**(6/19/2021)**

# Programing QCSPCChart using Angular

Angular is an open-source web application framework promulgated by Google. It is used in conjunction with the TypeScript programming language, and HTML to define page templates. It is well documented online and you can start learning it here: AngularJS: Developer Guide: Introduction .

Since this software is written entirely in TypeScript (and transpiled into JavaScript) it is particularly useful for adding charts to Angular web applications. There are only a couple of small techniques you need to know.

An Angular web application is going to have at least one HTML file that is part of the project. The HTML file is considered a template and contains Angular specific elements that define the overall look of the web page. It is updated dynamically by the programming and database elements of the framework. QCSPCChart must be displayed in a HTML *canvas* element. So you need to add a canvas element to your Angular HTML template file, positioned where you want the SPC chart to appear. You can add as many canvas elements as you want and put unique SPC chart in each. Just keep the variable name (#chartCanvas1 in the example below) of the canvas elements unique. Something like this:

```
  <div id="buttondiv">
    <button type="button" (click)="buttonClickFunction()" >Display Chart</button>
</div>

<div id="canvasdiv">

    <canvas #chartCanvas1 width=800 height=600 ></canvas>
 </div>
```

In one of the introductory Angular examples you can download, this code was added to the body of the app.component.html file at the desired location. The button element is not really needed, we just added it to generate an event which in turn loads the chart. The canvas element is giving the variable name (which is sort of like an HTML ID, but accessible directly from the Angular TypeScript behind code) **#chartCanvas1** with a height and width of 800x600.

The TypeScript behind code for this template is the file app.component.ts. It will end up looking something like this. The **BuildXBarRChart** and **SimulateData** functions are exactly the same as similar function in all of our example programs.

```
import { Component, ViewChild, ElementRef, AfterViewInit } from '@angular/core';
import * as  QCSPCChartTS from '../../../QCSPCChartTS/qcspccharts.js';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent implements AfterViewInit {
  /** Template reference to the canvas element */
  @ViewChild('chartCanvas1', { static: true }) chartCanvas1!: ElementRef<HTMLCanvasElement>;
  title = 'Hello World!';

  private spcchart: QCSPCChartTS.SPCChartBase | null = null;

  constructor() {

  }
  buttonClickFunction() {
    this.BuildXBarRChart();
  }

  ngAfterViewInit() {

  }

  public BuildXBarRChart() {

    let htmlcanvas: HTMLCanvasElement | null = <HTMLCanvasElement>this.chartCanvas1.nativeElemen
t;
    let spccharttype: number = QCSPCChartTS.SPCControlChartData.MEAN_RANGE_CHART;
    let subgroupsize: number = 5;
    let numberpointsinview: number = 12;
    let charttitle: string = "XBar-R Example";

  let xbarrchart: QCSPCChartTS.SPCChartBase = QCSPCChartTS.SPCBatchVariableControlChart.newSPCBa
tchVariableControlChartChartTypeSubgroupSize(htmlcanvas, spccharttype, subgroupsize, numberpoint
sinview);

.
.
.

    // Rebuild the chart using the current data and settings
    xbarrchart.rebuildChartUsingCurrentData();


  }


  SimulateData(spcchart: QCSPCChartTS.SPCChartBase, count: number, mean: number, sigma: number)
{
    // batch number for a given sample subgroup
    let batchCounter = 0;
.
.
.

  }

}
```

The important Angular specific elements of this module are the following:

Make sure that the import statement points to a directory location where the QCSPCChartTS/qcspccharts.js library file can be found.

```
import * as  QCSPCChartTS from '../../../QCSPCChartTS/qcspccharts.js';
```

Include this statement in the class member variable section, referring back to the variable name *chartCanvas1* in the canvas element you placed in the app1.component.html file.

```
  @ViewChild('chartCanvas1', { static: true }) chartCanvas1!: ElementRef<HTMLCanvasElement>;
```

When you actually build the SPC chart (making calls into our library) you will need to convert this variable name into the underlying HTML canvas element (type HTMLCanvasElement) using code similar to this:

```
    let htmlcanvas: HTMLCanvasElement | null = <HTMLCanvasElement>this.chartCanvas1.nativeElemen
t;
```

The resulting *htmlcanvas* TypeScript variable can then be used to pass the canvas element into the one of the many newSPCBatchVariable... (or other) constructors.

```
 let xbarrchart: QCSPCChartTS.SPCChartBase =

QCSPCChartTS.SPCBatchVariableControlChart.newSPCBatchVariableControlChartChartTypeSubgroupSize(h
tmlcanvas, spccharttype, subgroupsize, numberpointsinview);
    if (!xbarrchart) return;
```

Assuming you have a standard global  Angular development environment installed, you can run it by going to the command line, going into the angular example folder where the project is, starting a development server by running "ng serve" from the root of the project folder. Then point your web browser to "http://localhost:4200/" and the test page should appear. As with most development environments, make sure you can run your application before you start trying to add the charts to it.

You will find the Angular specific files discussed in this section (app.component.html, app.component.ts) in the TypeScriptExamplesQCSPCChart/AngularExampleCode/src/app folder. The overall Angular project (a very simple one) ends up so huge we did not want to burden the download with the entire project.