IN4MATX 133: User Interface Software

Lecture:
Software & Visualization
Tools

Goals for this Lecture

By the end of this lecture, you should be able to…

- Explain the relative threshold and ceilings of visualization tools like Protovis, D3, and Vega-Lite
- Describe common visualization primitives like marks, axes, and scales
- Implement simple visualizations with Vega-Lite
- Explain the different roles HTML, CSS, and JavaScript play
- Describe how JavaScript standards evolved
- Follow JavaScript syntax for traditional programming concepts like typing, variable assignment, loops, and conditionals

Socrative Quiz!

Enter your UCI Email when prompted name!!!
e.g.,

xxxxx@uci.edu

https://api.socrative.com/rc/CvereT

Declarative languages

HTML

Declarative language

CSS

Declarative language

JS

Imperative language

Goals for this Lecture

By the end of this lecture, you should be able to…

- Explain the relative threshold and ceilings of visualization tools like Protovis, D3, and Vega-Lite

- Describe common visualization primitives like marks, axes, and scales

- Implement simple visualizations with Vega-Lite

- Describe the different roles JavaScript has in client-side and server-side development

- Explain the role of the Document Object Model (DOM)

Declarative languages

HTML

```
<main class="container">
  <div class="row">
    <div class="col-3">A</div>
    <div class="col-6">B</div>
    <div class=“col-4">C</div>
    <div class=“col">D</div>
    <div class="col">E</div>
  </div>
</main>
```

What should be rendered, but not how

JS

```
var array = ['1', 'fish', 2, 'blue'];
array[5] = 'dog';
array.push('2');
array[2] = array[array.length - 1] - 4;
array[0] = typeof array[2];
array[4] = array.indexOf('blue');

console.log(array.join('*'));
```

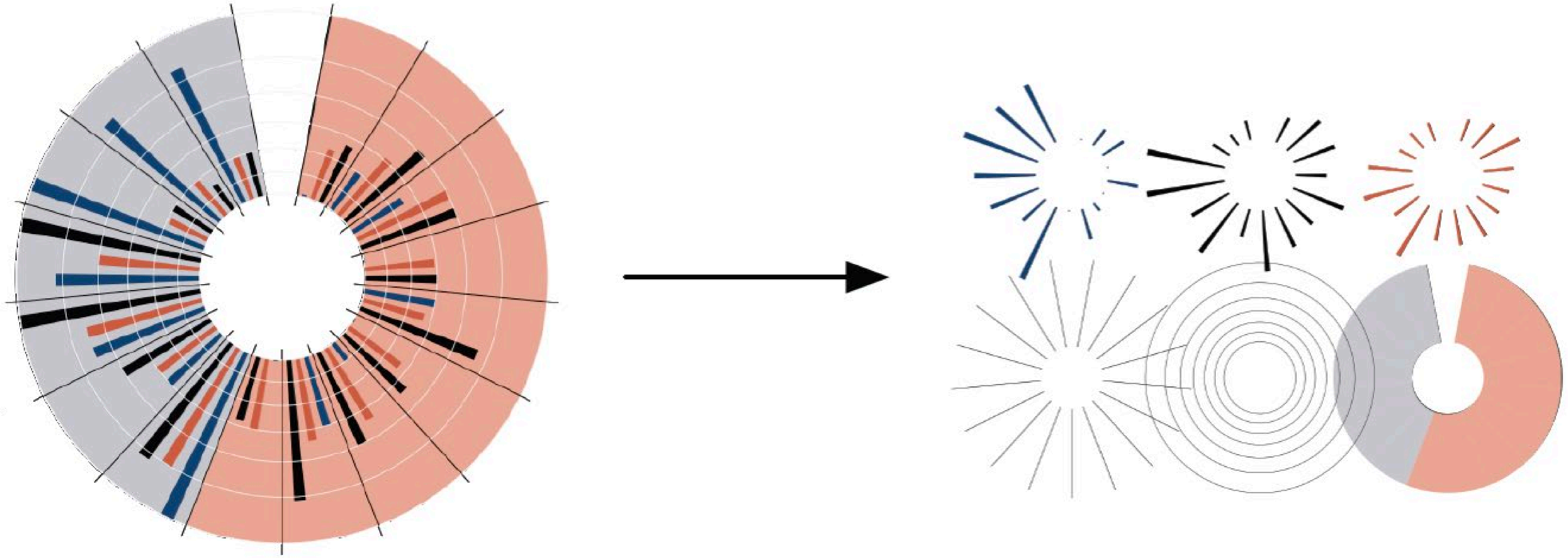
Step-by-step

Why declarative languages?

