Programmatically Drawing Graphics in the Web Browser

Group G3
Valerio Mariani, Christoph Söls, Sebastian Überreiter
11 May 2022

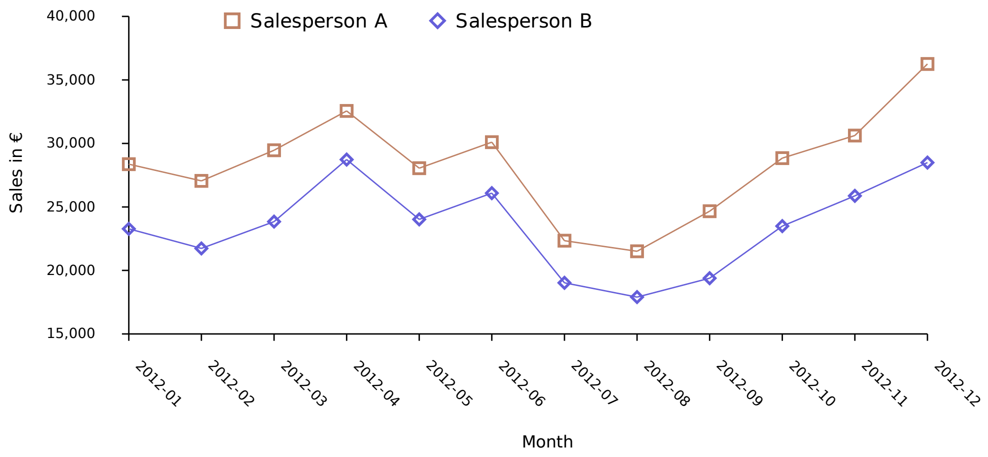
Introduction

- Built in graphics technologies
  - Canvas2D, SVG-DOM, WebGL
- Drawing libraries
  - PixiJS: WebGL, Canvas2D
  - Two.JS: WebGL, Canvas2D, SVG
  - D3: SVG-DOM

Dataset

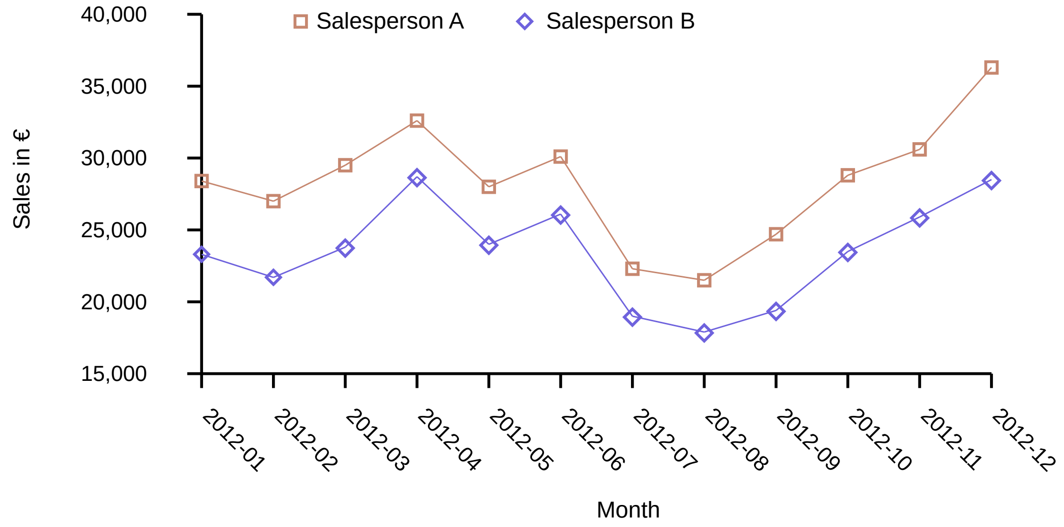
Reproduce the same chart with the different methods

Month	Salesperson A	Salesperson B
2012-01	28366	23274
2012-02	27050	21732
2012-03	29463	23845
2012-04	32561	28732
2012-05	28050	24023
2012-06	30100	26089
2012-07	22343	19026
2012-08	21506	17903
2012-09	24664	19387
2012-10	28842	23490
2012-11	30621	25873
2012-11	36254	28490



SVG

- Specification for two dimensional vector graphics
- Based on XML
- Vector graphics allow for scalability, but many elements in the same chart slow down the computation
- Time consuming on change

