Tutorial 02 - Creating a Chart with scichart-react

In this tutorial we're going to show how to create a JavaScipt Chart in React using scichart.js and **scichart-react**.

> The previous tutorial Tutorial 01 - Understanding the scichart-react boilerplate should serve as a reference. We will be using this boilerplate (npm setup, webpack configuration and package.json) as a starting point for the tutorial.
>
> Go ahead and copy the boilerplate to a new folder or project. You can get the code from here: Boilerplates/scichart-react

The <SciChartReact/> React component

Props and Configuration

`<SciChartReact />` has the following props which you can use to configure a chart.

Name	Description
fallback	ReactNode a React component that would be rendered while the chart is being initialized
onInit	a callback function used after the chart is initialized
onDelete	a callback function used when the component with initialized chart is unmounted

innerContainerProps	TDivProps props passed to the inner container container where the chart is hosted
initChart	An initialization function which passes an HTMLDivElement and should return surface instance created on the provided root element.
config	A string with chart definition or configuration object acceptable by SciChart Builder API
style	Optional style that may be passed to the outer element created by <SciChartReact/>

Initializing a chart with `<SciChartReact />` is simple. At a minimum you must pass either an **initChart** function using the programmatic JavaScript API to SciChart, e.g. SciChartSurface.create(), or a **config** object using the JSON Builder API.

You can specify a **fallback** ReactNode which will be shown while the chart is initializing. You can pass **onInit** and **onDelete** callbacks which can be used to setup data

`<SciChartReact />` will create the where the chart resides. An optional **style** may be passed to size or position the outer div. **innerContainerProps** are passed

DOM Outputted by <SciChartReact/>

Each `<SciChartReact />` element outputs three divs. The DOM created by `<SciChartReact />` looks like this.

Example Title Copy Code

```
<div> <!-- style property is applied here -->
   <div> <!-- innerContainerProps property applied here -->
       <div id="scichart-root"> <!-- Chart hosted here -->
   </div>
</div>
```

To set these properties, try something like this:

Example Title Copy Code

```
<SciChartReact
  initChart={initChartFunc} // (divElementId) => { return { sciChartSurface } };
  onInit={onInitFunc} // (initResult) => console.log(`surface: ${initResult.sciChartSurface.i
  onDelete={onDeleteFunc} // (initResult) => console.log(`surface: ${initResult.sciChartSurfa
  innerContainerProps={{ style: { width: "100%"}}}
  style={{ maxWidth: 900, height: 600 }}
```

```
/>
```

Next we're going to dig into the function signatures for **initChart**, **onInit** and **onDelete** as well as show how **innerContainerProps** and style affect the chart layout.

Creating a scichart-react Chart using initChart

So let's create our first chart using scichart-react and **initChart** function.

Go ahead and copy the Boilerplates/scichart-react folder into a new project. Change your App.jsx as follows:

APP.JSX REACT COMPONENT

```
import React from "react";
import { SciChartReact } from "scichart-react";

function App() {
  // SciChart.js will work out of the box with a community license.

  // For commercial licenses (to remove the watermark), set your license code here
  // SciChartSurface.setRuntimeLicenseKey("YOUR_RUNTIME_KEY");

  // to use WebAssembly and Data files from CDN instead of the same origin
  // SciChartSurface.loadWasmFromCDN();

  return (
    <div className="App">
      <header className="App-header">
        <h1>&lt;SciChartReact/&gt; with initChart Tutorial</h1>
      </header>
      <SciChartReact
        initChart={initChart}
        onInit={onInit}
        onDelete={onDelete}
        innerContainerProps={{ style: { width: "100%" } }}
        style={{ maxWidth: 900, height: 600 }}
      />
    </div>
  );
}

export default App;
```

The `<SciChartReact />` component renders a single scichart chart. **initChart** is the function which is called to setup your chart. We'll show the code for this in the next step. **onInit** and **onDelete** are optional callbacks on mount/unmount of the react component. Finally **innerContainerProps** allows you to style the which contains the chart, while **style** is

applied to the where the scichart chart is hosted.