- Faster iteration, less code, lower threshold
- Can be generated programmatically
- Generally considered easier to learn than programming/imperative languages like JavaScript

Protovis

- Initial grammar for visualization
- A composition of data-representative marks
    - Self-contained JavaScript model (doesn't export to SVG or anything else)

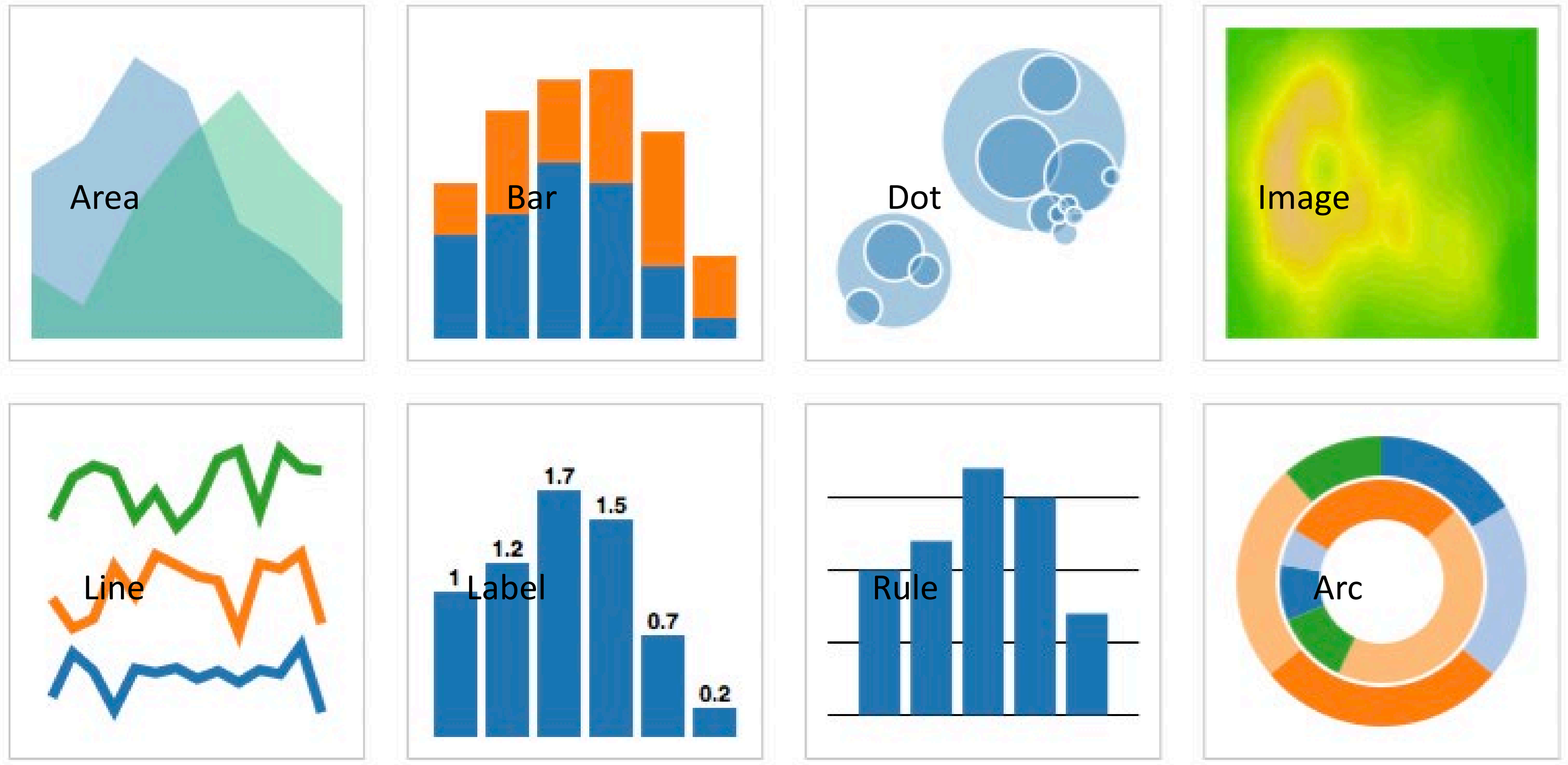


Michael Bostock, Jeffrey Heer. IEEE Vis, 2009. Protovis: A Graphical Toolkit for Visualization.

https://doi.org/10.1109/TVCG.2009.174

Protovis

- Marks: graphical primitives
  - Marks specify how content should be rendered



Michael Bostock, Jeffrey Heer. IEEE Vis, 2009. Protovis: A Graphical Toolkit for Visualization.

