Table of Contents

Architecture of the client-side code

Before we explain how to write scripts for different components it is useful if a developer understands roughly the general architecture of the client-side code.

This is what happens behind the scenes when a screen is displayed in the browser by ***Aware IM***. A screen (usually represented by visual perspectives) consists of multiple queries, forms, content panels and so on (we will call them “components”).

Each component, such as a form or a query is handled by the appropriate “controller” (also called “parser”). The fist thing the controller does is ask the server to provide the definition of the layout of the component. Then the controller parses XML returned by the server and prepares two Javascript objects:

1. jQuery object containing HTML of the component (form or query). This object is called “markup”.
2. An array of “widget configurations”. Each member of the array represents configuration for some widget of the Kendo UI library that the component includes. For example a query usually just includes a single widget that implements a query, such as the grid widget, for example. But a form may have a number of widgets – almost one per each attribute displayed by the form

Then the screen to be displayed is assembled from HTML markups of different components that the screen contains and the final HTML for the screen is built. This HTML is then given to the Kendo UI library, which creates all widgets of the screen based on the screen HTML and configurations of the widgets prepared by the controllers. Kendo UI library modifies this HTML to add its own classes and performs other steps to ensure that its widgets are displayed correctly. Finally the resulting HTML document is given to the browser which draws it on the screen.

To summarize:

1. A screen consists of components. Each component is represented by its own controller
2. The process starts by each controller asking the server for the definition of components
3. The controller then prepares HTML markup of the component and configurations of containing widgets
4. The HTML of the screen is created from the markups of components returned by controllers
5. The HTML and widget configurations are then given to the Kendo UI library to prepare its widgets
6. Final HTML of the screen is produced and is drawn by the browser.

So where in this process do the client scripts come in? For most components there are two types of scripts – the “initialization” script and the “render” script. As a developer you can define one or both – depending on what the script needs to do. The initialization script if defined is run by ***Aware IM*** just before the controller of the component returns the markup of the controller and the array of widget configurations (so immediately after step 3 above). The script, therefore, has a chance to modify the markup returned by the controller or the configuration of any of the widgets of the component.

The markup can be modified using jQuery functions that manipulate HTML. The script can only modify the markup for the component, but not the entire screen, because the entire screen hasn’t been built yet.

Widget configurations represent Javascript objects with properties described by Kendo UI API

Reference. For example, to see the API Reference of the Kendo UI grid widget that implements ***Aware IM*** queries in the standard form go to http://docs.telerik.com/kendo-ui/api/javascript/ui/grid and look up the Configuration section at the top. Note that you can only modify configuration of the widget (which also includes Events), but you cannot use Methods of the widget in the initialization script.

The render script, though, runs after everything has been drawn on the screen – i.e. after step 6 above. By this time all Kendo UI widgets will have been already created, so the script can access the widget and call its methods (see the Methods section in the Kendo UI API Reference for each widget). Configuration objects cannot be used at this stage.

The render script can also access the final browser document and manipulate it if need be using jQuery functions. The following sections describe how this can be done in more detail.