Data Visualization using **D3.js**

Jesús Alejandro Valdés Valdés
Philipp Müller

What is data visualization?

Definition **data**:

“facts and statistics collected together for reference or analysis.”

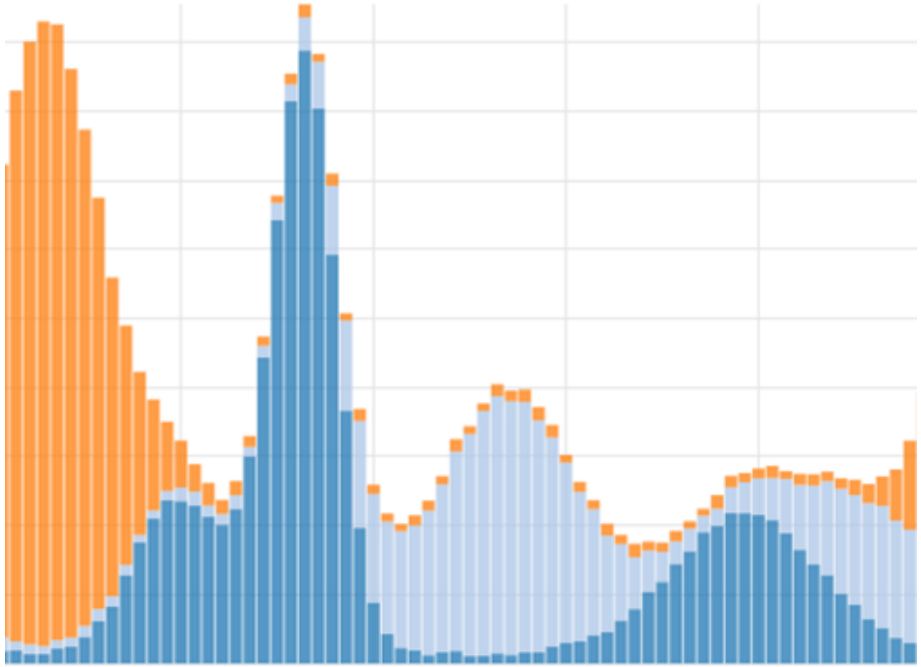
Definition **visualization:**

“(..)is any technique for creating images, diagrams, or animations to communicate a message.(...)”

Why is it important?

	0	1	2	3	4	5
677	0.8247666...	0.8247666...	0.8247666...	0.8247666...	0.8247666...	0.8263400...
678	0.8497600...	0.8497600...	0.8497600...	0.82134	0.82134	0.82134
679	0.854025	0.854025	0.854025	0.8252599...	0.8252599...	0.8252599...
680	0.9061	0.9061	0.9061	0.82685	0.82685	0.82685
681	0.9061	0.9061	0.9061	0.82685	0.82685	0.82685
682	0.8865666...	0.8865666...	0.8865666...	0.8586	0.8586	0.8586
683	0.8488083...	0.8488083...	0.8488083...	0.8488083...	0.8488083...	0.8488083...
684	0.84902	0.84902	0.84902	0.84902	0.84902	0.84902
685	0.8575666...	0.8575666...	0.8575666...	0.8575666...	0.8575666...	0.8575666...
686	0.8740749...	0.8740749...	0.8740749...	0.8740749...	0.8740749...	0.8740749...
687	0.8760199...	0.8760199...	0.8760199...	0.8760199...	0.8760199...	0.8760199...
688	0.876925	0.876925	0.876925	0.876925	0.8838545...	0.8838545...
689	0.8844200...	0.8844200...	0.8844200...	0.8844200...	0.86289375	0.86289375
690	0.8839454...	0.8839454...	0.8839454...	0.8839454...	0.8762222...	0.8762222...
691	0.8599916...	0.8599916...	0.8599916...	0.8599916...	0.8612	0.8612
692	0.8060333...	0.8060333...	0.8060333...	0.8060333...	0.8060333...	0.8060333...
693	0.8665200...	0.8665200...	0.8665200...	0.8665200...	0.8665200...	0.8665200...
694	0.8944571...	0.8944571...	0.8944571...	0.8944571...	0.8944571...	0.8944571...
695	0.8699833...	0.8699833...	0.8699833...	0.8699833...	0.8699833...	0.8699833...
696	0.8541333...	0.8541333...	0.8541333...	0.8541333...	0.8541333...	0.8766733...
697	0.8600277...	0.8600277...	0.8600277...	0.8600277...	0.8600277...	0.8403083...
698	0.8589399...	0.8589399...	0.8589399...	0.8589399...	0.8589399...	0.8123428...
699	0.8756999...	0.8756999...	0.8756999...	0.8756999...	0.8756999...	0.8756999...
700	0.9310999...	0.9310999...	0.9310999...	0.9310999...	0.9310999...	0.9310999...
701	0.9102909...	0.9102909...	0.9102909...	0.9102909...	0.9102909...	0.9102909...
702	0.9132888...	0.9132888...	0.9132888...	0.9132888...	0.9132888...	0.9132888...
703	0.8996833...	0.8996833...	0.8996833...	0.8996833...	0.8996833...	0.8996833...
704	0.8620368...	0.8620368...	0.8620368...	0.8620368...	0.8620368...	0.8620368...
705	0.8731578...	0.8731578...	0.8731578...	0.8731578...	0.8731578...	0.8731578...
706	0.8673000...	0.8673000...	0.8673000...	0.8673000...	0.8673000...	0.8673000...