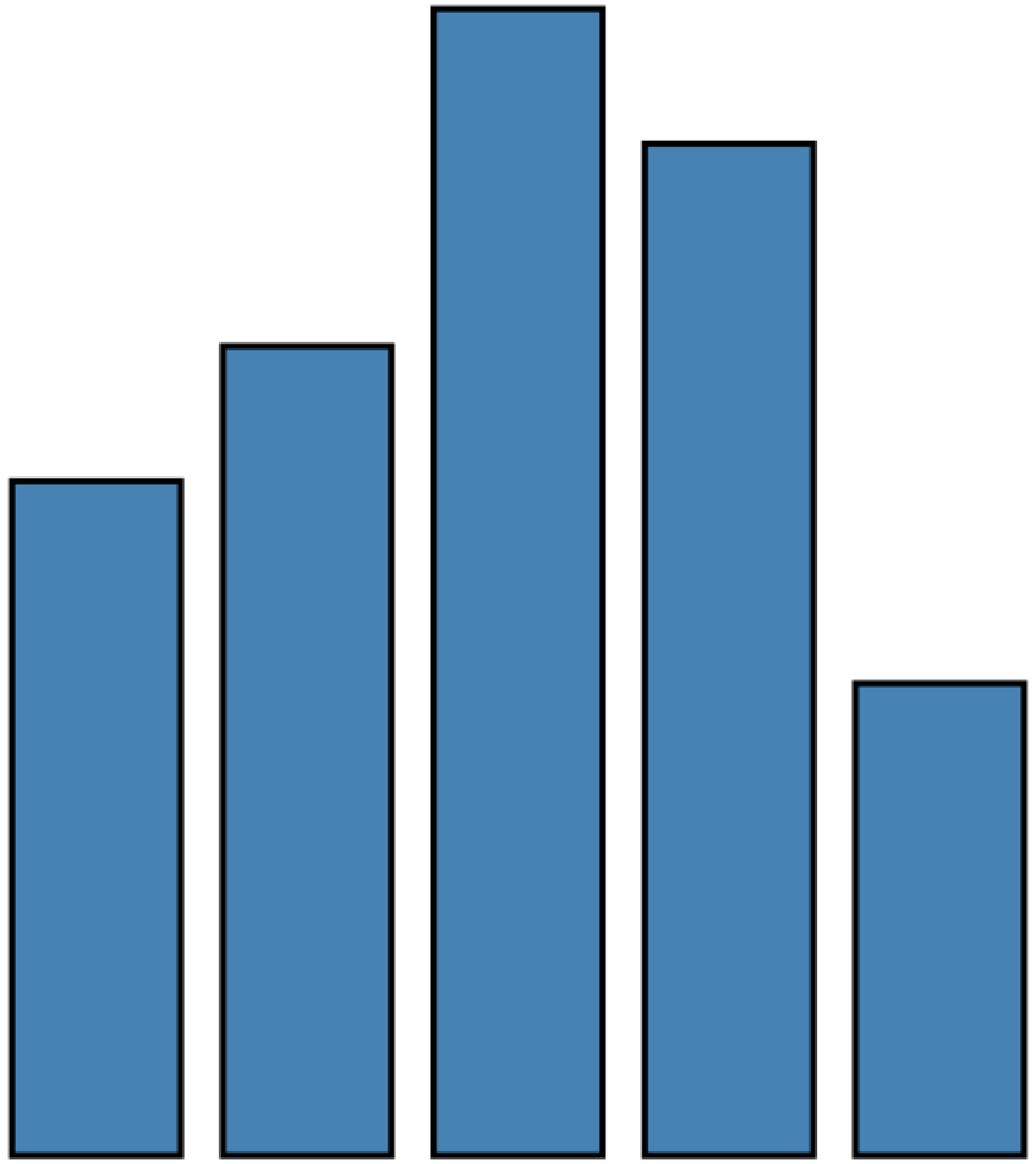
https://doi.org/10.1109/TVCG.2009.174

Protovis

```
var vis = new pv.Panel();
vis.add(pv.Bar)
.data([1, 1.2, 1.7, 1.5,
0.7])
.visible(true)
.left((d) => this.index * 25)
.bottom(0)
.width(20)
.height((d) => d * 80)
.fillStyle("blue")
.strokeStyle("black")
.lineWidth(1.5);
vis.render();
```

Mark

Literally specifies which pixel each bar should start at and how many pixels tall it should be

