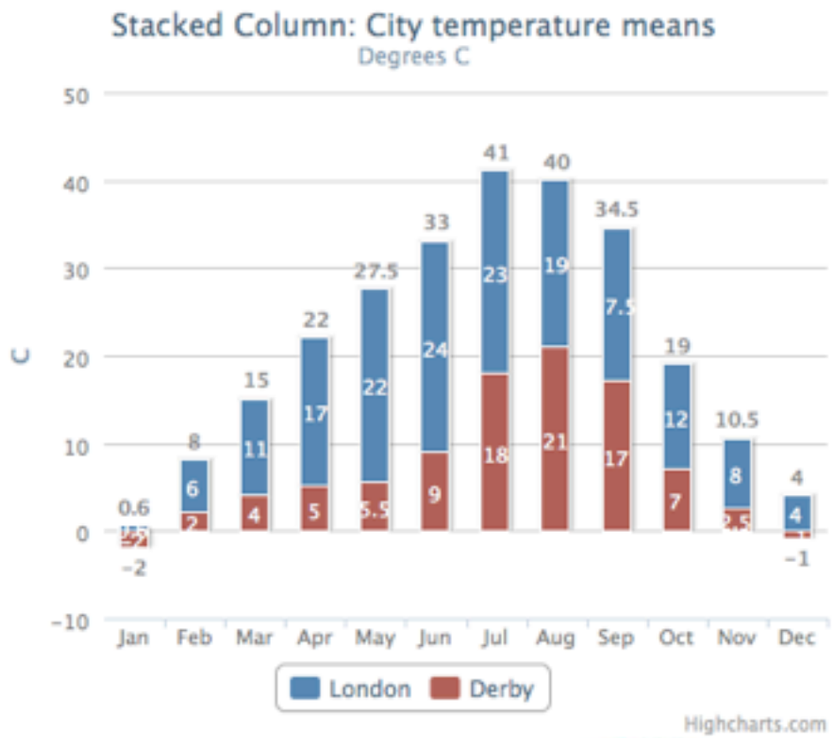
MarkLogic

HighCharts linked to MLJS









MarkLogic

Code required for HighCharts

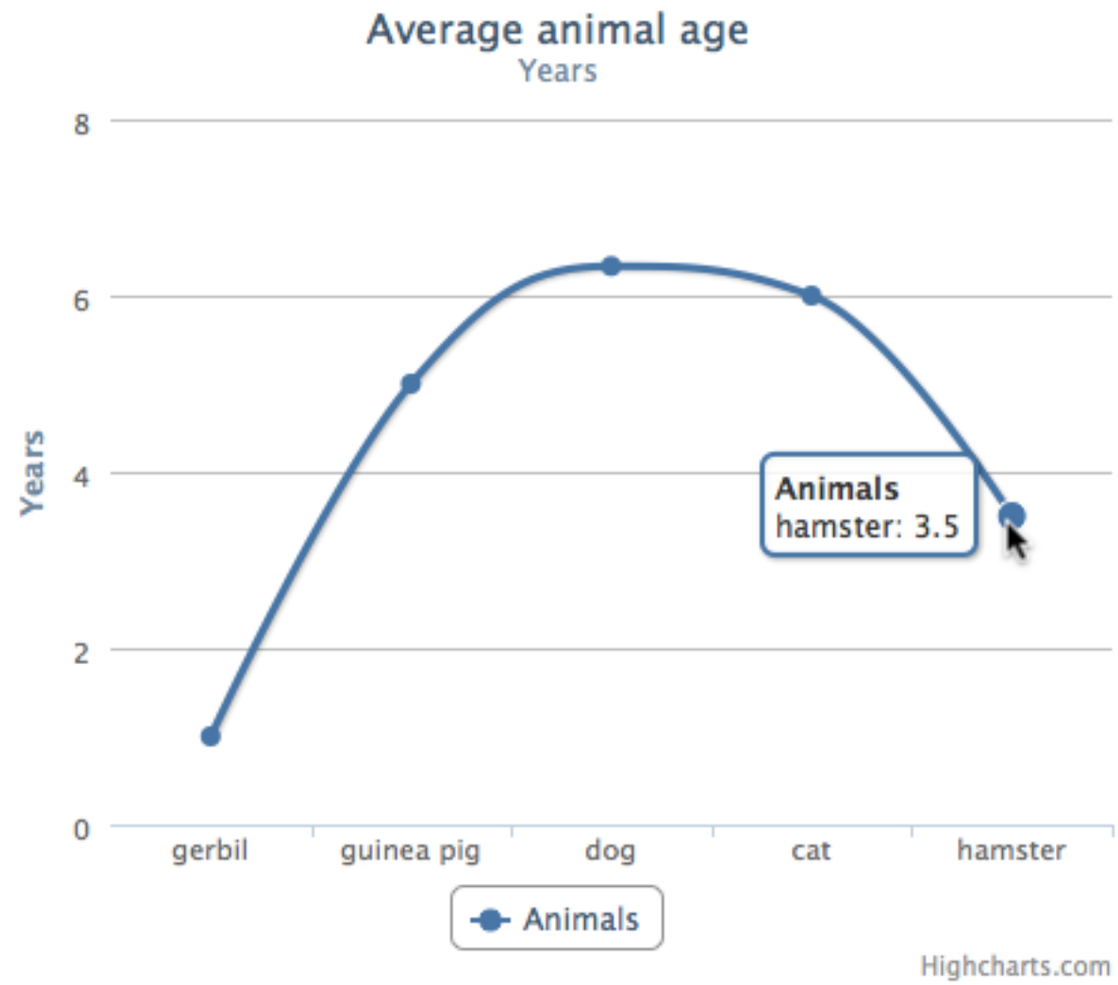
```
$(document).ready(function() {
  var db = new mldb();

  var optionsName = "page-charts-tempchart";
  var ob = new com.marklogic.widgets.options();
  ob.pageLength(100);
  var options = ob.toJson();

  var tempchart = new com.marklogic.widgets.highcharts("tempchart");
  tempchart.setSeriesSources("city","month","reading.temp");
  tempchart.options.title.text = "Line: City temperature means";
  tempchart.options.subtitle.text = "Degrees C";
  tempchart.options.yAxis.title.text = "C";

  // other chart definitions here

  var qb = new com.marklogic.widgets.query();
  qb.query(qb.collection("temperatures"));
  var query = qb.toJson();

  db.saveSearchOptions(optionsName,options,function(result) {
    db.structuredSearch(query,optionsName,function(result) {
      tempchart.updateResults(result.doc);
      // other chart updates here
    });
  });

});
```