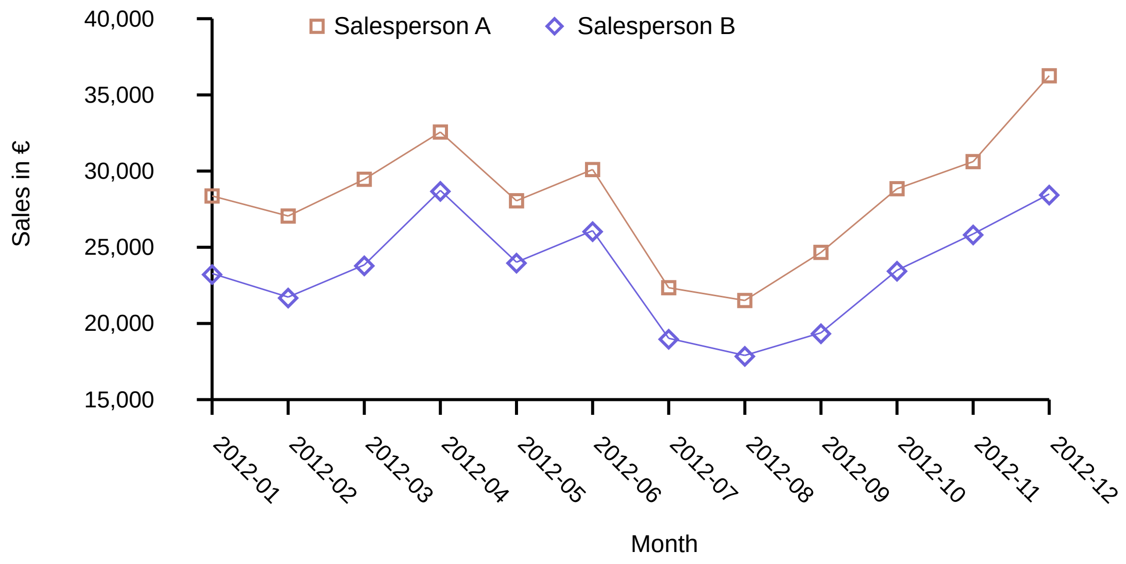


SVG-DOM

- http://sitepoint.com/surviving-the-zombie-apocalypse-manipulating-svg-with-javascript/
- JavaScript used to create and manipulate elements in the DOM.
- With JavaScript, numbers do not need to be hand coded.

SVG-DOM

- How to get started
    - Create an SVG doc or element
    - Define the viewbox
    - Use JavaScript and tags to draw graphics
- Basic workflow
    - <line...></line> tag to draw lines
    - <text...></text> tag for text
    - <g...></g> tag for common properties
- JavaScript allows for variables and loops, that you can not use with spurious SVG.



SVG-DOM Code Snippets

Initialise SVG

```
const svg = document.createElementNS('http://www.w3.org/2000/svg', 'svg');
      svg.setAttribute("viewBox", "0 0 " + view_box.width + " " + view_box.height + "");
      svg.setAttribute("width", "100%");
      svg.setAttribute("height", "100%");
```

Draw a line

```
const x_marking = document.createElementNS('http://www.w3.org/2000/svg', 'line');
```

Set attributes

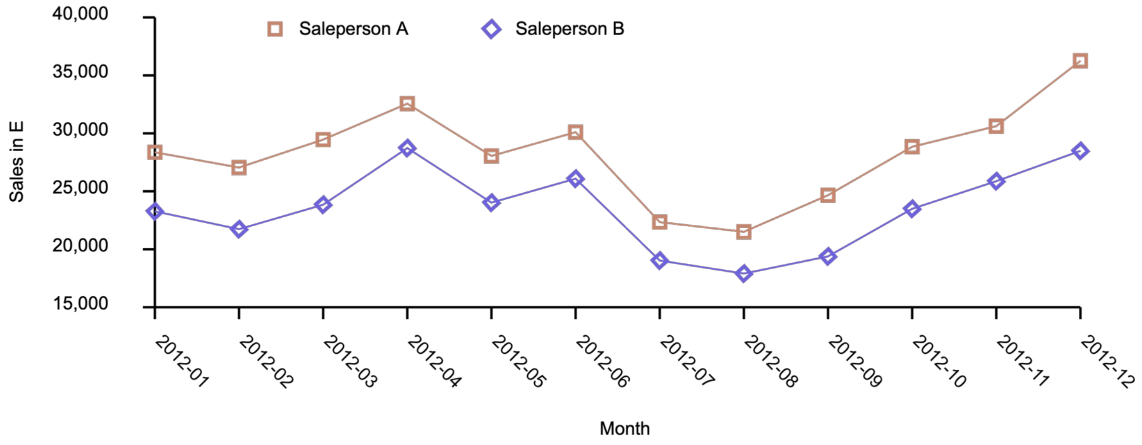
```
x_marking.setAttribute("x1", "x");
x_marking.setAttribute("x2", "x");
x_marking.setAttribute("y1", size.height + margin);
x_marking.setAttribute("y2", size.height + margin + 10);
```

Canvas2D

- https://w3.org/TR/2dcontext/
- HTML<canvas> element with context “2d”
- Use JavaScript interface
- Raster-based graphics allow fast, one-time only computation of the chart but does not automatically scale.