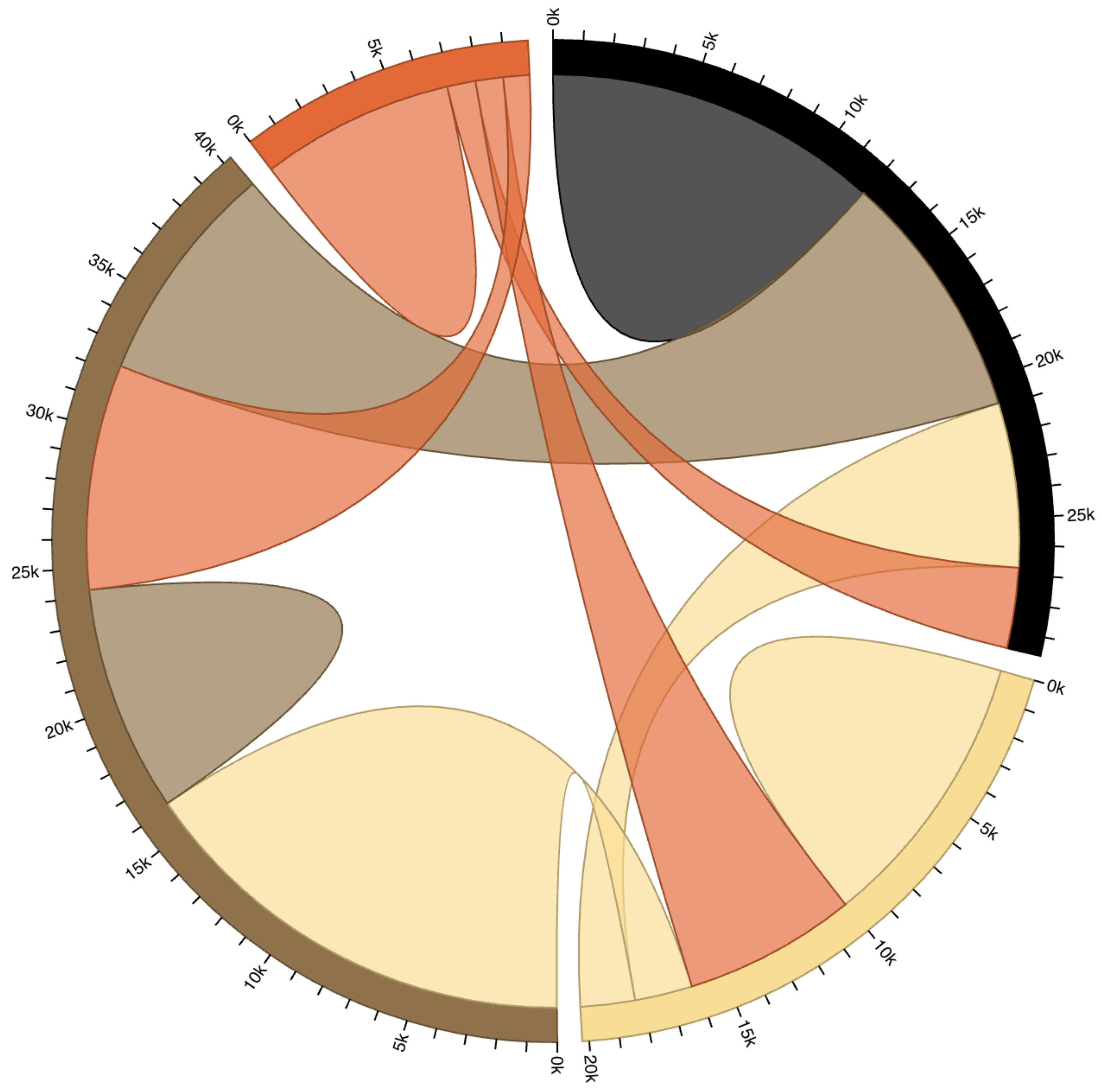


Michael Bostock, Jeffrey Heer. IEEE Vis, 2009. Protovis: A Graphical Toolkit for Visualization.

https://doi.org/10.1109/TVCG.2009.174

D3

- Binds data directly to a web page's DOM by editing a SVG
  - More expressive! Can make anything an SVG can make
  - Enables interactivity, can access mouse & keyboard events through the same tools as a browser
  - Much more complex…



Michael Bostock, Vadim Ogievetsky, Jeffrey Heer. IEEE Vis, 2011. D3: Data Driven Documents.

