Programing QCSPCChart using NextJS (Next.js)

Introduction

NextJS is an open-source web application framework promulgated by the American cloud services company Vercel. NextJS is built on top of the React Javascript library and adds many features that developers find useful in web app programming. NextJS and React are both well documented online and you can start learning it here: NextJS, and React. The upside is that you can develop websites with sophisticate UI very quickly using these tools. The downside is that they have a steep learning curve for the average HTML/JavaScript programmer, and they are moving targets because they are constantly changing.

Truth in Advertizing: We are not experts at programming NextJS. We are writing this programming guide because a customer of ours had a requirement that SPC charts be displayed in a NextJS app. Not being familiar with NextJS, we looked into it, and with the help of the customer were able to display the SPC charts in a simple NextJS app. The discussion that follows is based on that interaction. It is not necessarily the only way to utilize the QCSPCChartTS library with NextJS, nor the best, nor the most efficient. But it seems both fast loading and reliable.

NextJS can be used with both JavaScript and TypeScript, though 99% of the examples on the web are for JavaScript. Using it TypeScript requires that you monkey around with the *.json files defining the project settings, and we didn't get into that. So this example assumes you are using JavaScript.

NextJS normally generates HTML pages on the server side of the web application and sends them to the client browser. But HTML libraries which use the HTML 5 canvas element, such as QCSPCChartTS, are setup for server-side rendering and must be rendered directly on the browser. There are special features of NextJS that you need to use to restrict canvas rendering to within the browser. Also, because of the asynchronous nature web browsers, you must import (load) the QCSPCChartTS library in a special way to make sure that it is completely loaded before you start making calls into the library.

Install the NextJS-blog example project from the NextJS website

First, start with the standard first project tutorial described by NextJS on their website. You will find that here: Create a Next.js App. If you don't already have NextJS installed, you will need to follow all of the steps in the Create a Next.js App example. It's pretty well automated with npm commands and easy to do. Once NextJS is installed you can also use npm to install the apps project. Ultimately you end up with a folder named **nextjs-blog** on your development computer. Under that folder you will have more folders representing different elements of the application. You may find that you have an old versions of npm, react and next.js installed on your development computer and have to update them to a newer version. *Once you have the app installed, and can compile and run it,* then you are ready to make the modifications necessary to utilize the QCSPCChartTS library to display SPC Charts.

The nextjs-blog example, being the most basic example of the NextJS framework, only has a single page. This is represented by the index.js file in the /pages folder. If you develop a multi-page application your /pages folder will most likely have a separate *.js file for every page. While these files are JavaScript files, they serve to define HTML templates for their associated page. So that when a function is called (Home in the index.js file), it returns an HTML template for the web page. The objective is to insert an SPC chart within the HTML template.

Modify the NextJS-blog example with our files

Once you can run the nextjs-blog example, download the following three files from our website:

index.js – replace the index.js file in the projects /pages folder
HelloSPC.js – place in the projects /public folder.
QCSPCChart_NextJS.pdf – this document.

You can download them from our website at:
https://quinn-curtis.com/downloadsoftware/qcspcchart_nextjs01.zip .

You must also place a copy of our qcspcchartts.js file in the projects /public folder, next to the HelloSPC.js file from the download. You will find the original qcspcchartts.js file to copy from in the Quinn-Curtis/JSTS/QCSPCChartTS folder. You can use either commercial version of the library, or the trial version.

As with most of our example programs, an SPC chart is defined in an external JavaScript (or TypeScript) file, keeping the chart creation code mostly segregated from the HTML layout code. In this case the external JavaScript file name is HelloSPC.js. This file is found in the projects /public folder, along with some files that were already part of the project. External JavaScript files such as this should be placed in the /public folder, and not the /pages folder. The HelloSPC.js file must be loaded into the browser when the parent HTML page is loaded, ruling out preprocessing on the server-side. This is accomplished using the built-in NextJS component <Script>. While <Script> is similar to the standard HTML <script> tag, it has some added features which apply specifically to NextJS applications. These features include delayed loading of the referenced JavaScript file (called strategy), and onLoad, onReady and onError events to trigger. So you will want to read about those in the NextJS <Script> documentation.

