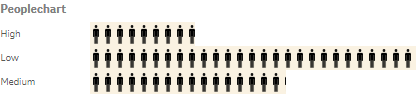
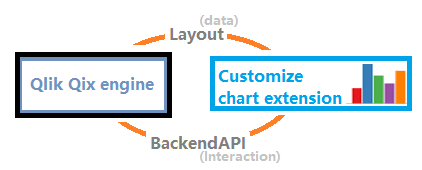
Sample extension (3) People chart- play with data



In this tutorial, you will learn:

1. Why requireJS? Modularize your code
2. What is $element and layout parameter
   1. The “$element” parameter – Jquery wrapper for sense
   2. The “Layout” parameter – data layer in sense
3. In the code- Interacting with Qix engine using capability API

At this stage, you are probably wondering how data and user interaction exchange between frontend and backend. For sense extensions, data is framed as hypercube; user interactions are delivered to server via various APIs. At the end of this tutorial, you should be able to retrieve data from server and using backend API to perform data selection. If you intend to build your own visualization with D3, chart.js or any other visualization library, take 30 minutes to study the code behind people chart, there is everything you need.



What is in the folder

|  |  |
| --- | --- |
|  | Main js file |
|  | Property definition js file |
|  | Css file |
|  | Jason metadata file |
|  | Workbench load file |

In people chart folder(C:\Users\[username]\Documents\Qlik\Sense\Extensions\PeopleChart), there are five files including two java script files: com-qliktech-peoplechart.js and peoplechart-properties.js.

We have seen most of these files in [hello world tutorial] [angular chart tutorial], but it is probably the first time you see more than one JavaScript file. com-qliktech-peoplechart.js is the core file, peoplecart-properties.js is an external file for property declaration. In this tutorial, you will learn to modularize your code and use requireJS to load external dependency.

1. Why requireJS?

In previous tutorials we briefly explained using “text!” prefix to load css files (as a text object) and assign it to html head. But requireJS are more capable than importing text.



Web programmers are familiar with <script> tags. to load a JavaScript, the following tag must be included in web page.



There are several problems with this approach.

First, all of the files have to be loaded in order. For example, jQuery is usually a dependency of many modules so jQuery must be loaded before these modules. developers can manage this manually if dependency pool is small and static, unfortunately that’s not the case for Qlik sense.

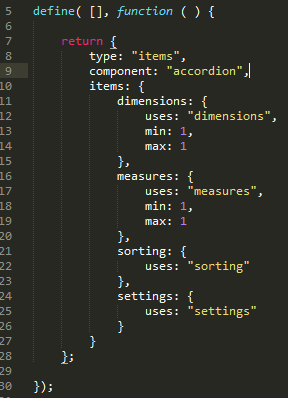
Second, key modules like jQuery and qlik API are referenced by almost every extension, loading same module repeatedly degrade performance significantly.

RequireJS is implemented for dependency management. It is a JavaScript module loader which enables asynchronous loading and internal module referencing, perfect solution to our problem. In this example, module “jquery” is imported as the first dependency. It is a pre-configured, ready to use module. There is another pre-built module for Qlik APIs, referenced as “qlik” which we tried in [the angular chart sample].



http://requirejs.org/docs/whyamd.html

Look at the code in Peoplechart-properties.js, this module does nothing but return a generic object.



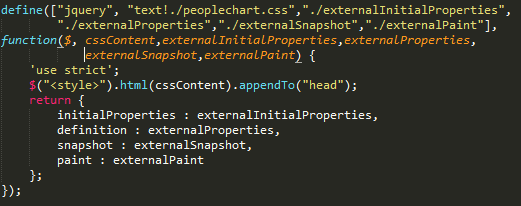
Back to com-qliktech-peoplechart.js, Peoplechart-properties is loaded as  parameter at line 1, then append to  at line 16. It is equivalent to declaring the property directly.



\*Why declaring properties externally, seems like we take an extra mile to achieve the same result.