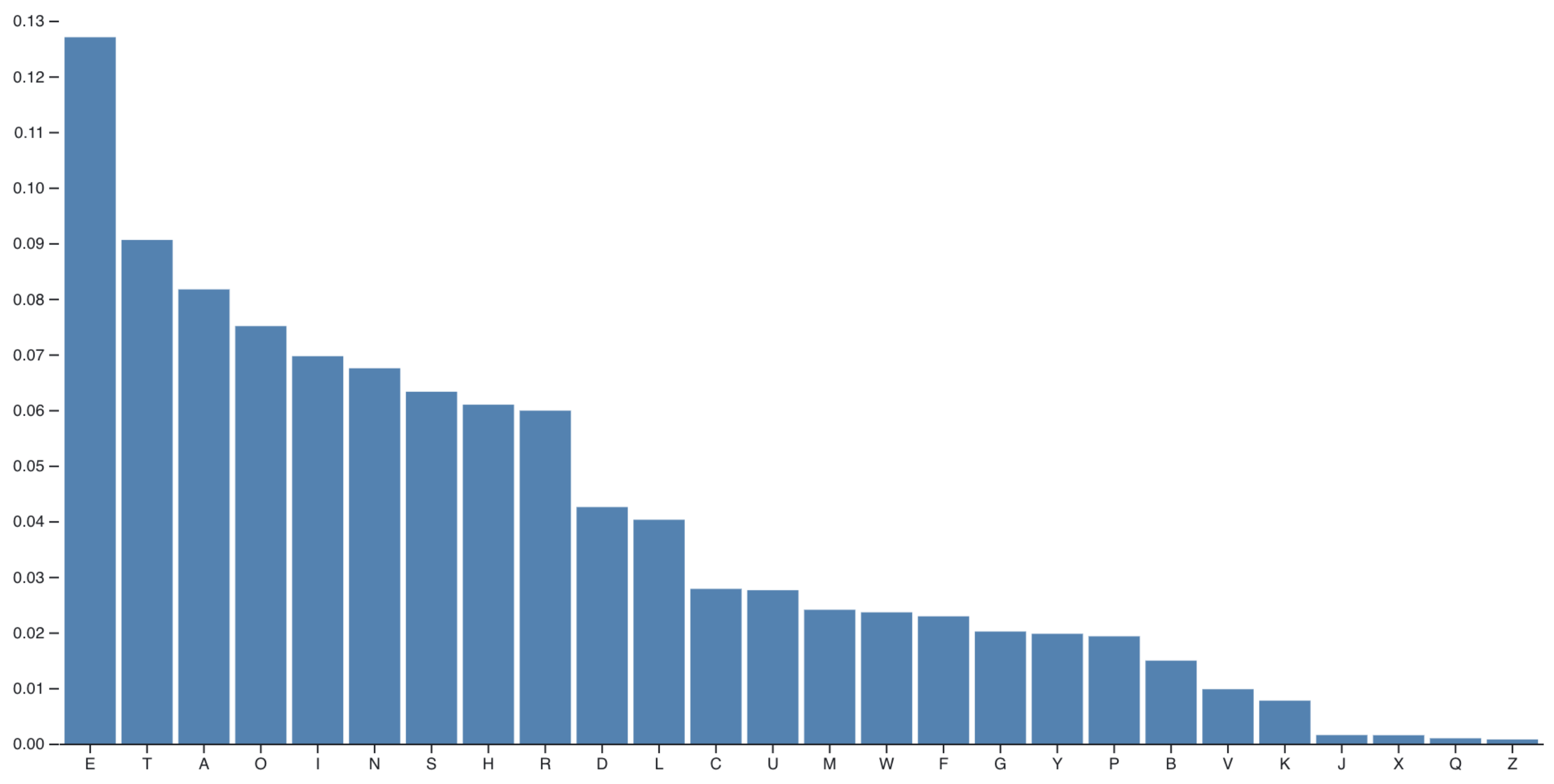
http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.185

D3

```
var svg = d3.select(DOM.svg(width,
height));

svg.append("g")
   .attr("fill", "steelblue")
.selectAll("rect").data(data).enter()
.append("rect")
   .attr("x", d => x(d.name))
   .attr("y", d => y(d.value))
   .attr("height", d => y(0) -
y(d.value))
   .attr("width", x.bandwidth());

svg.append("g")
   .call(xAxis);

svg.append("g")
   .call(yAxis);
```

Find SVG in the DOM

No more mention of marks!