VS



Data visualization libraries

- D3.js
- Raphäel.js
- vis.js
- Paper.js
- Chart.js
- Cytoscape.js
- and many more...

How can we visualize data in a browser?

HTML elements

+Supported on any browser

-Rigid format

-Simple shapes

Canvas

+Faster for many objects (>1000)

-Undefined behaviour

-No events and callbacks per element

SVG

+Flexible

+DOM handling

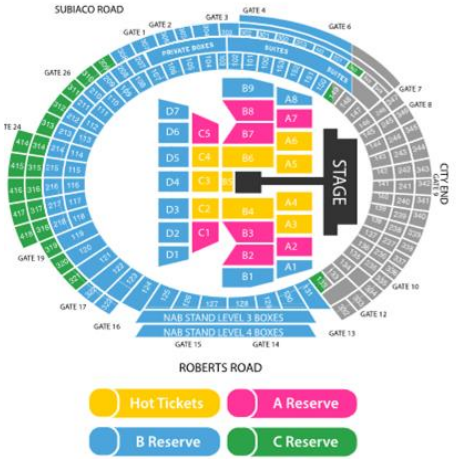
+Events and callbacks

+Resolution independent

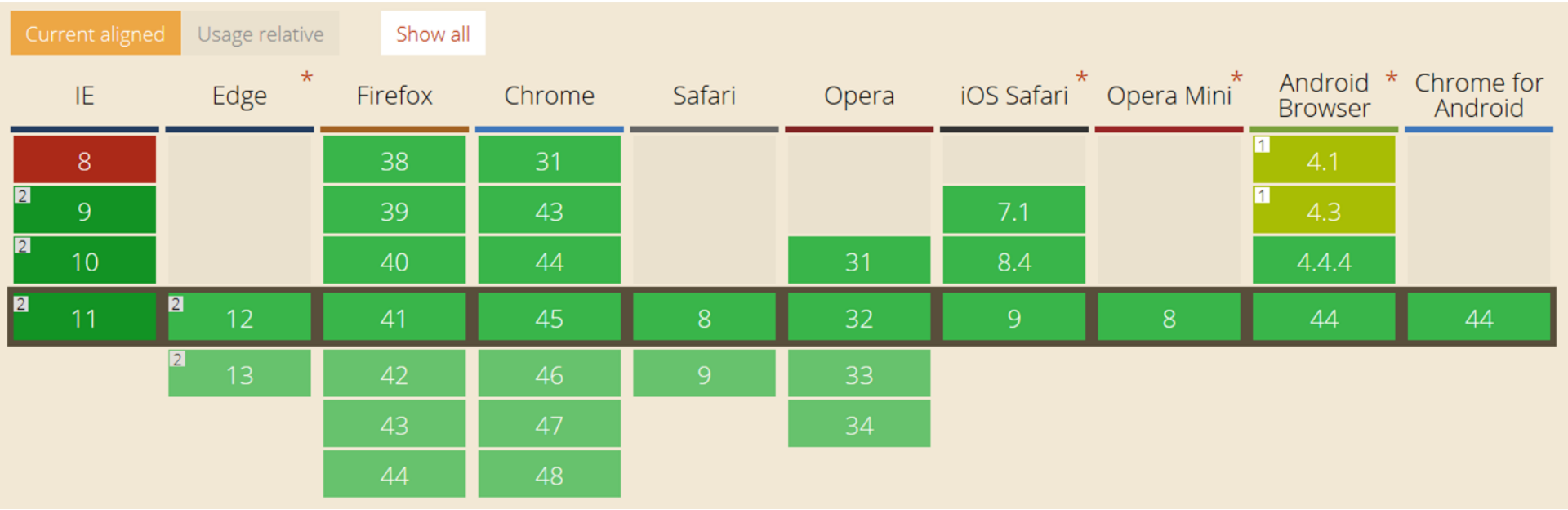
-Slower for many objects (>1000)

SVG

- **S**calable **V**ector **G**raphics
- W3C
- 2D
- Transformation and Animation
- Text or drawing software