Starting with the /pages/index.js file from the default installation of the nextjs-blog example, we add the following lines of code, highlighted in red, in the <main> section (directly above the line “<div className={styles.grid}” :

```
import Head from 'next/head';
import styles from '../styles/Home.module.css';
import Script from 'next/script';


export default function Home() {
  return (
    <div className={styles.container}>
      <Head>
        <title>Create Next App</title>
        <link rel="icon" href="/favicon.ico" />
      </Head>


      <main>
        <h1 className={styles.title}>
          Welcome to <a href="https://nextjs.org">Next.js!</a>
        </h1>
        <p className={styles.description}>
          Get started by editing <code>pages/index.js</code>
        </p>
      <div id="buttondiv">
        <button type="button" id="xbarr_menuitem">XBar-R</button>
      </div>

    <div id="canvasdiv" >
        <canvas  id="spcChartCanvas1" width="800" height="600"></canvas>
     </div>

       <Script
        src='/HelloSPC.js'
        strategy='afterInteractive'
        onLoad={() =>
          {
          BuildXBarRChart("spcChartCanvas1");
          //  xbarrchart is a parameterless function found in the loaded src file.
          document.getElementById("xbarr_menuitem").onclick = xbarrchart;
          console.log(`script loaded correctly, window.FB has been populated`)
          }
        }
        onError={(e) => {
            console.error('Script failed to load', e)
          }
        }
        />

        <div className={styles.grid}>
          <a href="https://nextjs.org/docs" className={styles.card}>
            <h3>Documentation &rarr;</h3>
            <p>Find in-depth information about Next.js features and API.</p>
          </a>
.
.
.
.
```

The “buttondiv” section defines an HTML button that is used to trigger an update of the chart. The “canvasdiv” section defines the HTML canvas element where the SPC chart will be placed. The canvas element is given the name “spcChartCanvas1”, and a size. The name of this canvas element is later passed into the chart building function, so that it can place the chart in the canvas. The <Script> component will delay loading the HelloSPC.js file until the time determined by the *strategy* prop, in this case “afterInteractive”, which mean after the interactive elements of the page have been rendered in the browser. Then the source file for the Script (src=/HelloSPC.js) will be loaded right before the onLoad event is called. Inside the onLoad event the SPC chart is built by calling the function BuildXBarChart, which is a function in the HelloSPC.js file. The “afterInteractive” value for strategy is the default for the <Script> component so you can explicitly set it or leave it out. Other settings may not, or won’t work. Note: you must place an import for Script at the top of the file, see the example.

The HelloSPC.js chart building file is almost identical to the one from our standard JavaScript example HelloSPC. The only change is in the way the QCSPCChartTS.js library is imported. Our standard JavaScript and TypeScript examples all use what is known as an ES6 import,

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

This won’t work from a file loaded inside of <Script>. It complains about the loaded code not being a module. This may be fixable by adjusting the code in some way, but a simpler variant does work. A few things need to be changed. First the import is changed to a dynamic import looking like this:

```
QCSPCChartTS = await import('/qcspccharts.js');
```

the default location of our library, in folder QCSPCChartTS, is not accessible in this application. That is because the NextJS server running at localhost:3000 assumes that the nextjs-blog folder is the root directory and it will not permit file access beyond the root location for security reasons. So instead you must copy the QCSPCChartTS/qcspcchartts.js file from the QCSPCChartTS folder and into the nextjs-blog/public folder, next to the HelloSPC.js file. Then you access the file using a file location of /qcspcchartts.js, as in the example.