Here's the code required to initialize the chart, and the optional onInit and onDelete functions:

APP.JSX INITCHART

```
import {
  SciChartSurface,
  SciChartJsNavyTheme,
  NumericAxis,
  FastLineRenderableSeries,
  XyDataSeries,
  EllipsePointMarker,
  ZoomPanModifier,
  MouseWheelZoomModifier,
  ZoomExtentsModifier,
  WaveAnimation,
  SweepAnimation,
} from "scichart";

// Called to initialize the chart. rootElement is passed in which is the <div> that will hos
// Create the SciChartSurface, add axis, series, data, annotations etc.
// return { sciChartSurface } to <SciChartReact /> to be used in onInit, onDelete
const initChart = async (rootElement) => {
  const { sciChartSurface, wasmContext } = await SciChartSurface.create(
    rootElement,
    {
      id: "New SciChart Chart",
      theme: new SciChartJsNavyTheme(),
      title: "SciChart-React with initChart",
      titleStyle: { fontSize: 16, color: "White " },
    }
  );
  sciChartSurface.xAxes.add(
    new NumericAxis(wasmContext, { axisTitle: "X Axis" })
  );
  sciChartSurface.yAxes.add(
    new NumericAxis(wasmContext, { axisTitle: "Y Axis" })
  );

  // Add some series and data
  sciChartSurface.renderableSeries.add(
    new FastLineRenderableSeries(wasmContext, {
      dataSeries: new XyDataSeries(wasmContext, {
        xValues: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
        yValues: [
          0, 0.0998, 0.1986, 0.2955, 0.3894, 0.4794, 0.5646, 0.6442, 0.7173,
          0.7833,
        ],
      }),
      stroke: "SteelBlue",
      pointMarker: new EllipsePointMarker(wasmContext, {
        fill: "LightSteelBlue",
```

```
        stroke: "White",
        size: 9,
      }),
      animation: new SweepAnimation({ duration: 750 }),
    })
  );

  // Add some interactivity modifiers
  sciChartSurface.chartModifiers.add(
    new ZoomPanModifier({ enableZoom: true }),
    new MouseWheelZoomModifier(),
    new ZoomExtentsModifier()
  );

  return { sciChartSurface };
};

const onInit = (initResult) => {
  // You can get the sciChartSurface, wasmContext here to perform any initialization
  const sciChartSurface = initResult.sciChartSurface;
  const wasmContext = sciChartSurface.webAssemblyContext2D;

  console.log(
    `SciChartSurface has been initialized: id=${sciChartSurface.id}, divElementId=${sciChart
  );
};

const onDelete = (initResult) => {
  // You can get the sciChartSurface, wasmContext here to perform any cleanup
  const sciChartSurface = initResult.sciChartSurface;
  const wasmContext = sciChartSurface.webAssemblyContext2D;

  console.log(
    `SciChartSurface with id=${sciChartSurface.id} is deleted = ${sciChartSurface.isDeleted}
  );
};
```