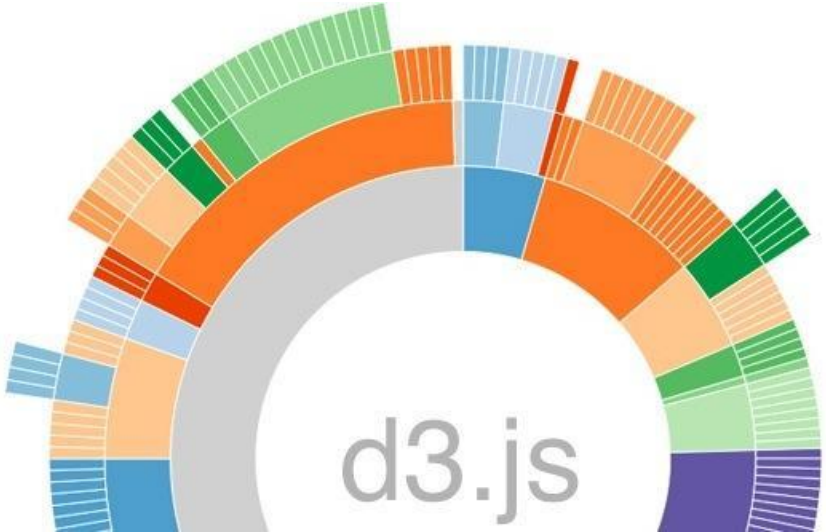
SVG - Basic support in browsers



➢ SVG basic functionality supported in ~95%.

D3

Data **D**riven **D**ocuments.
Developed by Mike Bostock 2011 [1]
Formerly known as Protovis
DOM data binding



How can we get data?

- d3.csv
- d3.htm
- d3.jsor
- d3.tsv
- d3.xml
- d3.xhr

```
var dataEx;
var apiEx = "http://api.openweathermap.org/data/2.5/weather?q=munich";

d3.json(apiEx, function(error, json) {
  if (error) return console.warn(error);
  dataEx = json;
  console.log(dataEx);
});

d3.xhr(apiEx)
    .responseType('json')
    .get( function( error, data ) {
        if ( error ) alert('error');
        console.log(data.response);
} );
```

