You can use custom javascript to implement nearly any interaction with objects, a canvas or the entire page. General functions that work globally:

```
/**
 * Get current register value by id
 * @param regId {int|string}
 * @returns {boolean|string}
 */
getRegisterValueById(regId)

/**
 * Get current register value by its variable
 * @param variable {string}
 * @returns {boolean|string}
 */
getRegisterValueByVariable(variable)
```

Functions for interaction with dashboard (in the case of screens, functions are performed in isolation):

```
/**
 * Creates canvas overlay
 * @returns {CanvasRenderingContext2D}
 */
createCanvas2DContext()

/**
 * Sets property to dashboard object.
 * @param id {int} required. You can find id in ObjectInspector or in object Properties tab (first row)
 * @param properties {Object} required. An object, contains properties to set {property: value, secondProperty: diffValue, ...}
 * Typical list of properties is: width, height, top, left, stroke, fill, hide, globalAlpha and so on
 * @param settings {Object|false} optional. An object, contains animation settings.
 * Possible keys are:
 *      duration {int} ms optional, default transition duration is dashboard refresh interval,
 *      onComplete {function} optional, callback function
 *      easing {string} optional, list: easeInQuad, easeOutQuad, easeInOutQuad, easeInCubic, easeOutCubic, easeInOutCubic, easeInQuart, easeOutQuart, easeInOutQuart, easeInQuint, easeOutQuint, easeInOutQuint, easeInSine
 */
setObjProperty(id, properties, settings)
```

Events observer can be implemented via callback functions. As for now handling available for 'alert', 'message' and general scan update `('register:newValue')` with `App.on(event, callback([{}, …]))` Most handy event `'register:newValue'` brings array of registers, changed during update interval as `[{regId: int, state: string, value: string}, …]`

Example:

```
var minValue = getRegisterValueById(74);
var maxValue = getRegisterValueByVariable('max');
var currentValue = getRegisterValueById(55);

function getSmoothTemperatureColor(){
    var delta = currentValue - minValue;
    return 'hsl(' + (150 - parseInt(delta)) + 'deg 50% 50%)';
}

if (minValue && maxValue)
    setObjProperty(12, {fill: getSmoothTemperatureColor()});

App.on('register:newValue', data => {
    let needle = data.filter(r => r.regId === 55);
    if (needle.length === 1) {
       currentValue = needle[0].value;
    }
    setObjProperty(12, {fill: getSmoothTemperatureColor()});
    if (currentValue >= maxValue) {
        // do some other stuff
    }
});
```