oVirt JavaScript SDK

Design Proposal

Vojtech Szöcs

Software Engineer at Red Hat

April 7, 2014

Topics covered

- Java SDK overview
- JavaScript SDK design overview
- JavaScript Binding API proposal
- Future plans

REST API – example

- **GET** `/api/datacenters` ... list all DCs
- **POST** `/api/datacenters` ... add new DC
- **GET** `/api/datacenters/{id}` ... get DC
- **PUT** `/api/datacenters/{id}` ... update DC
- **DELETE** `/api/datacenters/{id}` ... remove DC

Java SDK

http://www.ovirt.org/Java-sdk

- Create API entry point

```
Api api = new Api(
   "http://127.0.0.1:8080/ovirt-engine/api",
   "admin@internal", "secret");
```

- Extra API options
  - `noHostVerification` ... bypass Engine certificate (ca.crt) verification
  - `filter` ... results are filtered based on user's permissions
  - `persistentAuth` ... activate cookie-based persistent authentication

Java SDK

http://www.ovirt.org/Java-sdk

- Resource collection decorator

```
DataCenters dcCol = api.getDataCenters();
```

- Operations on resource collection

```
// GET /api/datacenters
List<DataCenter> dcList = dcCol.list();

// POST /api/datacenters
DataCenter newDc = dcCol.add(dcPojo);

// GET /api/datacenters/{id}
DataCenter dc = dcCol.get(id);
```

Java SDK

http://www.ovirt.org/Java-sdk

- Operations on resource

```
DataCenter dc = dcCol.get(id);

// Update resource state
dc.setName("lala");

// PUT /api/datacenters/{id}
dc = dc.update();

// DELETE /api/datacenters/{id}
dc.delete();
```

Java SDK

http://www.ovirt.org/Java-sdk

- Entity vs. Resource

```
// Entity & Entity collection = POJO (just data, no logic)
// Generated from REST API entity XML schema – api.xsd

@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "DataCenter", propOrder = {
    "storageType", ...
})
public class DataCenter extends BaseResource {

    @XmlElement(name = "storage_type")
    protected String storageType;

    // Rest of DC-specific attributes

    // JavaBeans-style getters and setters

}
```

Java SDK

http://www.ovirt.org/Java-sdk

- Entity vs. Resource

```
// Resource & Resource collection = entity decorated with logic
// Generated from REST API RESTful description – api.rsdl

public class DataCenter extends
        org.ovirt.engine.sdk.entities.DataCenter {

    // XML marshalling and HTTP communication
    private HttpProxyBroker proxy;

    // Resource collection → getClusters
    private volatile DataCenterClusters dataCenterClusters;

    // RESTful methods like update, delete

}
```

Java SDK

http://www.ovirt.org/Java-sdk

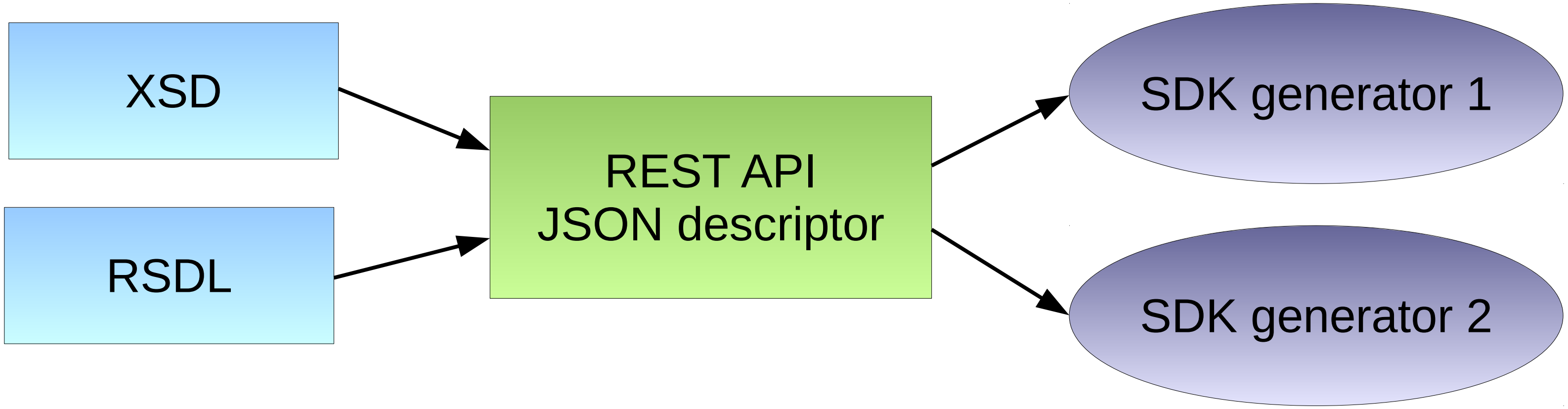
- API is blocking (no callbacks)

```
// GET /api/datacenters
List<DataCenter> dcList = dcCol.list();
```