D3 Selectors

- An array of elements from the document
- CSS3 selectors
- Select by
  - tag (“div”)
  - class (“.class”)
  - id (“#id”)
  - attribute (“[color=red]”).
- Selectors can also be...
  - ...intersected via logical AND: ".this.that"
  - or unioned via logical OR: ".this, .that"

D3 Selectors

- d3.select(selector): Selects the first element that matches the selector
- d3.selectAll(selector): Selects all elements that match the selector

```
d3.select(".foo")
Array [ Array[1] ]
d3.selectAll(".foo")
Array [ Array[3] ]
```

```
<div class="foo">class 1</div>
<div class="foo">class 2</div>
<div class="foo">class 3</div>
```

D3 Selectors - Comparison

W3C - **W**orld **W**ide **W**eb **C**onsortium

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "white", null);}
```

D3 (Selections)

```
d3.selectAll("p").style("color", "white");
```

Joins

- Tell D3 that the selection corresponds to data.

```
var values = [0,2,4,...];
// OR
var values = function() { ... return [0,2,4,...]; }
selection.data([values[, key]]);
// key is optional comparison function
```

- Base for enter, update and exit methods.
- No ifs or fors needed

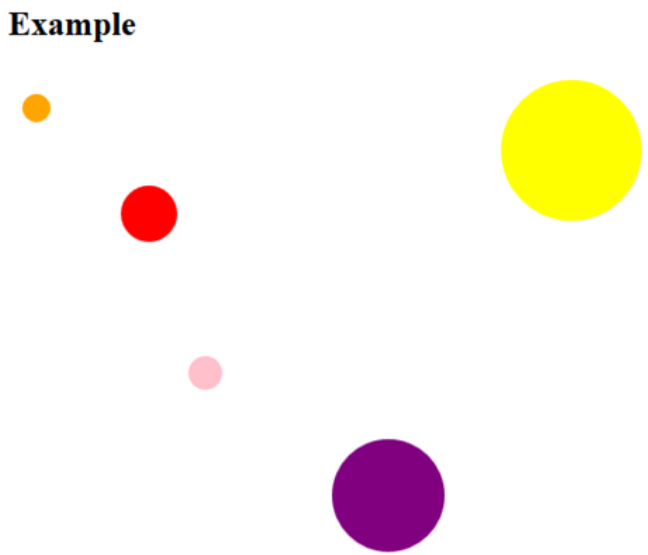
D3 - Declarative Approach

"Tell it what to do, not how to do it."

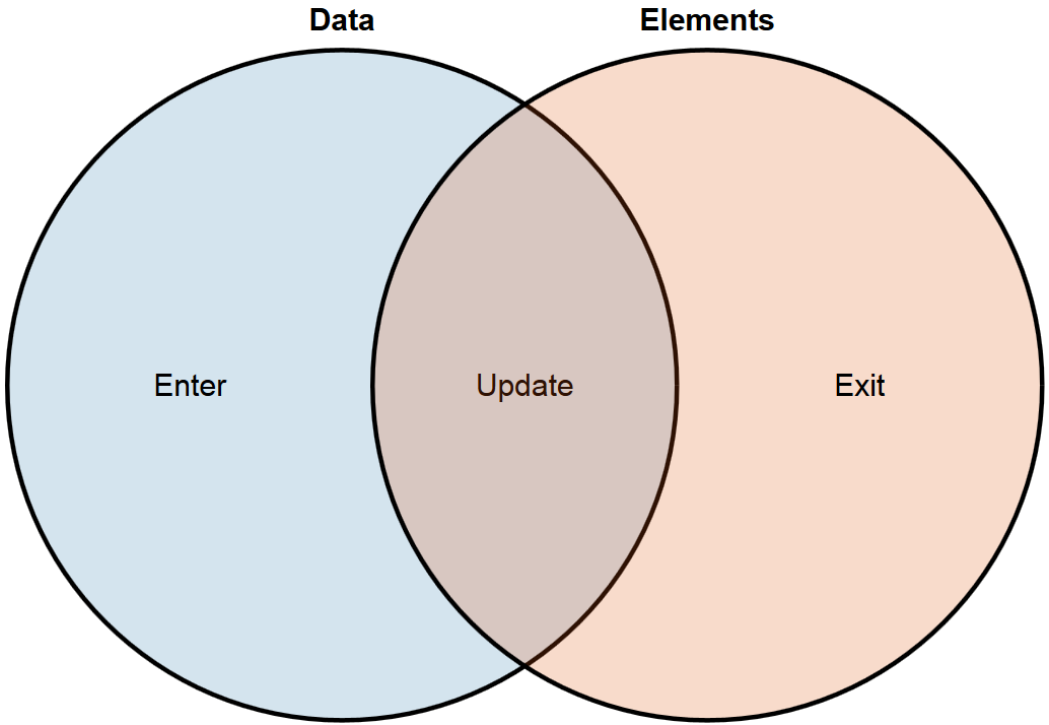
```
var svg = d3.select("#example").append('svg')
    .attr("width", 600)
    .attr("height", 600);