Well it is not obvious in this example because it’s small. Complicated extensions usually depend heavily on third party libraries/frameworks and with hundreds line of code. It is always desired to modularize the code to improve readability and maintainability.

Cool, now we have learnt to use requireJS loading external modules. You can extract the entire extension if you wish. But design carefully. Granularity does not equal to good code. Too much of granularity is no better than one piece of long code.



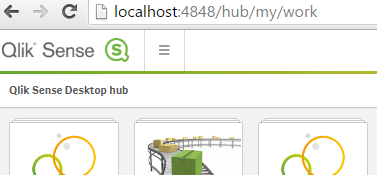
1. The “$element” and “Layout” parameter



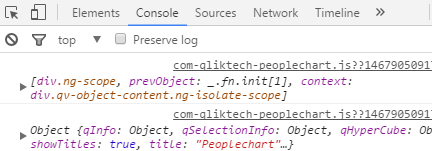
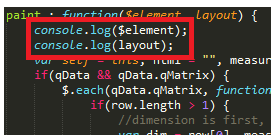
Like any other extension, paint at line 20 is the core method. There are two parameters here: $element and layout. $element is a jQuery wrapper for extension UI. Layout is the data and property layer for backend communication. Now let’s print out the two objects on console to take a closer look.

\*Debugging extension

The process of debugging extension is similar to web development. If you are using Qlik sense desktop, open a browser, enter localhost:4848/hub, Qlik sense will launch in your browser instead of desktop client. Make sure sense client is turned on during the process.

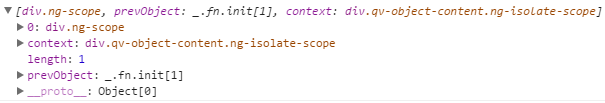


Press 12 to open developer’s tool on your browser, in this tutorial Chrome is used. Two objects are printed on console. The first is $element object, second is layout object.



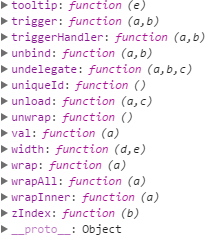
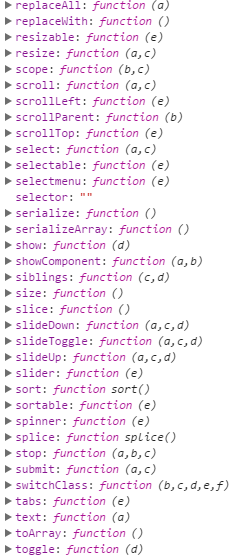
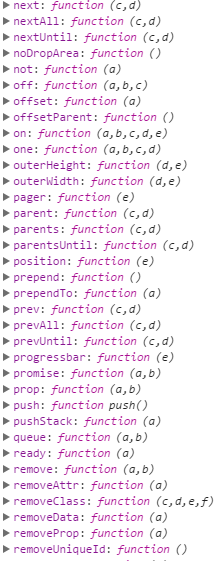
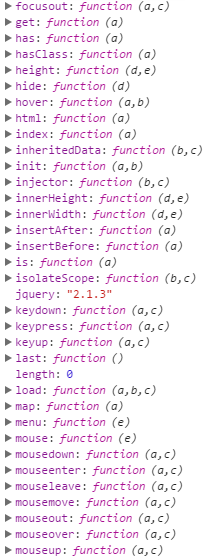
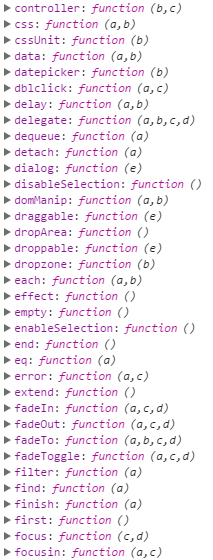
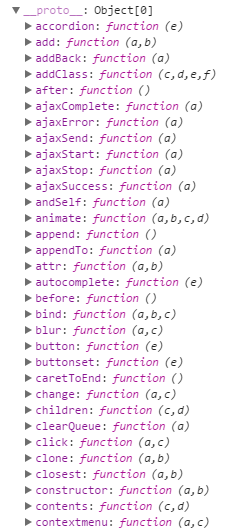
* 1. What is in $element

$element is an ng-scope object.