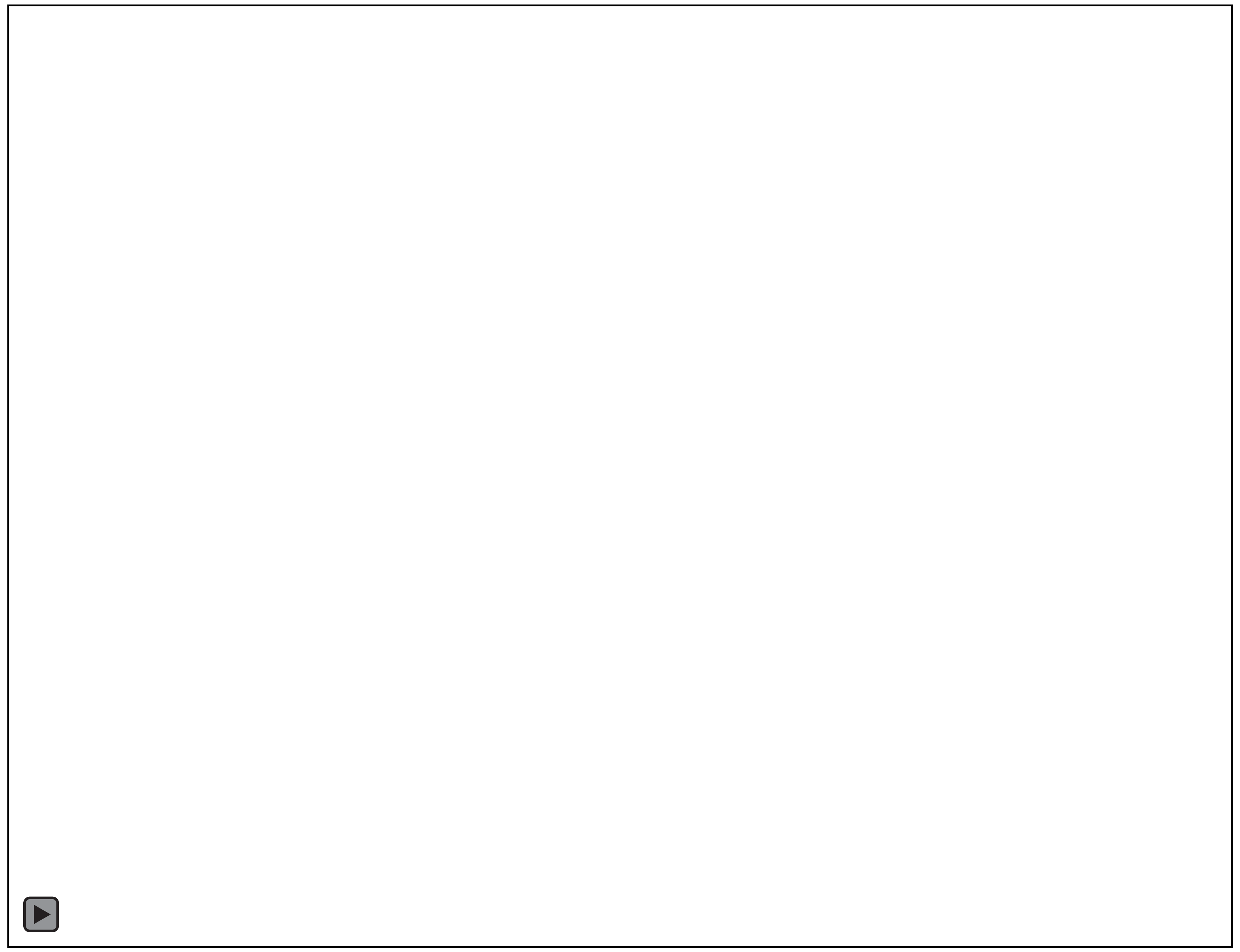
Michael Bostock, Vadim Ogievetsky, Jeffrey Heer. IEEE Vis, 2011. D3: Data Driven Documents.