It is important that the import statement uses the *wait* modifer. Otherwise the imported library will be loaded asynchronously with everything else that is going on. This will cause an error, because calls will start to made into the library before it finishes loading. So you must use await, which makes sure that execution on subsequent lines of code do not happen before the library finishes loading. Since the import method uses *await*, it must be called inside of a function marked *async*, which in this case is the BuildXBarRChart function. So that is where to place the import, before any calls to the qcspcchartts.js library are made. In order to make the QCSPCChartTS namespace globally available we made QCSPCChartTS a var at the top of the file and assign it a value of *null*. Then it is reassigned to the imported qcspcchartts.js library when it is imported inside the BuildXBarRChart function.

```
var xbarrchart;
var overallStats;
var sampleIntervalStats;
var QCSPCChartTS = null;

async function BuildXBarRChart(canvasid) {

     QCSPCChartTS = await import('/qcspcchartts.js');

     if (canvasid == null) return;
    var htmlcanvas = document.getElementById(canvasid);
    var spccharttype = QCSPCChartTS.SPCControlChartData.MEAN_RANGE_CHART;
    var subgroupsize = 5;.
    var numberpointsinview = 12;
     var charttitle = "XBar-R Example";
.
.
.
```

Note how the function obtains the HTML canvas element from the canvas id string passed in as canvasid. It does this by calling the HTML document function getElementById. If this code was being preprocessed on the server, as a lot of NextJS code is, the document element would not be present, because it is part of the client-side browser, so the function would fail, and everything would fail. But, by placing the SPC chart creation in a NextJS <Script> component, the execution of the code is delayed until a browser is present, and the corresponding document element available.

From that point on, everything seems to work as documented in our manual.

Summary

In summary, in order to make use of this example you must do the following:

1. Install the Quinn-Curtis QCSPCChart for JavaScript/TypeScript on your computer. It can be either the commercial version or the trial version. It should end up in a folder name Quinn-Curtis with the JavaScript/TypeScript specific stuff in the sub folder JSTS. You should probably follow our own

example at the end of the user manual and run one or more examples for JavaScript. Once you can do that you are ready to combine it with NextJS.

2. Go to the NextJS tutorial located here: Create a Next.js App. Install the nextjs-blog example app in our subfolder JSTS. You will need to use some sort of cmd window with executive privileges in order to do that. Study the NextJS tutorial until you can get the nextjs-blog app to run. You run and test it by pointing a browser to the location http://localhost:3000, which is the location of the NextJS test server on your computer.