$element is a Jquery wrapper and Jquery is a JavaScript library supported by almost all major browsers. Jquery makes DOM manipulation much easier.

Expand proto, this is the full list of functions available:



Almost all jquery functions are supported by Qlik. Details of these functions can be found on jquery website, you are safe to use the official API document as reference for $element: <http://api.jquery.com/>.

In this extension, $element is called 3 times.

At line 23, $element.width() is called to retrieve the width of the frame: ;

At line 43, $element.html() is called to append html content to the view:;

At line 44, $element.find() is called to select all ‘.selectable’ elements and setup interacting event on these elements:;

You may also find these functions on jquery official website:

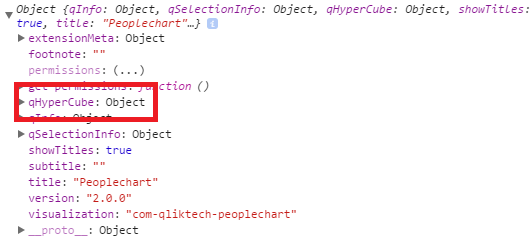
http://api.jquery.com/width/

<http://api.jquery.com/html/>

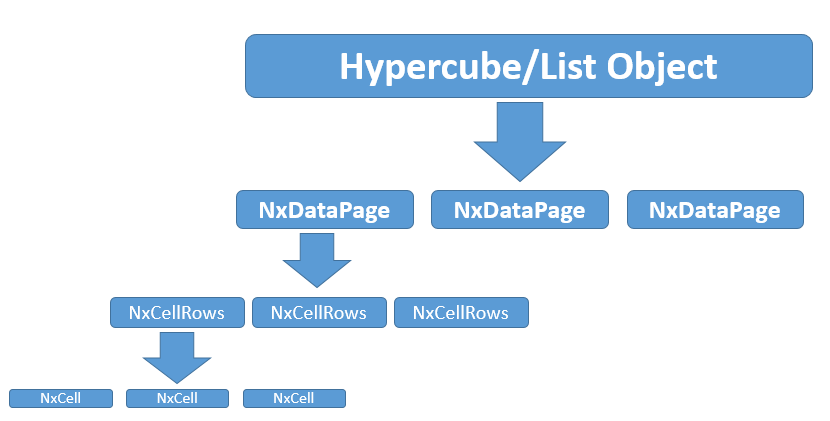
https://api.jquery.com/find/

.

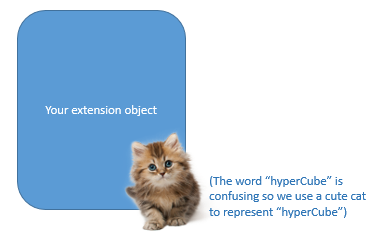
* 1. What is in layout



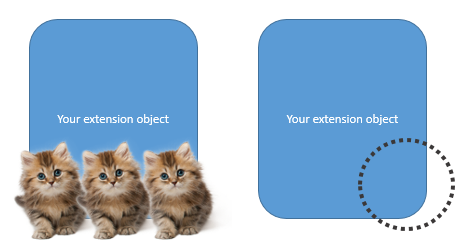
Expand layout object there is a qHyperCube property. For Qliksense, data is transferred and stored in hypercube (n\*dimension+ n\*measure) or listObject (1\*dimension, no measure) [Apendix2 dimension measure lisObj and hyperCube].



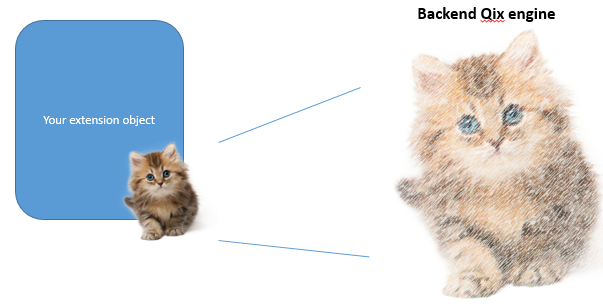
\*\*What is hypercube?