http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.185

D3

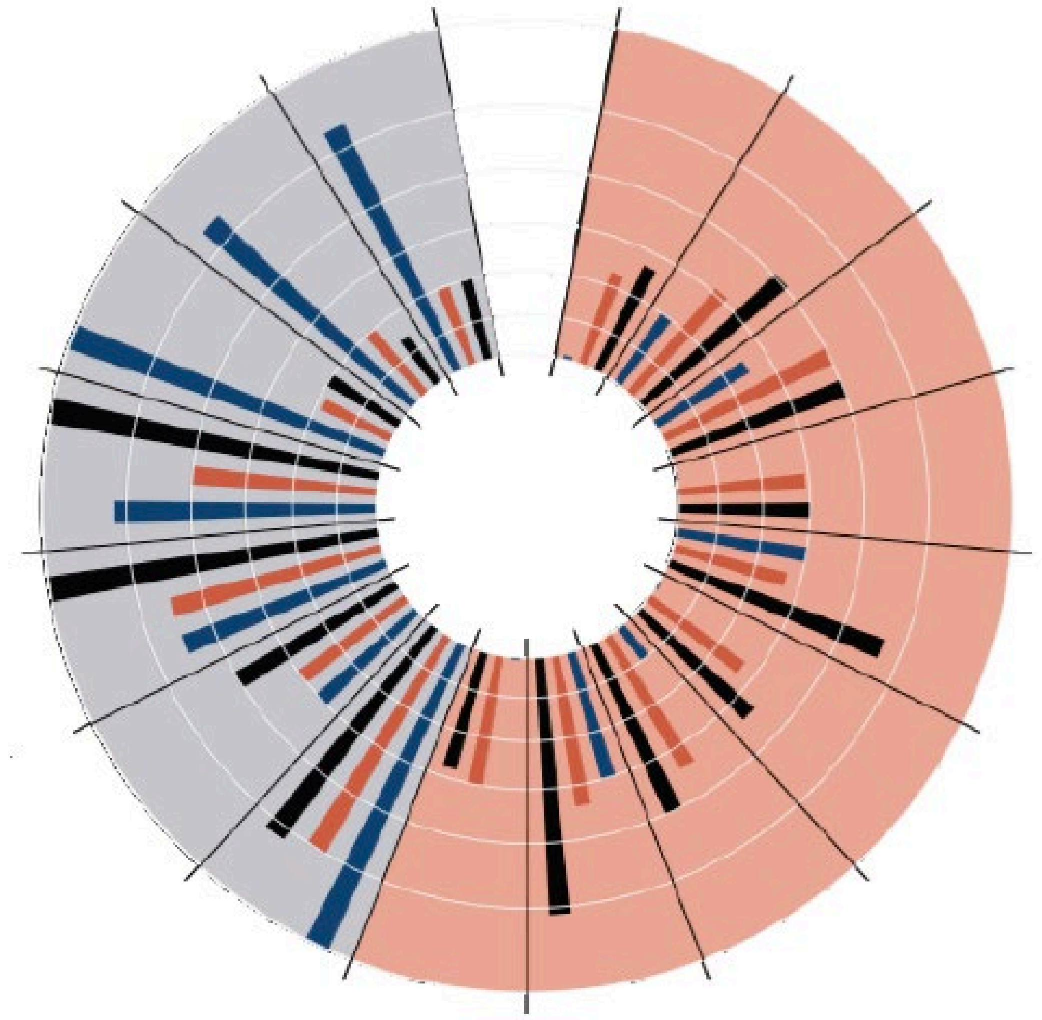
```
var svg = d3.select("body").append("svg")
      .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
      .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

d3.tsv("VotingInformation.tsv", function(error, data){

      // filter year
      var data = data.filter(function(d){return d.Year == '2012';});
      // Get every column value
      var elements = Object.keys(data[0])
            .filter(function(d){
                  return ((d != "Year") & (d != "State"));
            });
      var selection = elements[0];

      var y = d3.scale.linear()
                  .domain([0, d3.max(data, function(d){
                        return +d[selection];
                  })])
                  .range([height, 0]);

      var x = d3.scale.ordinal()
                  .domain(data.map(function(d){ return d.State;}))
                  .rangeBands([0, width]);


      var xAxis = d3.svg.axis()
            .scale(x)
          .orient("bottom");

      var yAxis = d3.svg.axis()
            .scale(y)
          .orient("left");

      svg.append("g")
      .attr("class", "x axis")
      .attr("transform", "translate(0," + height + ")")
      .call(xAxis)
      .selectAll("text")
      .style("font-size", "8px")
      .style("text-anchor", "end")
      .attr("dx", "-.8em")
      .attr("dy", "-.55em")
      .attr("transform", "rotate(-90)" );


      svg.append("g")
      .attr("class", "y axis")
      .call(yAxis);

      svg.selectAll("rectangle")
            .data(data)
            .enter()
            .append("rect")
            .attr("class","rectangle")
            .attr("width", width/data.length)
            .attr("height", function(d){
                  return height - y(+d[selection]);
            })
            .attr("x", function(d, i){
                  return (width / data.length) * i ;
            })
            .attr("y", function(d){
                  return y(+d[selection]);
            })
            .append("title")
            .text(function(d){
                  return d.State + " : " + d[selection];
            });

      var selector = d3.select("#drop")
      .append("select")
      .attr("id","dropdown")
      .on("change", function(d){
            selection = document.getElementById("dropdown");

            y.domain([0, d3.max(data, function(d){
                        return +d[selection.value];})]);

            yAxis.scale(y);

            d3.selectAll(".rectangle")
                  .transition()
                  .attr("height", function(d){
                              return height - y(+d[selection.value]);
                        })
                        .attr("x", function(d, i){
                              return (width / data.length) * i ;
                        })
                        .attr("y", function(d){
                              return y(+d[selection.value]);
                        })
                  .ease("linear")
                  .select("title")
                  .text(function(d){
                        return d.State + " : " + d[selection.value];
                  });

            d3.selectAll("g.y.axis")
                  .transition()
                  .call(yAxis);

         });

    selector.selectAll("option")
      .data(elements)
      .enter().append("option")
      .attr("value", function(d){
        return d;
      })
      .text(function(d){
        return d;
      })


});
```