Canvas2D Workflow

- How to get started
  - Grab a canvas element
  - Extract the 2d object context
  - Use the exposed interface
- Basic workflow
  - Create array of points
  - BeginPath()
  - MoveTo() through the points in the array
- No need to hand code values
- Need to write a function to scale values and make them fit the ViewBox.



Canvas2D Code Snippets

Retrieve the “2d” context

```
var canvas = document.getElementById("chart")
var ctx = canvas.getContext("2d");
```

Draw lines

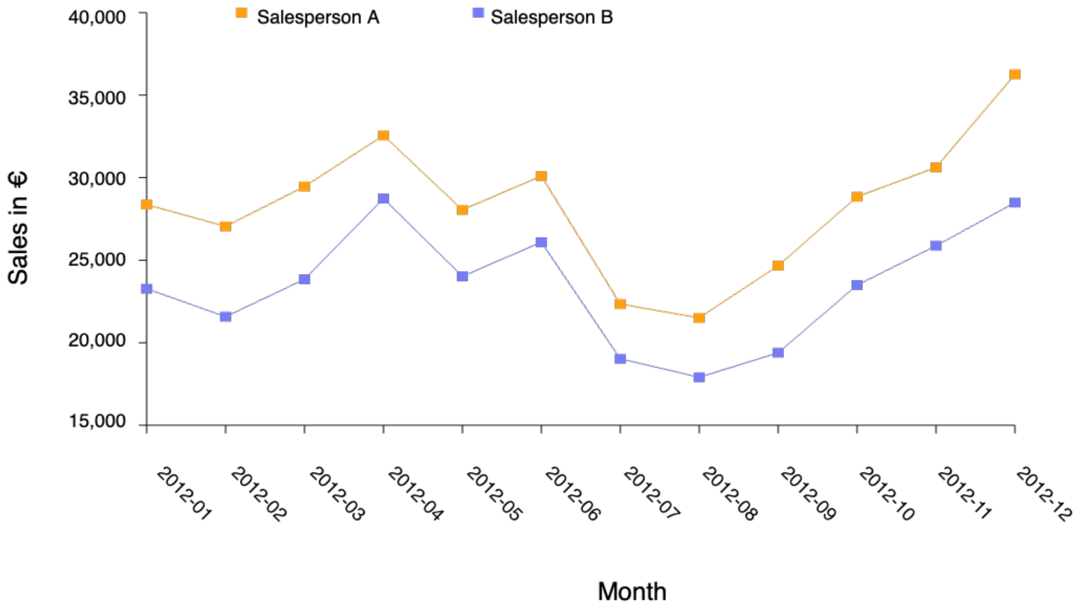
```
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x1, y1);
ctx.stroke();
```

WebGL

- https://khronos.org/webgl/wiki/
- JavaScript API for rendering 2D and 3D graphics in the web browser.
- Based on OpenGL
- Hardware accelerated → much faster than other libraries.
- GLSL language for shaders
- WebGL renders into HTML <canvas> element

WebGL Workflow

- Hard to get started
  - Introduction needed
  - Understanding of shaders + colors
  - Bind everything
- Basic workflow
  - Create array of points
  - Create + bind buffer
  - Drawarrays() with different parameters



WebGL Text Problem

- WebGL has no text API – must use workaround
    - Create Bitmap for every letter in alphabet (Difficult solution).
    - Draw letters by rendering lines (Vector Font).
    - Overlay second canvas (most-common solution).