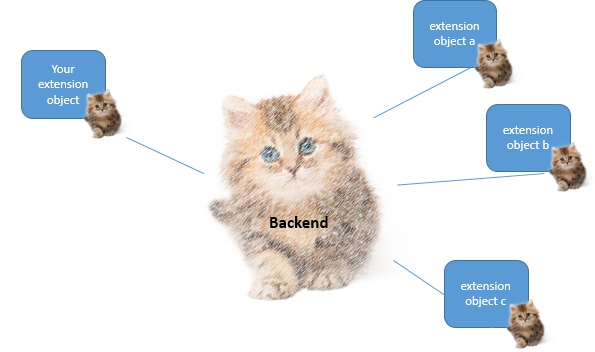
Hypercube is the data framework behind every extension. When an extension is created, we also need to define the hypercube associate with it. In paint function, layout parameter is the access point to this data.



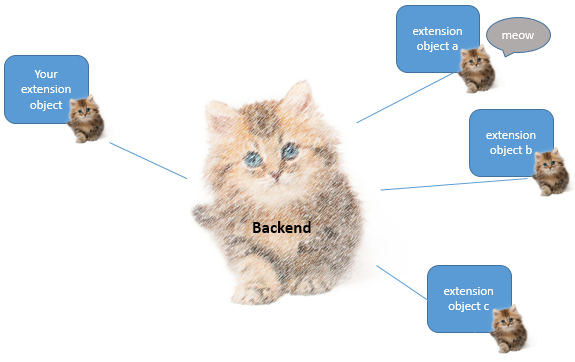
There can be more than one hypercube, or no hypercube.



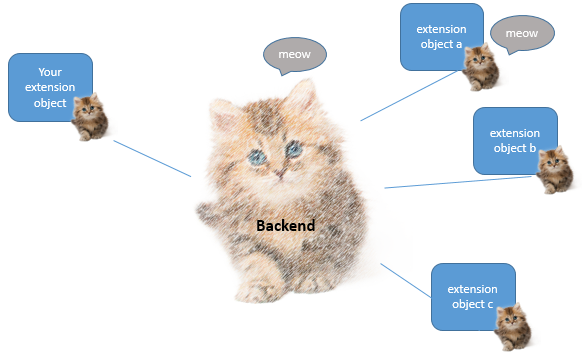
Hypercube is a projection of backend data. The dimensions and measures define the structure of hypercube. Hypercube stays alive at the backend but it is actually one of many ways to represent main data.



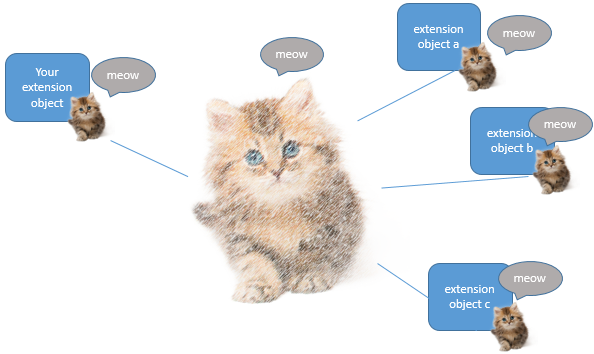
You may have multiple extensions and hypercubes defined for one app. But all of them are linked to the same backend data.



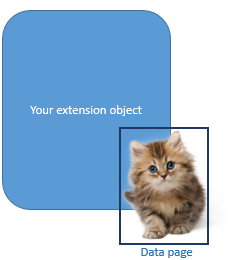
If an action is performed at any extension, for example, a selection on certain dimension.



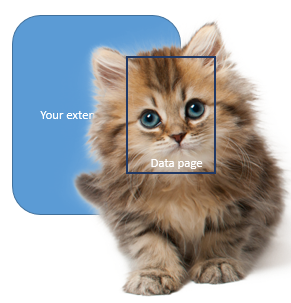
Through capability API, this action is passed to backend. Selection state of main data change.



