Rendering Large Visualizations with WebGL

Before switching to WebGL:

- Doubling SVG Performance at Khan Academy:
http://www.crmarsh.com/svg-performance/

- Speeding up D3.js: A Checklist
https://blog.safaribooksonline.com/2014/02/20/speeding-d3-js-checklist/

- Think of other ways to represent your data that do not involve tens of thousands of DOM elements

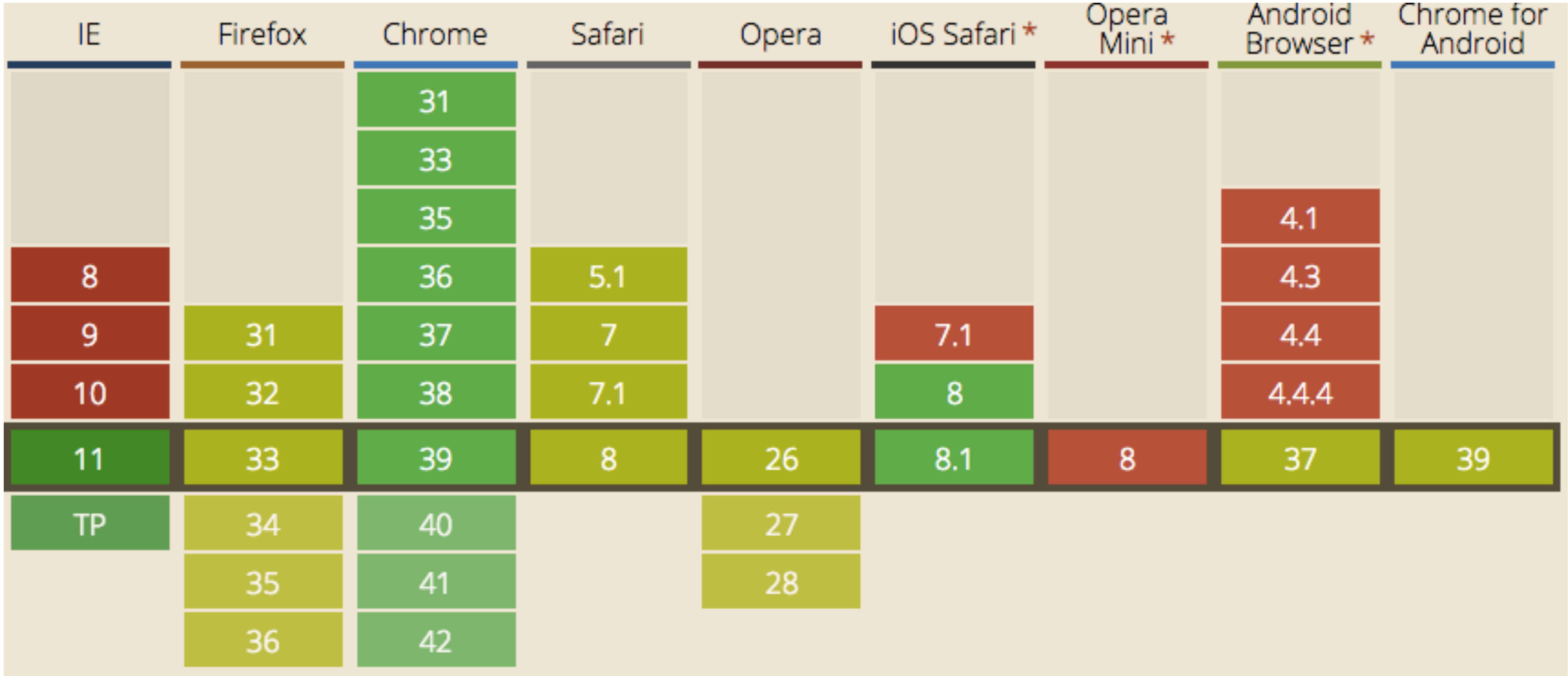
WebGL goodness

- better performance for large visualizations via GPU acceleration

- better performance for the rest of your DOM
(transitions render faster; more control over repainting frequency)

WebGL badness

- browser support (getting better…)



IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		33						
		35					4.1	
8		36	5.1				4.3	
9	31	37	7		7.1		4.4	
10	32	38	7.1		8		4.4.4	
11	33	39	8	26	8.1	8	37	39
TP	34	40		27				
	35	41		28				
	36	42						

caniuse.com/webgl

WebGL badness

- code overhead

- with a d3 stack, we can create a network that zooms, pans, makes node selections, and changes the node colors in about 140 lines of code

- with pixi.js (a 2D WebGL API), we need 160 lines of code, plus 400 from our Grapher API, plus the additional dependency of pixiJS (190kb unminified)

- With sigma.js (a WebGL API for networks), we need about 160 lines of code and the additional dependency (85kb unminified)

- your CSS has no power in canvas land

What can be salvaged from D3?

D3	pixi.js v2 Made by goodboy sigmajs
• philosophy	• create API for enter / exit / update
• zoom & pan	• API exposes transform function that takes a transform object
• brush	• with a vacant SVG DOM element, let D3 brush calculate extent, API exposes selection FN (slightly hacky)

- using webGL does not guarantee faster results
  - drawing circles and line primitives through PIXI’s graphics API results in performance similar if not worse than SVG

- one way to increase performance is generating your sprites and putting them in sprite batch containers

- another is to use vertex shaders (sigma.js)

Compare & Contrast

4363 nodes, 32039 links	**SVG**	**Grapher** (on top of Pixi.js)	**SigmaJS**
Transform (translate & zoom)	4-5 FPS	50-60fps	50-60FPS
Changing all node & link colorings	0.05s to complete	0.15s to complete	0.01s to complete

Tested on 2013 Macbook Pro

Processor: 2.8 GHZ Intel Core i7

Memory: 16 GB 1600 MHz DDR3

Try it yourself

- github.com/dannycochran/d3meetup
- github.com/ayasdi/grapher

- Explaining grapher
    - First, explain what a network is and what functionality we want
    - Go through the init functions
        - Explain hacky brush solution
    - Go though the render functions
    - Explain why we build textures:
        - Less network traffic & faster
        - Need unique color 36 colors, and
    - Explain how zoom works
    - Take a look at Grapher.js
        - 54 textures for nodes and links
        - Explain dependencies
        - Explain how sprite batches work – they are sort of like SVG group elements, but they define the color of their children
        - Enter: creates sprites
        - Update: finds appropriate batch for sprite, adds it there