WebGL Code Snippets

Create a point and bind to a buffer and call drawfunction

```
Function drawPointMarkerA(){
        let vertices = new Float32Array([
        -0.48, 0.5]);
        createBindBuffer(vertices);
        gl.drawArrays(gl.POINTS, 0, 1);
}
```

Function to bind point array to the buffer

```
createBindBuffer(array){
        let vertexBuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
        gl.bufferData(gl.ARRAY_BUFFER, array, gl.STATIC_DRAW);

        program.position = gl.getAttribLocation(program, ‘position’);
        gl.enableVertexAttribArray(program.position);
        gl.vertexAttribPointer(program.position, 2, gl.FLOAT, false, 0, 0);}
```

Comparison of Basic Drawing Technologies

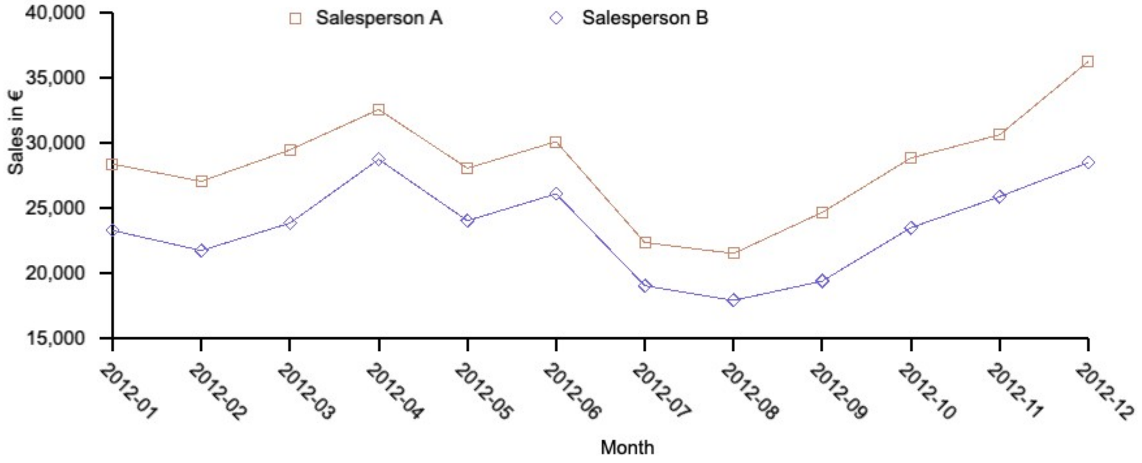
	SVG	Canvas2d	WebGL
Rendering technique	Vector-based graphics	Raster-based graphics	Raster-based graphics
Language	XML	JavaScript	JavaScript + OpenGL
Text API	yes	yes	no
Hardware acceleration	no	no	yes

PixiJS

- https://pixijs.com
- 2D WebGL rendering Library
- Uses canvas or WebGL to render
- No prior knowledge of WebGL needed
- Seamlessly falls back to Canvas

PixiJS Workflow

- Easy to get into
  - “Drawing” like with a pencil
  - Text capabilities
  - Fast creation/ simple
- Basic workflow
  - Create ViewBox + data array
  - Move around the graph and select coordinates where from/ to draw.
- PIXI’s text API
  - “The text is created using the Canvas API.”[1]



[1] https://pixijs.download/dev/docs/PIXI.Text.html

PixiJS Code Snippets

Initialise PIXI and append to document

```
let chart = new PIXI.Application({ width: view_box.width, height: view_box.height,
backgroundColor: 0xffffff});
document.body.appendChild(chart.view);
let graph = new PIXI.Graphics();
```

Set line style

```
graph.lineStyle(2, 0x000000);
```

Draw a line

```
graph.moveTo(margin,margin);
graph.lineTo(margin,margin + size.height);
```