This action then propagates to other hypercubes. For extensions, paint method is triggered if change of hypercube is detected.

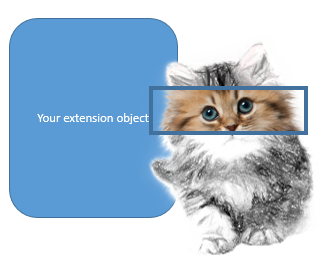
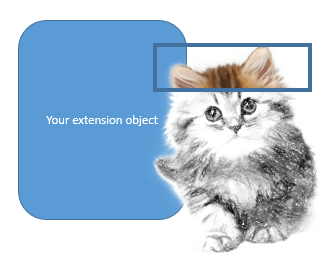


By defining , we define the data to be fetched when extension is created. No problem if data is small.



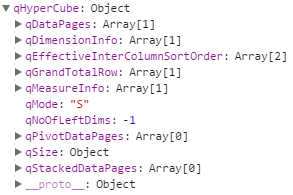
If there is too much data (the limit is 10,000), extension fetches only the first set of data which is defined by the height and weight in ;

https://help.qlik.com/en-US/sense-developer/3.0/Subsystems/APIs/Content/ExtensionAPI/initialproperties-property.htm



Use getData function in backendAPI to retrieve additional data on different data pages: <https://help.qlik.com/en-US/sense-developer/3.1/Subsystems/APIs/Content/BackendAPI/getdata-method.htm>.

Expand qHyperCube, there are 10 properties.

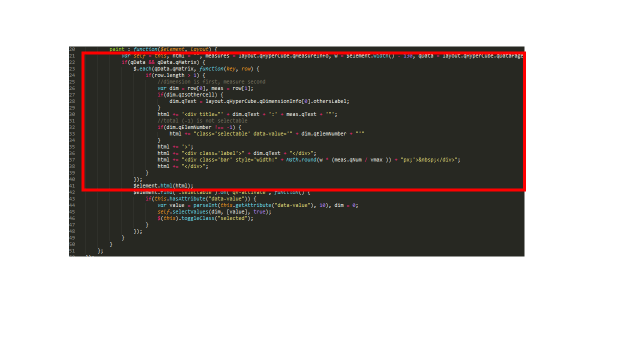


This is probably everything you need for your front end visualization. Take a look at the table below.

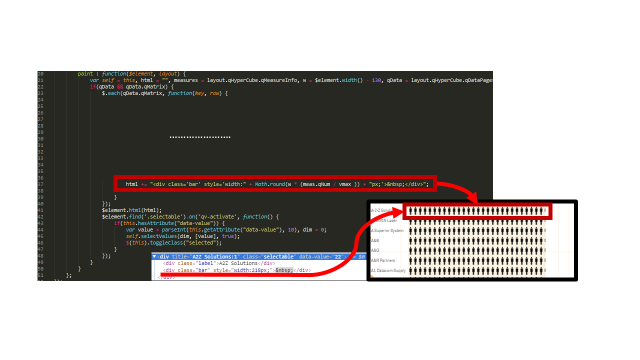
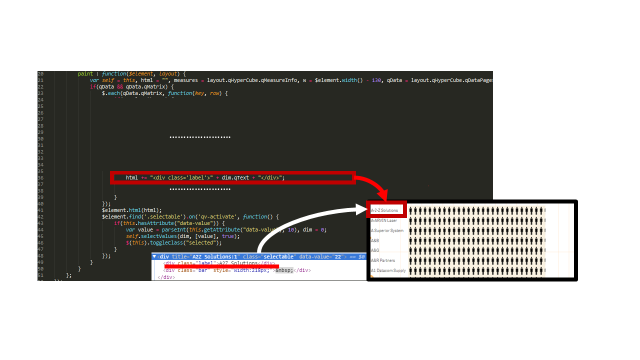
|  |  |
| --- | --- |
| **Name of class** | **Description** |
|  | Set of data.  Is empty if nothing has been defined in qInitialDataFetch in HyperCubeDef. |
|  | Information on the dimension. |
|  | Sort order of the columns in the hypercube.  Column numbers are separated by a comma.  Example: [1,0,2] means that the first column to be sorted was the column 1, followed by the column 0 and the column 2. |
|  | Aggregate for measures of all values in the field.  The result value depends on the qAggrFunc defined in HyperCubeDef. |
|  | Information on the measure. |
|  | S for DATA\_MODE\_STRAIGHT; straight table representation  P for DATA\_MODE\_PIVOT; pivot table representation  K for DATA\_MODE\_STACK; stacked table representatio |
|  | Number of left dimensions.  The index related to each left dimension depends on the position of the pseudo dimension (if any). |
|  | Set of data for pivot tables.  Is empty if nothing has been defined in qInitialDataFetch in HyperCubeDef. |
|  | Defines the size of the hypercube. |
|  | Set of data for stacked tables.  Is empty if nothing has been defined in qInitialDataFetch in HyperCubeDef. |
|  |  |