3. Once you can run the nextjs-blog example, download the following three files from our website:

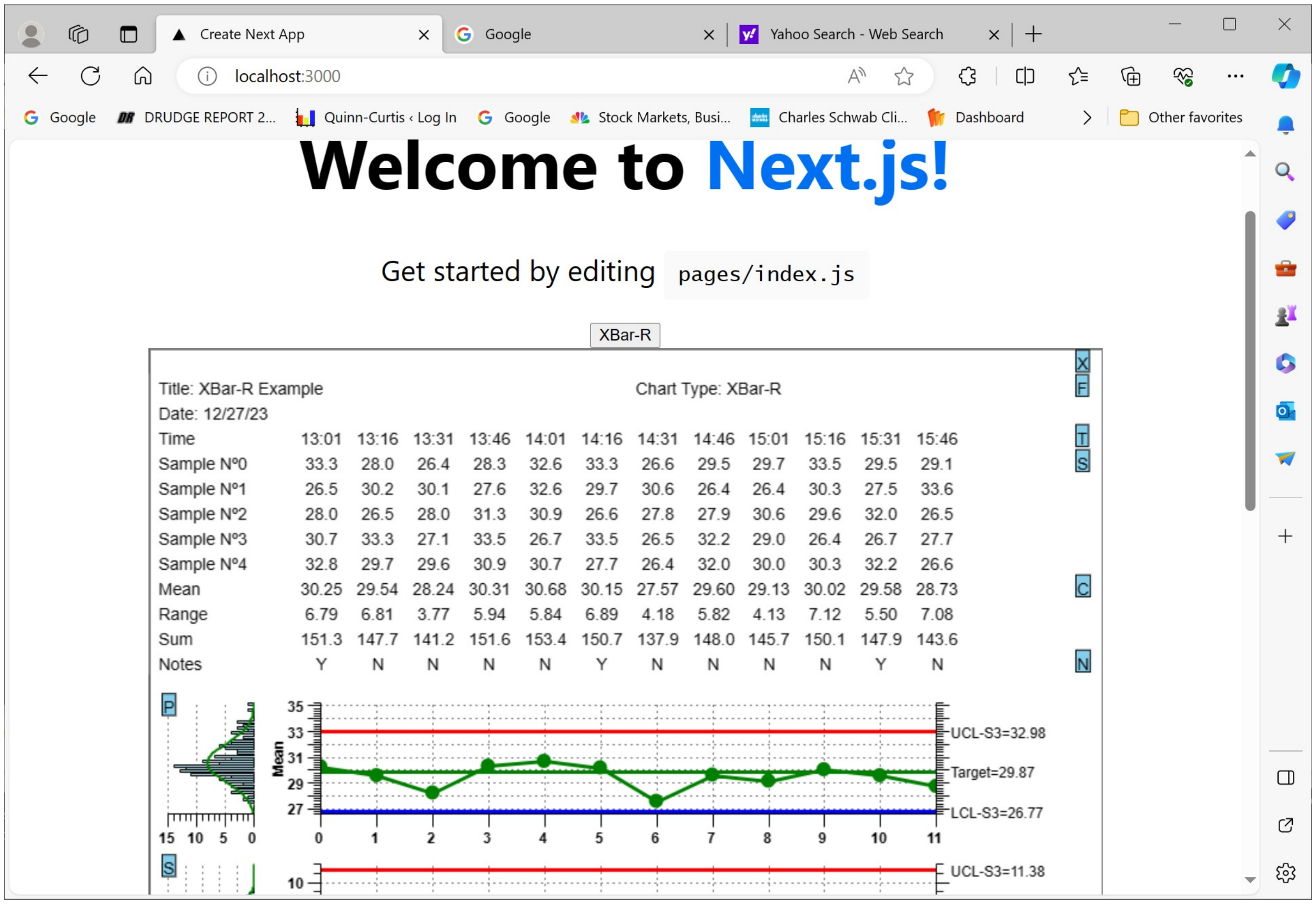
index.js – replace the index.js file in the projects /pages folder
HelloSPC.js – place in the projects /public folder.
QCSPCChart_NextJS.pdf – this document.

You can download them from our website at:
https://quinn-curtis/downloadsoftware/qcspschart_nextjs01.zip.

You must also place a copy of our qcspcchartts.js file in the projects public folder, next to the HelloSPC.js file from the download. You will find the original qcspcchartts.js file to copy from in the Quinn-Curtis/JSTS/QCSPCChartTS folder.

Special note

If you have a better way of importing external JavaScript (such as our HelloSPC.js chart building code, and the qcspcchartts.js library) into NextJS apps and are willing to share it, send us an email.



Welcome to Next.js!

Get started by editing `pages/index.js`

Title: XBar-R Example   Chart Type: XBar-R

Date: 12/27/23

Time	13:01	13:16	13:31	13:46	14:01	14:16	14:31	14:46	15:01	15:16	15:31	15:46
Sample Nº0	33.3	28.0	26.4	28.3	32.6	33.3	26.6	29.5	29.7	33.5	29.5	29.1
Sample Nº1	26.5	30.2	30.1	27.6	32.6	29.7	30.6	26.4	26.4	30.3	27.5	33.6
Sample Nº2	28.0	26.5	28.0	31.3	30.9	26.6	27.8	27.9	30.6	29.6	32.0	26.5
Sample Nº3	30.7	33.3	27.1	33.5	26.7	33.5	26.5	32.2	29.0	26.4	26.7	27.7
Sample Nº4	32.8	29.7	29.6	30.9	30.7	27.7	26.4	32.0	30.0	30.3	32.2	26.6
Mean	30.25	29.54	28.24	30.31	30.68	30.15	27.57	29.60	29.13	30.02	29.58	28.73
Range	6.79	6.81	3.77	5.94	5.84	6.89	4.18	5.82	4.13	7.12	5.50	7.08
Sum	151.3	147.7	141.2	151.6	153.4	150.7	137.9	148.0	145.7	150.1	147.9	143.6
Notes	Y	N	N	N	N	Y	N	N	N	N	Y	N