04 Animated bar chart

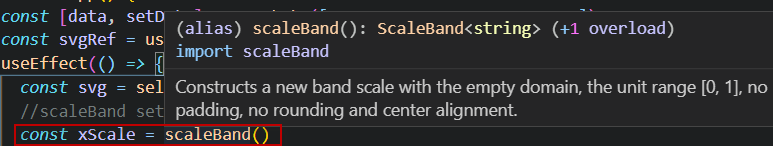
<https://www.youtube.com/watch?v=LQHt0wr3ybw&t=45s>

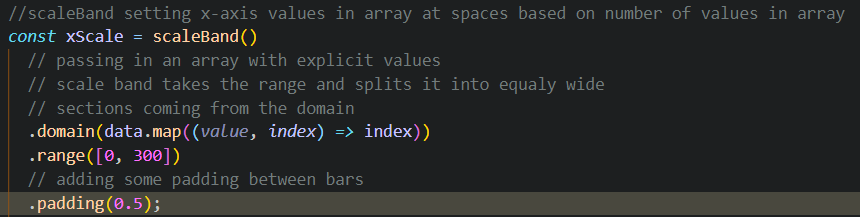
Animates when page loads , and with changes

SVG is centered and responsive

Move from line to bar chart

New imports are scaleBand



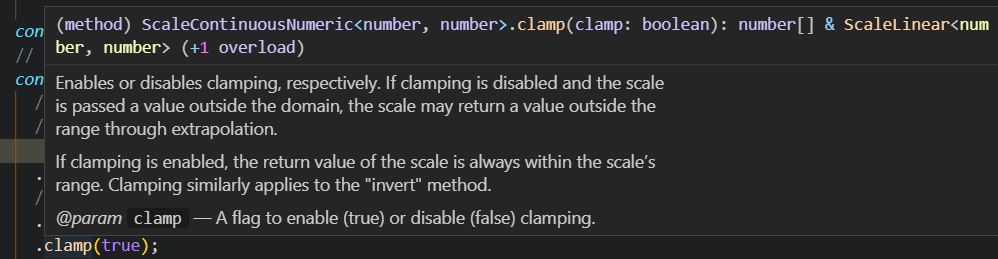


ScaleLiniear vs scaleBand –

With scaleBand – mapping arbitrary values in domain to a set of values in the range, have to be explicit in the domain.

With scaleLinear- maps a linear range of input values to a linear range of output values

Clamp



[xyz](https://www.youtube.com/channel/UCb-PfEzCNQkX5nb_iaoCx5w)[3 weeks ago](https://www.youtube.com/watch?v=LQHt0wr3ybw&lc=UgyNOzDzgY8wh40pKJh4AaABAg)

Isn't there a downside of using 'selectAll' and the likes in combination with react? I heard that it's not good if both take care of the dom updates?

REPLY

Hide 2 replies

[The Muratorium](https://www.youtube.com/channel/UCKfcSawDV88REF9jVwqqbag)

[The Muratorium](https://www.youtube.com/channel/UCKfcSawDV88REF9jVwqqbag)

[3 weeks ago](https://www.youtube.com/watch?v=LQHt0wr3ybw&lc=UgyNOzDzgY8wh40pKJh4AaABAg.96eqbTACLcg96ex-iKI_OV)

xyz No, there shouldn't be. In these examples, React only cares about the svg element being rendered. Everything inside that svg is d3 territory. selectAll is always used on the svg-selection, that means we never select stuff outside of it, and that is why it is scoped and fine.

REPLY

[The Muratorium](https://www.youtube.com/channel/UCKfcSawDV88REF9jVwqqbag)

[The Muratorium](https://www.youtube.com/channel/UCKfcSawDV88REF9jVwqqbag)

[3 weeks ago](https://www.youtube.com/watch?v=LQHt0wr3ybw&lc=UgyNOzDzgY8wh40pKJh4AaABAg.96eqbTACLcg96exAfKyeQu)

You can however make React handle everything in that svg too, for that I would suggest you watch / read articles from Amelia Wattenberger.

<https://wattenberger.com/>

her git hub repo

<https://github.com/Wattenberger>

The code from this vis

import React, { useRef, useEffect, useState } from "react";

import "./App.css";

import { select, axisBottom, scaleLinear, axisRight, scaleBand } from "d3";

*function* App() {

*const* [data, setData] = useState([25, 30, 45, 60, 10, 65, 75]);

*const* svgRef = useRef();

  useEffect(() *=>* {

*const* svg = select(svgRef.current);

    //scaleBand setting x-axis values in array at spaces based on number of values in array

*const* xScale = scaleBand()

      // passing in an array with explicit values

      // scale band takes the range and splits it into equaly wide

      // sections coming from the domain

      .domain(data.map((*value*, *index*) *=>* index))

      .range([0, 300])

      // adding some padding between bars

      .padding(0.5);

*const* yScale = scaleLinear().domain([0, 150]).range([150, 0]);

    // the colorScale is just another linear scale

*const* colorScale = scaleLinear()

      // maps liniear range of input values to linear range of output values

      // so 75 is green, 100 orange, 150 red

      .domain([75, 100, 150])

      // change output values to colors

      .range(["green", "orange", "red"])

      // use clamp to keep the colors green orange and red

      .clamp(true);

*const* xAxis = axisBottom(xScale).ticks(data.length);

    svg.select(".x-axis").style("transform", "translateY(150px)").call(xAxis);

*const* yAxis = axisRight(yScale);

    svg.select(".y-axis").style("transform", "translateX(300px)").call(yAxis);

    svg

      // select all the elements with a class of .bar

      .selectAll(".bar")

      // sync with data array

      .data(data)

      // create a rect element for every piece of data

      .join("rect")

      // attach the class of bar to every entering and updating element

      // in the svg

      .attr("class", "bar")

      // set the origin of the y scale to the origin

      // flips the y axis

      .style("transform", "scale(1, -1)")

      //possition on the x axis, need the index value and to pass it to the xScale

      .attr("x", (*value*, *index*) *=>* xScale(index))

      // set to origin of the fixed value of the negative total hight

      .attr("y", -150)

      // add a width to the rectangles

      // make it the bandwidth of the xScale

      .attr("width", xScale.bandwidth())

      // put transition here so the colors also anniate smoothly

      .transition()

      // pass in the colorScale

      .attr("fill", colorScale)

      // set the height of the bars

      .attr("height", (*value*) *=>* 150 - yScale(value));

  }, [data]);

  return (

    <React.Fragment>

      <svg ref={svgRef}>

        <g className="x-axis"></g>

        <g className="y-axis"></g>

      </svg>

      <br />

      <br />

      <br />

      <button onClick={() *=>* setData(data.map((*value*) *=>* value + 5))}>

        Update data

      </button>

      <button onClick={() *=>* setData(data.filter((*value*) *=>* value < 35))}>

        Filter data

      </button>

    </React.Fragment>

  );

}

export default App;