- Method-level error handling (exceptions)
- Using XML data representation

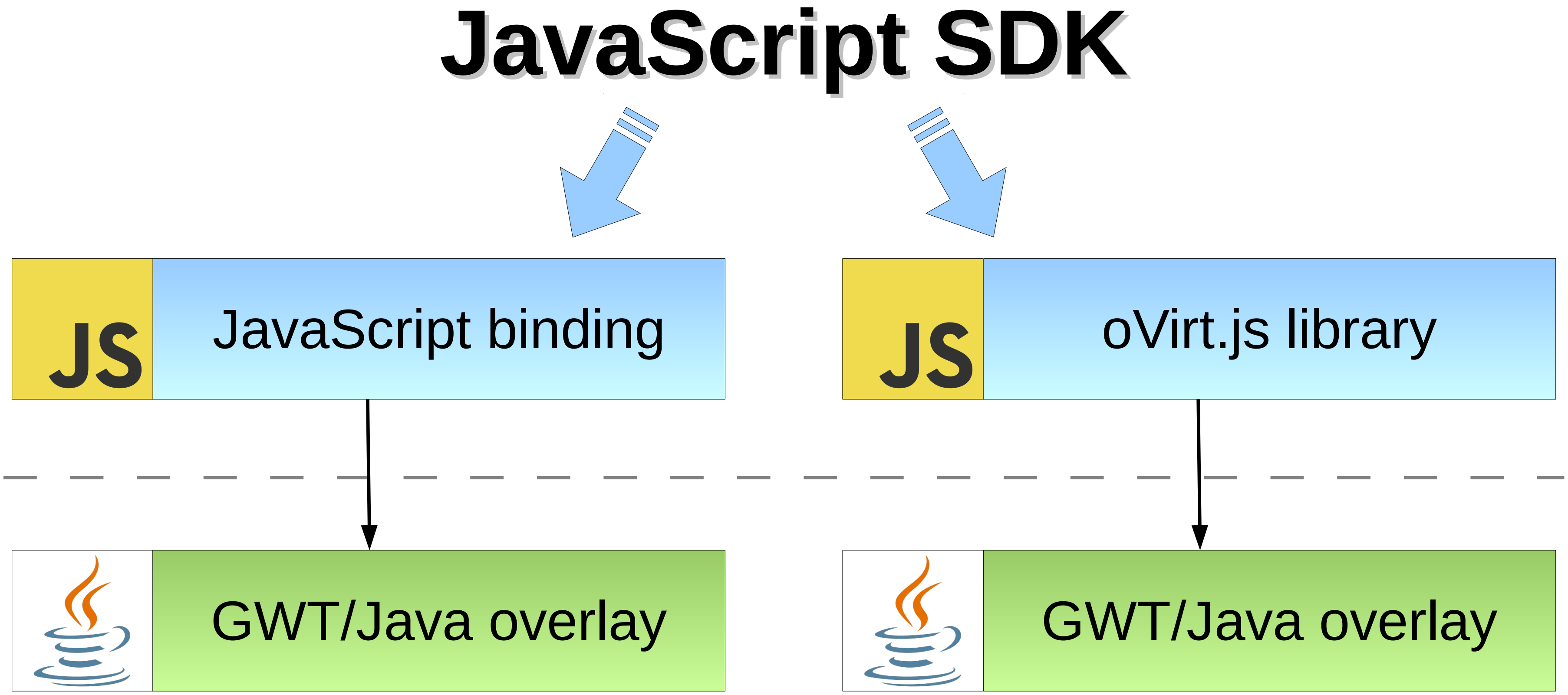
Improving code generator input

- XSD and RSDL generated during Engine build
- Produce JSON describing all resources and operations



JavaScript SDK

http://www.ovirt.org/Features/Design/Using_REST_API_In_Web_UI



JavaScript SDK

JS JavaScript binding

JS oVirt.js library

GWT/Java overlay

GWT/Java overlay

JavaScript SDK

http://www.ovirt.org/Features/Design/Using_REST_API_In_Web_UI

- **JavaScript binding**
  - focus on resources and operations
  - one binding per Engine version / REST API version
- **oVirt.js**
  - binding-agnostic, loads appropriate binding
  - common / useful functionality on top of binding