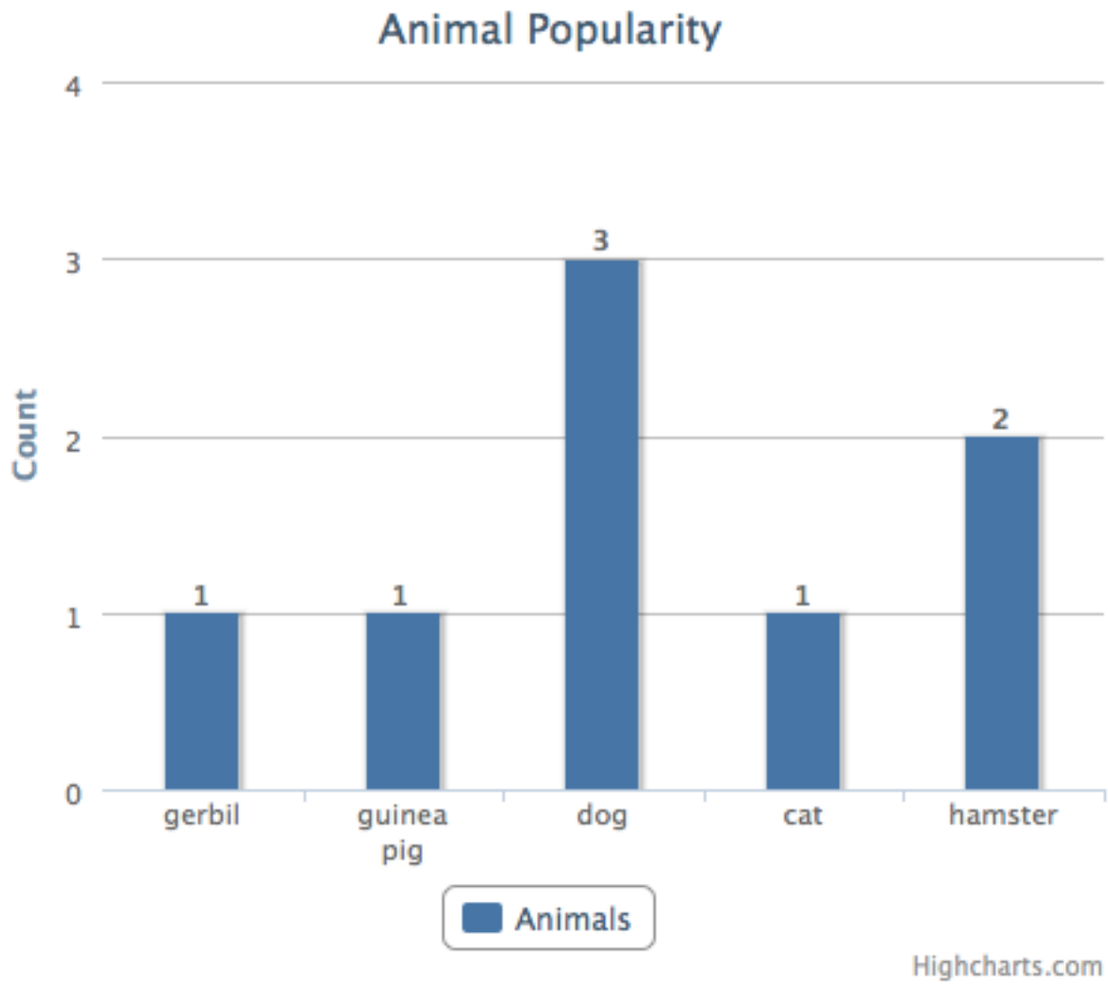
NB This demo shows values coming from the document. MLJS High Charts widget also supports facet values too.