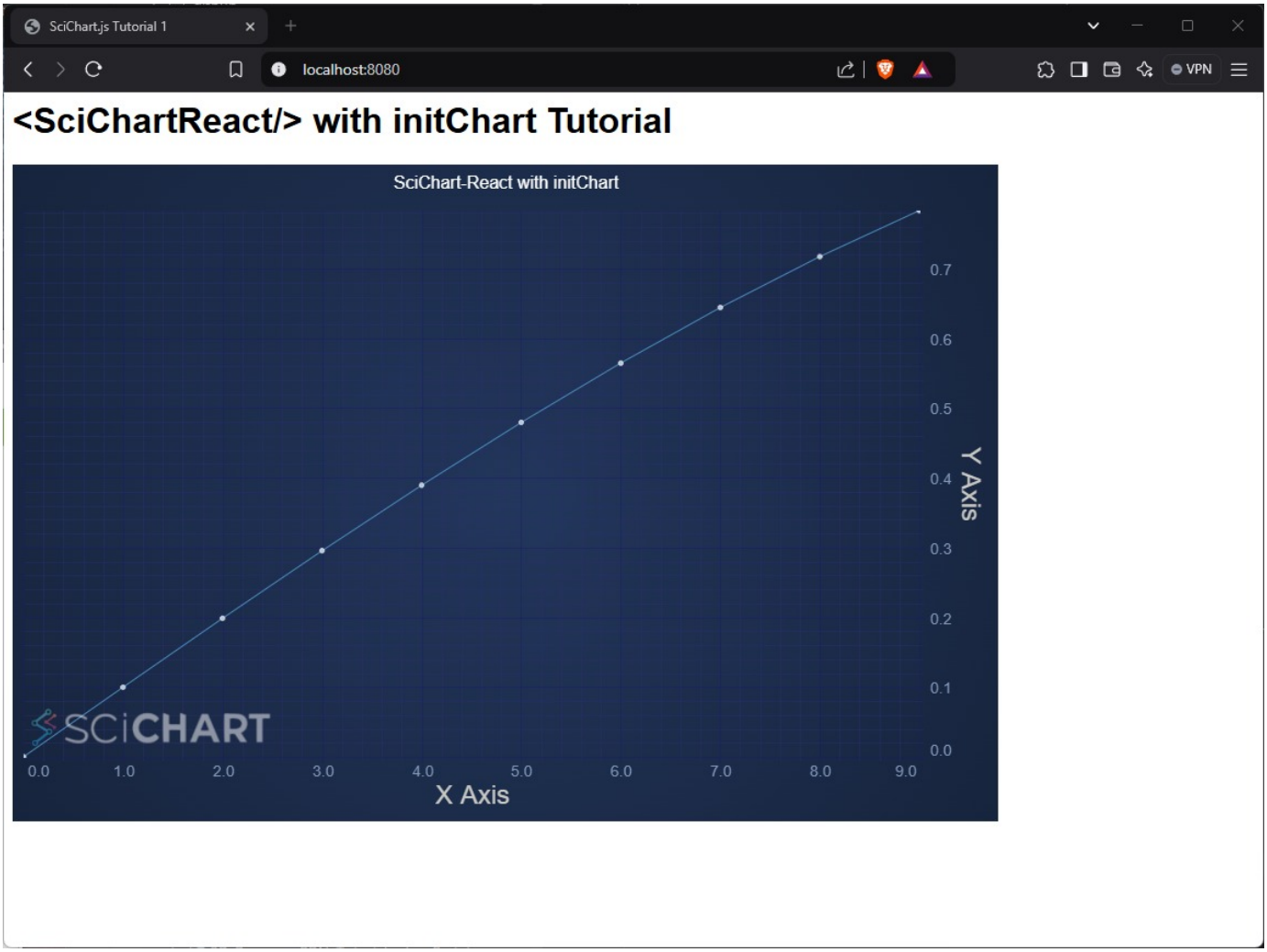
initchart: is an async function and has a single parameter which is the rootElement (where the chart is hosted). This is created by `<SciChartReact />` and will be given a unique ID. In this function you can call SciChartSurface.create(), add XAxis and YAxis, add RenderableSeries with data, and add interactivity modifiers.

onInit: is an optional function which has one parameter - initResult. This is called directly after component mount and allows you to access the sciChartSurface and it's wasmContext via *sciChartSurface.webAssemblyContext2D*. You can also access the HTMLDivElement which hosts the chart via *sciChartSurface.domChartRoot*.

onInit: is an optional function is called on component unmount. Use this to perform any chart specific cleanup such as disconnecting to data-sources.

Running the code

Let's run the code and see the result!



You can get the full source code for this tutorial over at scichart.js.examples on Github under the folder Tutorials/React/Tutorial_02_Creating_Charts_SciChart_React_initChart

A note about SciChartSurface.delete():

`<SciChartReact />` automatically calls *sciChartSurface.delete()*, ensuring that all wasm memory is disposed on component unmount. This cascades and calls *dataSeries.delete()* on all data currently in the chart. If you dynamically add/remove series however you will need to delete these as you go along. For more info see the article on Deleting DataSeries Memory.