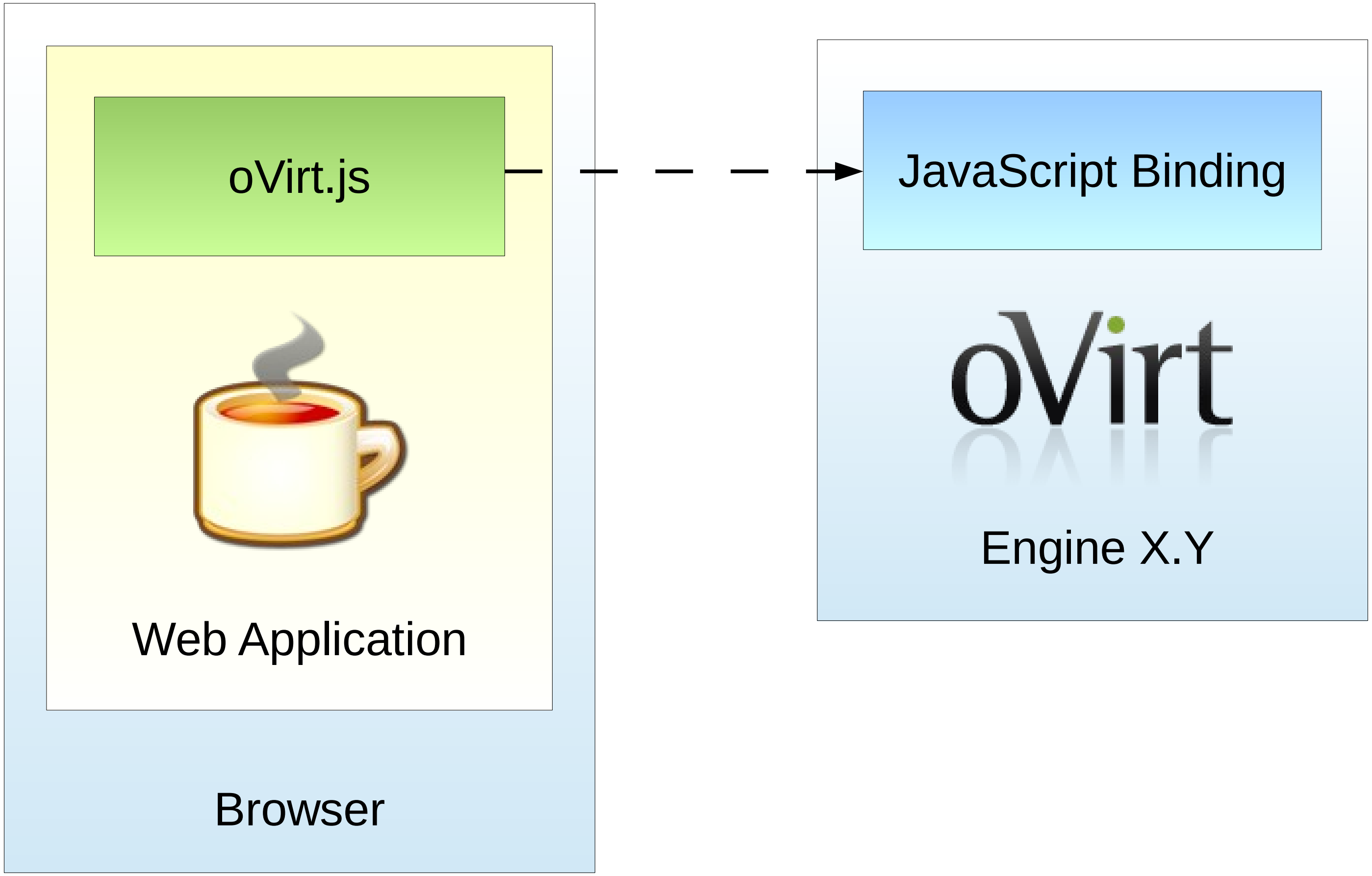
JavaScript SDK

http://www.ovirt.org/Features/Design/Using_REST_API_In_Web_UI

- WebAdmin / UserPortal will use oVirt.js
- Freedom to use oVirt.js or specific binding directly
- oVirt.js allows to evolve SDK beyond generated code

JavaScript SDK

http://www.ovirt.org/Features/Design/Using_REST_API_In_Web_UI



JavaScript Binding API

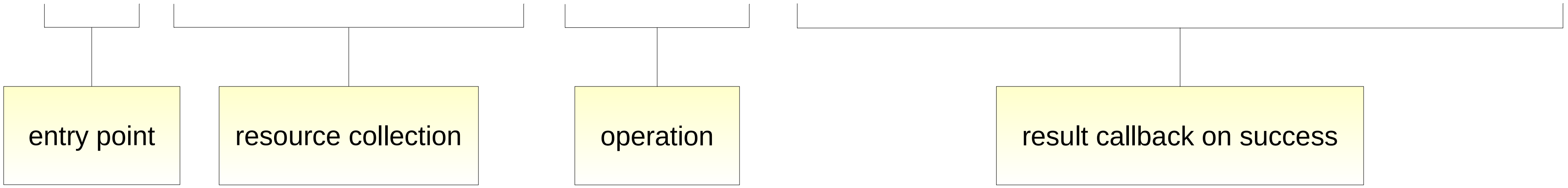
- **Design goals**
  - API usability and readability
  - avoid “frankencode”
  - embrace async nature through non-blocking API

JavaScript Binding API

```
// Use single global variable to contain entire SDK

// Utilize fluent interface:
//     http://martinfowler.com/bliki/FluentInterface.html

// Basic operations mapped to HTTP methods:
//     list,get    ... GET
//     add         ... POST
//     update      ... PUT
//     remove      ... DELETE

sdk.DataCenters.list().success(callback_dcList);

sdk.DataCenters.get(id).success(callback_dc);

sdk.DataCenters.add(dcObj).success(callback_dc);

dc.update().success(callback_dc); // success callback optional

dc.remove().success(callback); // success callback optional
```

JavaScript Binding API

```
// API call break-down
sdk.DataCenters.list().success(callback_dcList);
```