MarkLogic

Mixing search and HighCharts

Search: family:"pet" Search





My Application Help | Contact Us | Terms of Use

MarkLogic

```
$(document).ready(function() {
  var db = new mldb();
  db.logger.setLogLevel("debug");

  var optionsName = "page-charts-search";

  var tempspline = new com.marklogic.widgets.highcharts("splineline");
  tempspline.setSeriesSources("#Animals","animal","age");
  tempspline.setAggregateFunction("mean");
  tempspline.setAutoCategories(true);
  tempspline.options.title.text = "Average animal age";
  tempspline.options.subtitle.text = "Years";
  tempspline.options.yAxis.title.text = "Years";
  tempspline.options.chart.type = 'spline';
  tempspline.options.plotOptions = { spline: { dataLabels: { enabled: false }},tooltip:
    {formatter: function() { return Highcharts.numberFormat(this.y,1); } } };

  var tempcolumn = new com.marklogic.widgets.highcharts("column");
  tempcolumn.setSeriesSources("#Animals","animal","animal");
  tempcolumn.setAggregateFunction("count");
  tempcolumn.setAutoCategories(true);
  tempcolumn.options.title.text = "Animal Popularity";
  tempcolumn.options.subtitle.text = "";
  tempcolumn.options.yAxis.title.text = "Count";
  tempcolumn.options.chart.type = "column";
  tempcolumn.options.plotOptions = {column: {pointPadding: 0.2,borderWidth: 0,
    dataLabels: { enabled: true, style: { fontWeight: 'bold' } } } };

  var options = { options: { ... } };

  var bar = new com.marklogic.widgets.searchbar("cs-bar");
  bar.setOptionsName(optionsName);
  bar.setCollection("animals"); // restrict all search results

  bar.addResultsListener(function(results) {
    tempspline.updateResults(results);
    tempcolumn.updateResults(results);
  });

  db.saveSearchOptions(optionsName,options,function(result) {
    console.log("search options saved");
    bar.execute(); // show some search results by default
  });
});
```

MarkLogic

Demo time

- MLJS Web Test (5 minutes)
  - Standard Roxy based test app that comes with MLJS
  - Used for testing all widgets in MLJS in real life scenarios
  - Includes test data set up on first page (just bootstrap, deploy, view)
- Secure Document Management (5 minutes)
  - Document upload, conversion to XHTML, entity extraction.
  - UI Shows each paragraph allowing setting of security at paragraph level
  - Suggests triples based on entity proximity and likely relationship
  - Adds triples to triple store, linked back to original document
  - Perhaps a search across triples and web of facts to find suspects
- Team KoJAk's SYTYCD entry (7 minutes)
  - Migrate to disparate customer relational databases to the triple store using the RDB2RDF wizard widget
  - Search over this, visualising result in table and graph explorer, alongside facet information from related documents

MarkLogic

What the imminent future holds

Upcoming releases / dates

- End Oct 13 - MLJS 1.0 - First 'Production' release
  - Fully tested and documented MarkLogic V7 support. All today's features.
  - Aims to provide everything AppBuilder provides, including Mapping
  - Video and text how to guides for common use cases
- End Nov 13 - MLJS 1.1
  - Interim release for V7 General Availability for any compatibility issues
  - PROV-O Ontology support, windows support for tools (bash scripts)
- End Dec 13 - MLJS 1.2
  - Workplace Widget - no coding, just click configure on a page to choose layout, widget configuration, event actions
  - Facet buckets, sliders, hierarchies. Multi axes charts. Database Browser.
  - CORS support to power embedding MLJS workplaces in other apps, and WebSockets alerting via Node.js with simultaneous connection directly to MarkLogic for queries

MarkLogic

Themes for 2014 releases

- End Feb 14 - MLJS 1.4 - Visualisations
  - More chart types from HighCharts
  - Turn Situational Awareness demo in to a set of widgets
  - File upload (including multiple files)
  - HTML content editing
  - Form document creation (finish off ‘create’ widget)
- End Apr 2014 (MLW 14) - MLJS 1.6 - Management and Ingest
  - Widgets for server administration as required (E.g. Index creation, REST instance creation)
  - TSV and CSV ingest helpers (InfoStudio? REST extension?)
  - Convert AppBuilder app to MLJS app helper tool

MarkLogic

Any Questions?

MarkLogic