var data=[{"x": 100, "y": 100, "r": 20, "f":"red"},
    {"x": 20, "y": 25, "r":10, "f":"orange"},
    {"x": 400, "y": 55, "r":50, "f":"yellow"},
    {"x": 270, "y": 300, "r":40, "f":"purple"},
    {"x": 140, "y": 213, "r":12, "f":"pink"},]

svg.selectAll("circle")
    .data(data)
  .enter().append("circle")
    .attr("cx", function(d) { return d.x;})
    .attr("cy", function(d) { return d.y;})
    .attr("r", function(d) {return d.r;})
    .attr("fill", function(d) {return d.f;});
```



D3 - Enter, Update and Exit methods

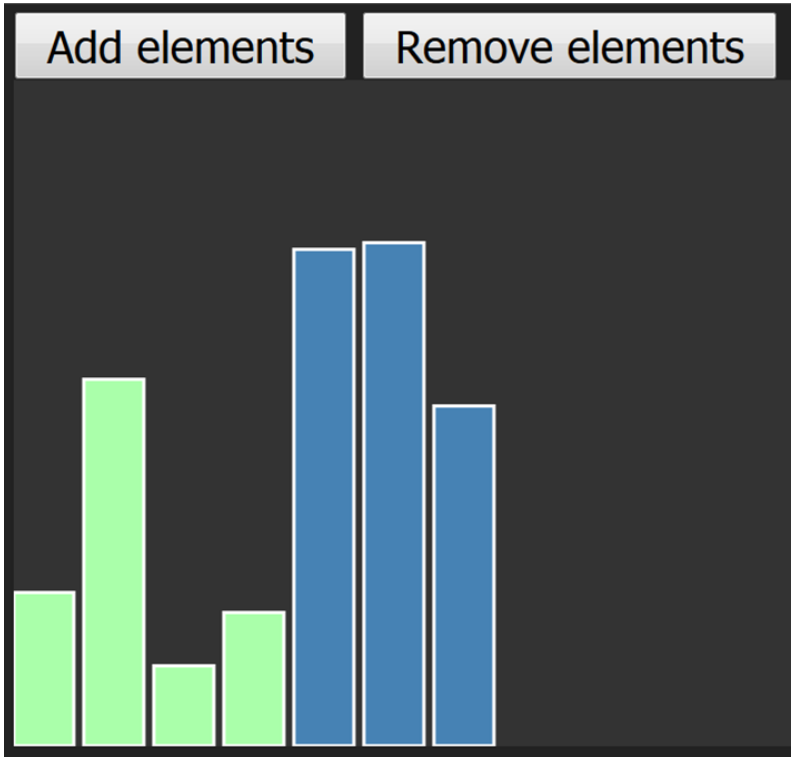


D3 - Enter, Update and Exit methods