entry point

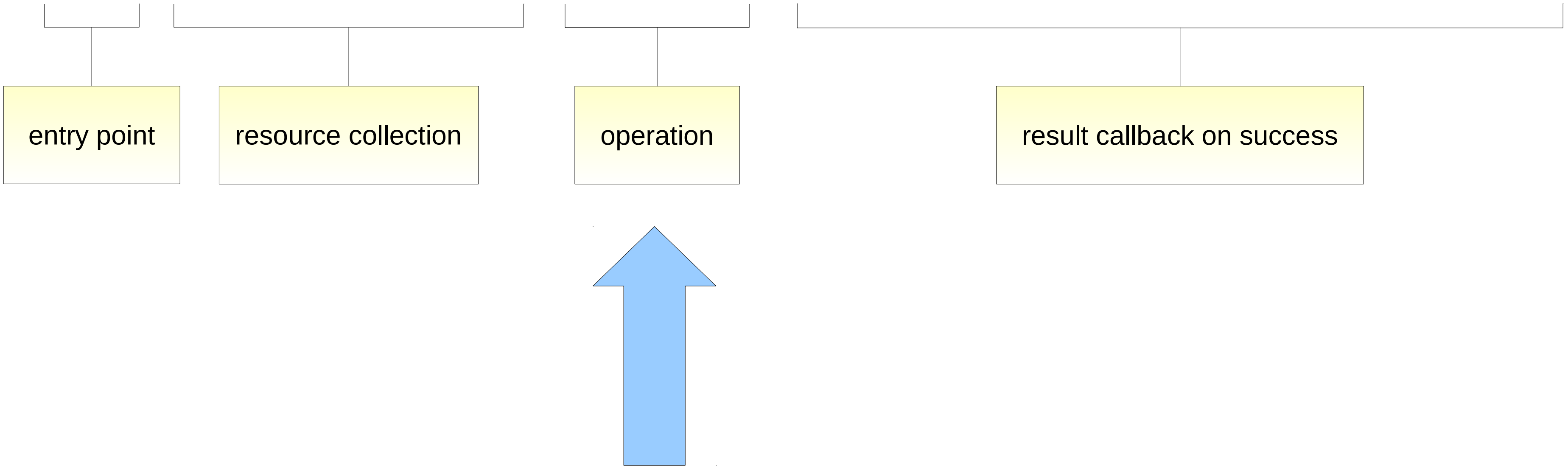
resource collection

operation

result callback on success

JavaScript Binding API

```
// API call break-down
sdk.DataCenters.list().success(callback_dcList);
```



entry point

resource collection

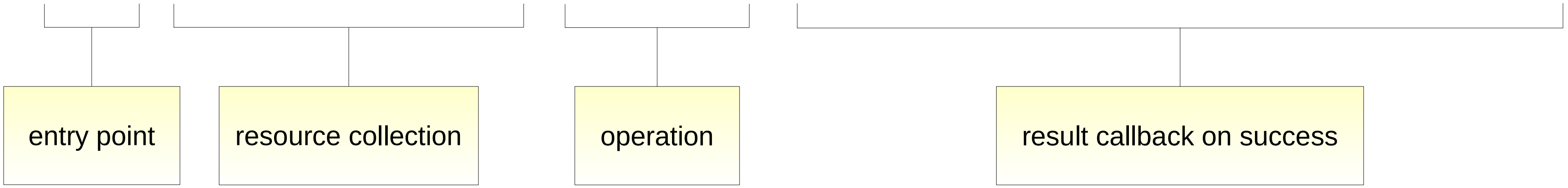
operation

result callback on success

Schedule XHR
by default

JavaScript Binding API

```
// API call break-down

sdk.DataCenters.list().success(callback_dcList);
```



entry point

resource collection

operation

result callback on success

- entry point / resource → resource collections
- resource / resource collection → operations
- operation = reusable RESTful method abstraction

JavaScript Binding API

✔ XHR control
and scheduling

```
// Execute XHR on-demand

sdk.DataCenters.list({runNow:false})
   .success(callback_dcList)
   .run();
```

```
// Control XHR

var op = sdk.DataCenters.list().success(callback_dcList);

var inProgress = op.pending(); // true if awaiting response

op.kill(); // if pending, kill current XHR

op.retry(); // kill() and run()
```

