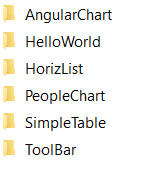
Take a look at Qlik sense sample extension.

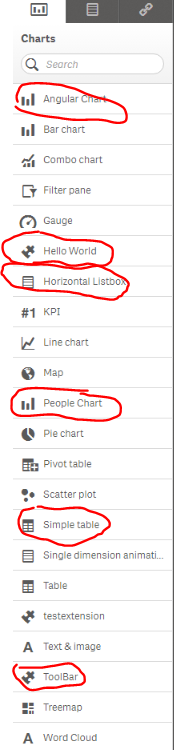
You should probably be very familiar with qlik sense but have you taken a look at the sample extensions? If you are about to build your first extension, these samples are arguably the most useful material to kick start your journey. These files are pre-installed with the software and you can find them in:

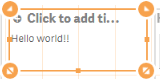
C:\Users\[your username]\Documents\Qlik\Examples\Extensions

There are seven folders here, these samples were designed to help developers learn the basic (but extremely useful) knowledge about extension. Copy and paste all the folders from the sample directory to your extension directory ( C:\Users\[your username]\Documents\Qlik\Sense\Extensions) and we are good to go!



Now let’s open the sense desktop. In this tutorial, I used helpdesk management sample app but you may use any data sets. You may found all the external extensions on the sidebar by now. Hmm where do we start…

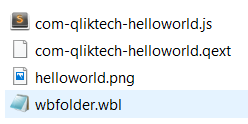


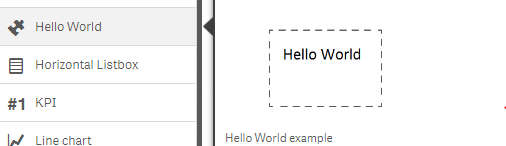


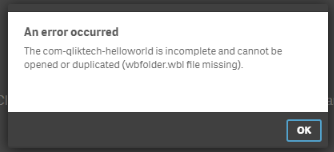
HelloWorld is always the first application a programmer builds so why don’t we start with the hello world extension. It does nothing than printing a single line of string.

Being the simplest, but fully functional extension, it is as clean as a working extension can be. By looking at the code, we will learn what are the necessary components and the minimal design of qlik sense extension.

There are four files in this folder. A js file, a qext file, a png file, and a wbl file.



The **wbl** file and png file are not mandatory but these are something good to have. The png file is for preview,🡺 ; the wbl file allows extension editor (http://localhost:4848/dev-hub/) to locate the neccessary components. If a wbl is missing, the following error will stop the developer from using editors in dev-hub. But the extension will work fine as long as js and qext files is present.



The js file and qext file are mandatory. The **QEXT** file is a json file that contains the metadata of the extension. Open com-qliktech-helloworld.qext, you will see the following properties:

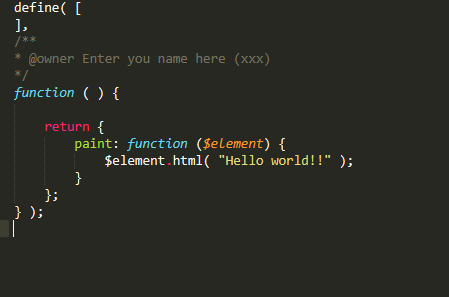


Below is a table I copied from <https://help.qlik.com/en-US/sense-developer/2.2/Subsystems/Extensions/Content/Overview/qext-file-overview.htm>. It is the official qlik help page for qext file. You may find detailed explanation about the extension properties here.

|  |  |  |
| --- | --- | --- |
| **Name** | **Options** | **Description** |
| **name** |  | **Mandatory**. Name of the visualization displayed in the library.  It is recommended to use an unique name of the visualization to avoid interference with other visualizations that may have the same name. |
| **type** |  | **Mandatory**. Should always be "visualization". |
| **description** |  | Description displayed in the library. |
| **icon** |  | Icon displayed in the library. Default is "extension". |
| **"extension"** | g |
| **"bar-chart-vertical"** | ! |
| **"line-chart"** | % |
| **"pie-chart"** | & |
| **"gauge-chart"** | ) |
| **"scatter-chart"** | + |
| **"text-image"** | / |
| **"table"** | ' |
| **"list"** | - |
| **"filterpane"** | . |
| **"treemap"** | » |
| **preview** | **"[CustomImage].png"** | Defines which preview image to be used. Preview image is displayed in a pop-up when you select the visualization in the library.  You can define a custom preview image file. It must be of .png file format. For an example of how this is defined, see [Hello world (extension example)](https://help.qlik.com/en-US/sense-developer/1.0/Subsystems/Workbench/Content/CodeExamples/ext_helloworld.htm).  If **preview** is undefined, the **icon** definition will be used. |
| **version** |  | Defines your individual version handling of the extension. This setting is manually defined. |
| **author** |  | Defines the author of the visualization. This setting is manually defined. |

The **js** file is the core of an extension. The name of js file and qext file must be identical to run the extensio. Qlik uses RequireJS as loader for extensions modules. The design and implementation of requireJs can be found here <http://requirejs.org/>.

In the array (which is empty in our helloworld example) we specify dependencies of our program and these dependencies will be passed to function as parameters. Methods defined in function will be executed, and the main program create/render extension object base on the properties returned. In this case, the one and only object is a function called paint. **Paint** function is the only mandatory object for an extension. What it does is…well…paint the extension. It is called whenever data changes or resizing happens. The main program passes two parameters to Paint function, $element and layout. In helloworld extension layout is omitted. We will talk about layout later. $element is a wrapper containing the html element where visualization should be rendered. In our code, $element.html replace the current displaying html with a new one.



For sense extension, there are two methods and two properties for developers to play with. Paint is one of them (may be the most important one), and there is also selectValues method, snapshot property and initialProperties property. We will see the rest in other samples.

|  |  |  |
| --- | --- | --- |
| Name | type | What is this |
| Paint | Function | Rendering logic, UI design and functional design for a extension |
| selectValues | Function | When this is called, the extension activates its selection mode and user will see this on top: |
| snapshot | Property | Enable or disable snapshot function(for story telling), this thing on top right hand corner: |
| initialProperties | Property | By modifying this property, the developer creates/implements variables for this extension. When I say implement, I mean: There are several predefined properties like dimension and measure we can refer to. We will talk about them later. |