```
<div>
    <button id="add-btn">Add elements</button>
    <button id="rm-btn">Remove elements</button>
</div>
<div>
    <svg width="800" height="200" style="background-color: #333;">
        <!-- D3 created bars -->
    </svg>
</div>
<script type="text/javascript" src="js/d3.min.js"></script>
<script type="text/javascript">
    var data = [42,64,128,32,42];
    joinDataWithDOM();

    d3.select("#add-btn").on("click", function(d) {
        for(var i=0; i<Math.floor(Math.random()*8+1); i++) {
            data.push(Math.round(Math.random()*200));
        }
        joinDataWithDOM();
    });

    d3.select("#rm-btn").on("click", function(d) {
        data.splice(0, Math.floor(Math.random()*data.length));
        joinDataWithDOM();
    });
```



D3 - Enter, Update and Exit methods

```
    function joinDataWithDOM() {
        var selection = d3.select("svg").selectAll(".bar").data(data);
        selection.attr("y", function(d) { return 200-d; })
            .attr("height", function(d) { return d; })
            .style("fill", "#AFA");

        var enterSubSelection = selection.enter();
        enterSubSelection.append("rect")
            .attr("class", "bar")
            .attr("x", function(d,i) { return i*21; })
            .attr("width", 18)
            .attr("y", function(d) { return 200-d; })
            .attr("height", function(d) { return d; })
            .on("click", function(d,i) {
                data.splice(i,1);
                joinDataWithDOM();
            });

        var exitSubSelection = selection.exit();
        exitSubSelection.remove();
    }
</script>
```

D3 - Transitions

```
<svg width="400" height="200" style="fill: #222;">
    <circle id="c1" cx="25" cy="25" r="25" style="fill: #FFF;
    stroke: #FFF;"></circle>
    <circle id="c5" cx="25" cy="175" r="25" style="fill: #BFF;
    stroke: #FFF;"></circle>
</svg>
<script type="text/javascript" src="js/d3.min.js"></script>
<script type="text/javascript">
    d3.select("svg").selectAll("circle").transition()
        .delay(150)
        .duration(10000)
        .ease("bounce")
        .attr("cx", 375)
        .attr("cy", 100)
        .style("fill", "#222");
</script>
```

D3 - Advantages and Disadvantages

Transformations & transitions	No IE8 compatibility
Fast development cycle	Steep learning curve
DOM manipulation	No own visualization methods
CSS selectors	D3 is not perfect
Creative freedom	
Great community, nice tutorials	

Our App

Performance

Cost	Entries	Functions
59.56%	20528	Graphics
34.72%	11966	Gecko
1.80%	619	Styles
0.86%	298	Input & events

- performance measured in mozilla firefox version 40.0.3

Problems we had

- Performance
  - many draw calls
  - large path data
- Transitions hard to do right
  - Adjustment of multiple interleaving transitions
- No easy way to run multiple concurrent transitions on one element
- API we used had problems with d3 (data.worldbank.org)

When to use D3?

- Prototyping getting work done fast and efficient.
- Reuseable code
- Simple and common graphs are included in the D3 layouts.

When not to use D3?

- Advanced big data operations are needed
- A real charting library is needed
- Steep learning curve

Thanks for listening!

Questions?

Sources

1. Mike Bostock: http://bost.ocks.org/mike/
2. Steps of visualizing data: https://www.dashingd3js.com/the-data-visualization-process
3. Ben Fry book Visualizing Data: Exploring and Explaining Data with the Processing Environment: http://www.amazon.com/gp/product/B0028N4WJC/ref=as_li_qf_sp_asin_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=B0028N4WJC&linkCode=as2&tag=dashi07-20
4. Craig Buckler's 7 Reasons to Consider SVG Instead of Canvas: http://www.sitepoint.com/7-reasons-to-consider-svgs-instead-of-canvas/
5. Javascript HTML DOM http://www.w3schools.com/js/js_htmldom.asp
6. Vegibit tutorial: http://vegibit.com/create-a-bar-chart-with-d3-javascript/
7. http://bost.ocks.org/mike/join/