JavaScript Binding API

✔ XHR control
and scheduling

```
// Schedule XHR execution

var op = sdk.DataCenters.list()
            .success(callback_dcList);

var cmd;

cmd = sdk.Scheduler.run(op).each(5000); // repeating command

cmd = sdk.Scheduler.run(op).once(1000); // future command

cmd = sdk.Scheduler.run(op).once(); // deferred command → once(1)

cmd.cancel();
```

JavaScript Binding API

✔ Multi-level options

```
// Options on operation level

var op = sdk.DataCenters.list({runNow:false})
            .success(callback_dcList);

op.options({search:'name=lala'}); // for next XHR execution


// Options on resource collection level

sdk.DataCenters.options = {maxResults:10};


// Options on global API level

sdk.options = {filterResults:true};
```

```
obj.options = {a:b};
obj.options['a'] = b;
obj.options.a = b;
```

JavaScript Binding API

✔ Multi-level error handling

```
// Error handling on operation level

sdk.DataCenters.list()
   .success(callback_dcList)
   .error(callback_error);


// Error handling on resource collection level

sdk.DataCenters.errorHandler = dc_error_handler;


// Error handling on global API level

sdk.errorHandler = global_error_handler;
```

JavaScript Binding API

- ✔ Multi-level result mapping

```
// Result mapping on operation level

sdk.DataCenters.list()
   .mapper(dc_mapper)
   .success(callback_dcList_mapped);


// Result mapping on resource collection level

sdk.DataCenters.resultMapper = dc_result_mapper;


// Result mapping on global API level

sdk.resultMapper = global_result_mapper;
```

JavaScript Binding API

- ✔ Resource manipulation

```
// Query and update resource
var dcName = dc.name();
dc.name('lala');

// Bulk update
dc.set({name:'lala'});

// Synchronize changes
dc.update();
```

JavaScript Binding API

- ✔ Dirty resource tracking

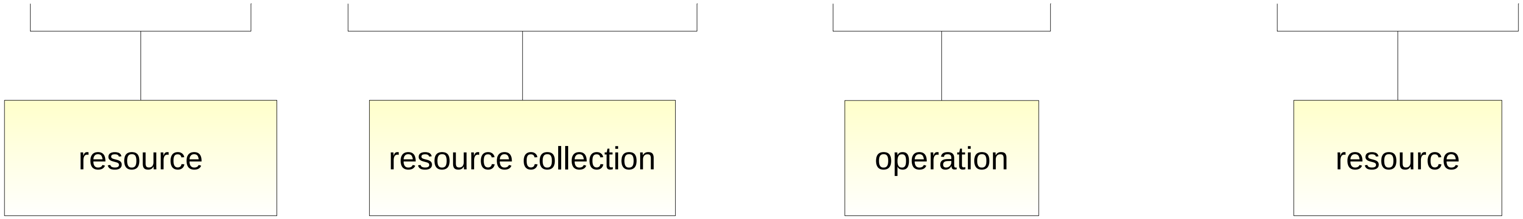
```
// Enable dirty tracking
dc.trackDirty();

// Update resource
dc.name('lala');
```

Schedule update

JavaScript Binding API

✔ Resource sub-collections

```
// Resource collection access
sdk       .DataCenters    .get(id)  ...  dc
dc        .Clusters       .get(id)  ...  cluster
cluster   .Networks       .get(id)  ...  network
```



JavaScript Binding API

- ✔ Asynchronous mappers

```
// Control success callback invocation

sdk.DataCenters.get(id)
   .mapper(dc_async_mapper)
   .success(callback_dc_mapped);
```

```
var dc_async_mapper = function (dc, control) {
   dc.Clusters.list().success(function (dcClusters) {
      var extendedDc = sdk.extend(dc, {
         clusters: dcClusters;
      });
      control.success(extendedDc);
   });
   return false; // Don't invoke success callback now
};
```

