Clients - JavaScript

Downloading the client

Please see https://github.com/smart-on-fhir/client-js (https://github.com/smart-on-fhir/client-js)

Initializing the client

Before you are able to run any operations against the FHIR API using the JS client, you will need to initialize it first. The following code snippet illustrates the basic boilerplate code that you can use for this purpose:

```
FHIR.oauth2.ready(function(smart){
    // now do something cool
});
```

Obtaining the context

Once the client is initialized, you can obtain the context in which it was launched (the user who authorized the client and, if applicable, the patient that has been selected) by using the following methods:

- `smart.user.read()`
- `smart.patient.read()`

Both of these return a jQuery Deferred object which you can register a success callback to process the returned FHIR resource.

Embedded FHIR client library

The SMART JS client uses the open-source fhir.js (https://github.com/FHIR/fhir.js) for interfacing with SMART API servers. Upon successfully initializing and negotiating the SMART authorization sequence, the client will expose one or two instances of the `fhir.js` client as applicable via the following handles (assuming that you use the name variable name `smart` for your SMART on FHIR client in scope):

- `smart.api` Non-context aware API for executing operations on all authorized resources
- `smart.patient.api` Context aware API which automatically applies its operations to the patient in context

Supported API operations

All operations available in `fhir.js` are supported. These include:

- `read` Read the current state of a given resource
- `search` Obtain a resource bundle matching specific search criteria
- `fetchAll` Retrieve the complete set of resources matching specific search criteria
- and many others

Please see the fhir.js documentation (https://github.com/FHIR/fhir.js) for the complete list of available operations.

Reading a single resource

Instance-level operations in FHIR work with a single resource instance at a time. Two key operations are `read` and `vread`:

- `smart.api.read({type: resourceType, id: resourceId})` Read the current state of a given resource
- `smart.api.vread({type: resourceType, id: resourceId, versionId: versionId})` Read a specific version of a given resource

Just change `resourceType` to the name of a FHIR resource from this complete list (http://www.hl7.org/implement/standards/fhir/resourcelist.html).

Searching for resources

The most important type-level operation is `search`, which allows you to find resources of a given type, with a set of filters applied.

To search for resources you'll use either:

- `smart.api.search({type: resourceType, query: queryObject})` when you want to search across patients, or
- `smart.patient.api.search({type: resourceType, query: queryObject})` when you only want results for the patient in context.

(Note: in any case your app will only be able to read the data it's authorized to see – so you could be lazy and query for "all" the data knowing that authorization restrictions will limit the data returned. But it's a good practice to write explicit patient-level queries when that's what you have in mind, because it will make your intentions more clear – and make your code more portable.)

FHIR Search Parameters

When you perform a FHIR `search` operation, you'll often want to apply filters to limit the results you get back. For example, you may want to fetch a list of HbA1c lab results when you're not interested in cholesterol readings. You can create an expressive set of filters using FHIR's "search parameters," a set of constraints passed along as URL parameters. The `fhir.js` client used in the SMART on FHIR JS client has built-in support for all search parameters defined in FHIR.

To learn how these work, let's look at an example from the FHIR spec. We'll examine the `Patient` resource. FHIR defines a set of search parameters (http://www.hl7.org/implement/standards/fhir/patient.html#search) for this resource (and all other resources) at the bottom of the resource documentation page. Let's say that we'd like to find all patients that have "John" or "Bob" in their first name and "Smith" in their last name. Here is how we can do this using the SMART on FHIR JS client:

```
smart.api.search({type: "Patient", query: {given: ["John", "Bob"], family: "Smith"})
```

Please see the fhir.js documentation (https://github.com/FHIR/fhir.js) for further examples.

Further Reading

To learn more about the SMART on FHIR JS client, please visit our tutorials page (http://docs.smarthealthit.org/tutorials/)

The SMART on FHIR API is evolving in parallel with the **FHIR ballot releases (http://hl7.org/fhir/timelines.html)**. If you spot problems, please **file an issue (https://github.com/smart-on-fhir/smart-on-fhir.github.io/issues)**. Or better yet, you can **edit this page (https://github.com/smart-on-fhir/smart-on-fhir.github.io/edit/master/clients/javascript/index.md)**.