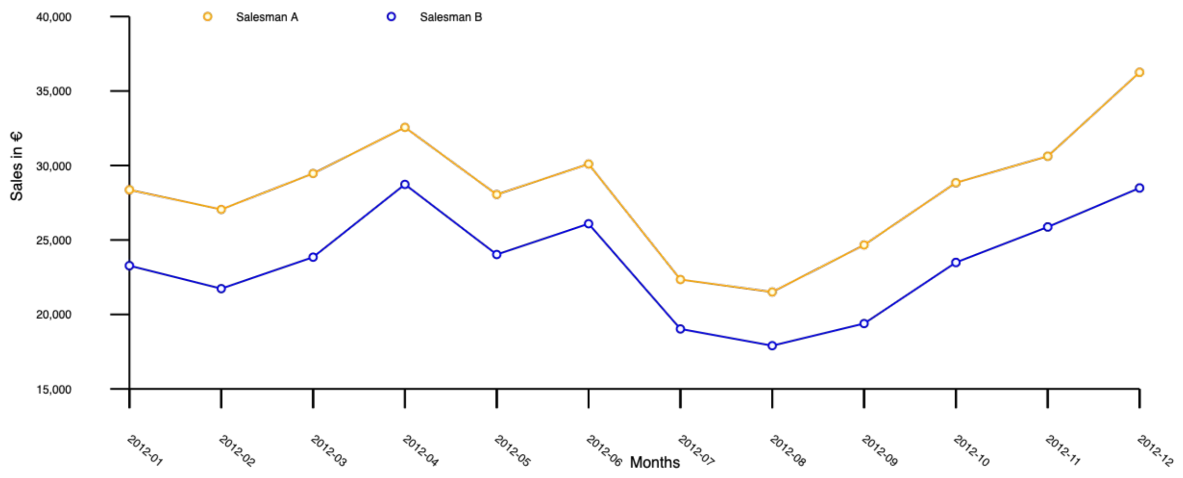
Two.JS

- https://two.js.org
- Two-dimensional drawing API
- Two.JS uses SVG-DOM
- “It is renderer agnostic enabling the same api to draw to multiple contexts: SVG, Canvas2D and WebGL”[2]

[2]: https://two.js.org/

Two.JS Workflow

- Easy to use
  - Drawing with coordinates
  - Text API
- Basic workflow
  - Create instance of Two
  - Set parameters for instance
  - Start drawing with instance
- Text API
  - Create “Shape” in form of text



Two.JS Code Snippets

Initialise Two.JS for drawing

```
let view_box = {width: 950, height:500};
    let canvas =
document.getElementById('chart');
    let two = new Two({
      fullscreen: true,
      width: view_box.width,
      height:view_box.height,
      autostart: true,
      type: Two.Types.canvas
    }).appendTo(canvas);
```

Draw line in Two.JS

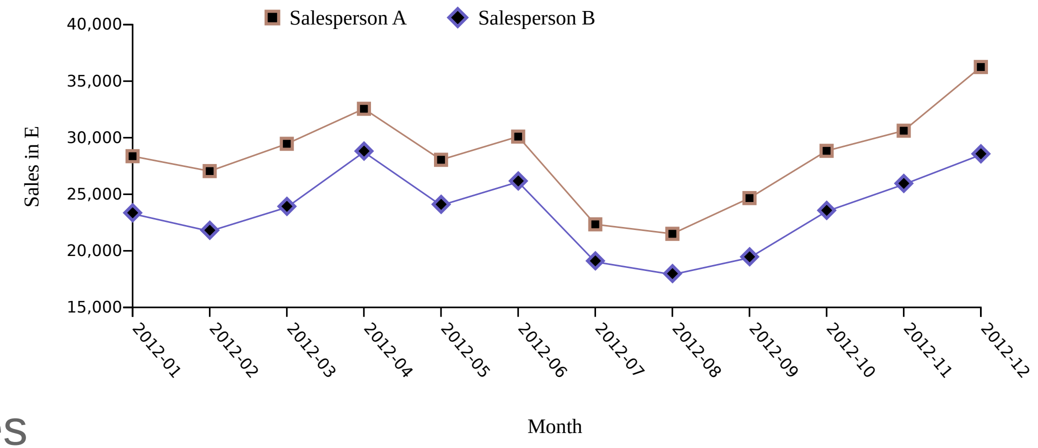
```
two.makeLine(two.width * 0.2,
two.height * 0.7, two.width * 0.75,
two.height * 0.7)
```

D3

- https://d3js.org
- Most widely used drawing library.
- Document manipulation with JavaScript interface.
- Bind document’s elements with the data, allowing for dynamic document creation.
- Can use selectAll() to change properties of blocks of code.

D3 Workflow

- How to get started
  - Grab SVG element
  - Define the ViewBox
  - Use the D3 interface to draw graphics
- Basic workflow
  - Possibly translate the origin
  - Define x and y axis scales
  - Define line() function
- Automatically scales the values through the d3.axis.
- If the values change or new values need to be plotted, the chart is automatically re-rendered properly.



D3 Code Snippets

Retrieve the SVG element and initialise the code

```
const svg = d3.select("#chart")
  .append("svg")
  .attr("viewBox", "0 0 " + xSize + " " + ySize);
```

Draw a line

```
svg.append("path")

  .attr("class", "line")

  .attr("d", line(points_a));
```

Comparison of Drawing Libraries

	PixiJS	Two.JS	D3
Render Technology	WebGL, Canvas2D	WebGL, Canvas2D, SVG	SVG
Text API	yes	yes	yes
Automatic computation of scaled values	no	no	yes
Data binding	no	no	no