JavaScript Binding API

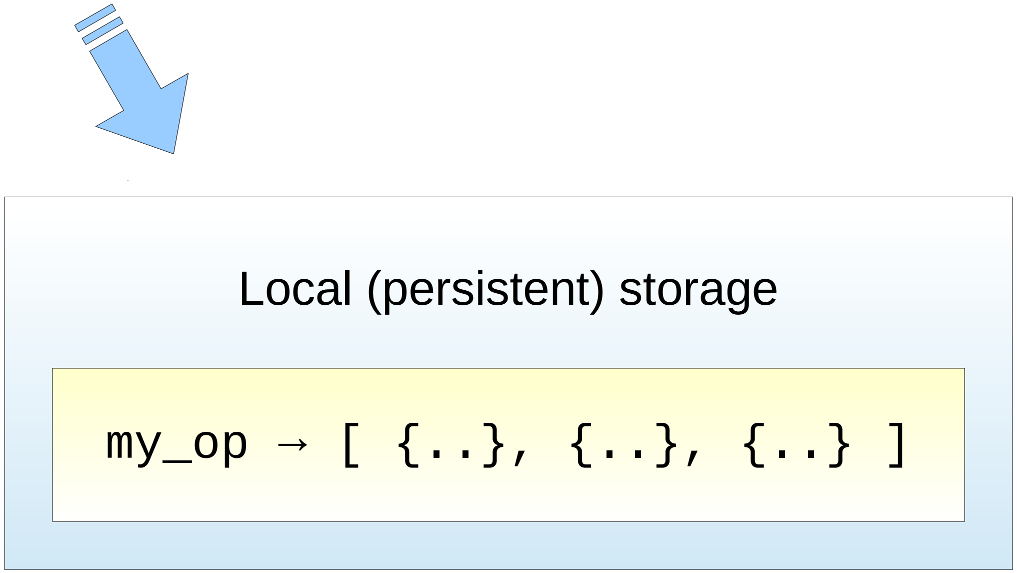
✔ Operation logging

```
// Log operation / XHR execution

var op = sdk.DataCenters.list()
            .success(callback_dcList);

op.options({log:true}); // use default log handler → console.log()

op.options({log:dcList_log_handler}); // use custom log handler


// Configure logging on resource collection and global API level

sdk.DataCenters.options.log = dc_log_handler;

sdk.options.log = global_log_handler;
```

JavaScript Binding API

✔ Resource cache

```
// Cache operation results (resources) on client

var op = sdk.DataCenters.list()
                .success(callback_dcList);

op.cache('my_op'); // define cache key
```



Local (persistent) storage

```
my_op → [ {..}, {..}, {..} ]
```

- update cache on success
- **online mode**
  access cache on error
- **offline mode**
  access cache only (no XHR)

JavaScript Binding API

- ✔ Multiple callbacks

```
// Add multiple callbacks (success,error,mapper)

var op = sdk.DataCenters.list();

op.success(f1)
  .success(f2,f3)
  .success(f_array);


// Control callback execution chain

var f1 = function (dcs, control) {
   control.preventSuccess(); // don't execute f2,f3,f_array
};
```

JavaScript Binding API – example

```
var dc_mapper = function (dc) {
   return sdk.extend(dc, {
      isCool: function () {
         return this.comment() === 'cool';
      }
   });
};

// Update all DCs based on their comment attribute

sdk.DataCenters.list().mapper(dc_mapper).success(function (dcs) {
   var i, dc;
   for (i = 0; i < dcs.length; i += 1) {
      dc = dcs[i];
      if (!dc.isCool()) {
         dc.comment('hot');
         dc.update();
      }
   }
});
```

Things to consider

- Cookie-based persistent authentication http://www.ovirt.org/Features/RESTSessionManagement
  - one cookie (JSESSIONID) for REST API endpoint
  - multiple web applications / UI plugins

Future plans

- JavaScript binding prototype for few select entities
- oVirt.js prototype (thin) to work with JavaScript binding
- Generate JavaScript binding during Engine build
- Expose JavaScript binding files from Engine server
- Generate GWT/Java overlay for binding & oVirt.js
- Migrate WebAdmin & UserPortal to use oVirt.js

Thank you!

vszocs@redhat.com

vszocs at #ovirt (irc.oftc.net)