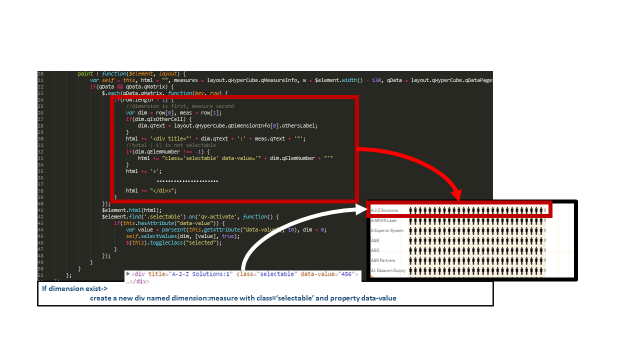
<https://help.qlik.com/en-US/sense-developer/3.0/Subsystems/EngineAPI/Content/GenericObject/LayoutLevel/HyperCube.htm>

4. What is in the code?



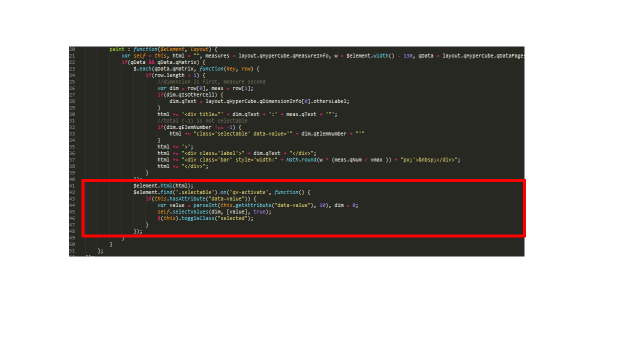
The first part is fairly straight forward. It renders the front end display. Let me explain the code in three pictures without word.





Interacting with Qix engine using capability API

From line 43 to line 50 is data selection part. In the sample code, a function is assigned to all selectable elements in this extension using jQuery. Refer to the third picture in previous section for selectable element.



Self.selectValue is the backendAPI function to perform selection. Refer to [Appendix1 How to use Qlik APIs in extension] for information on accessing various APIs. Backend API is one of the more important APIs. It is accessed using “this” keyword. It provides several wrapper functions to perform selection, retrieving data, get hypercube info and many more. Refer to this link (<https://help.qlik.com/en-US/sense-developer/3.1/Subsystems/APIs/Content/backend-api-reference.htm>) for more backend API functions.



Further reading

What is jquery and what is the benefits using jquery

<https://jquery.com/>

Requirejs is a powerful module loader for web development. Qlik sense uses requireJS for resource management,

<http://requirejs.org/>

Qlik extension backend API is the primary access to hypercube(data) manipulation and other backend related functions

<https://help.qlik.com/en-US/sense-developer/3.1/Subsystems/APIs/Content/backend-api-reference.htm>