~118 lines of code,
plus data in a separate file

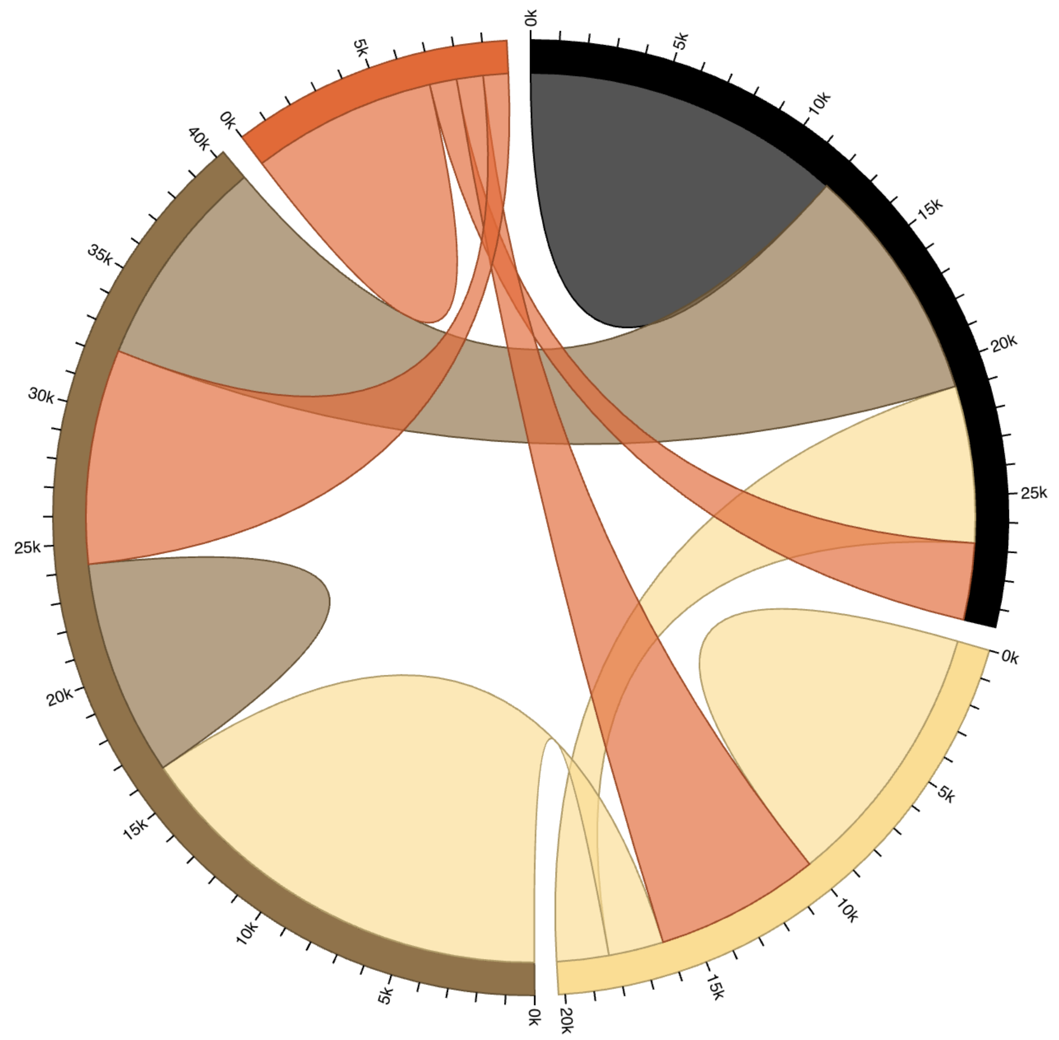


Michael Bostock, Vadim Ogievetsky, Jeffrey Heer. IEEE Vis, 2011. D3: Data Driven Documents.

http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.185



D3



Protovis
Low(er) ceiling

D3
High(er) ceiling

Compared to excel, etc., both have a high ceiling

But both have a pretty high threshold!

Vega-Lite: lowering the threshold

Lowering the threshold

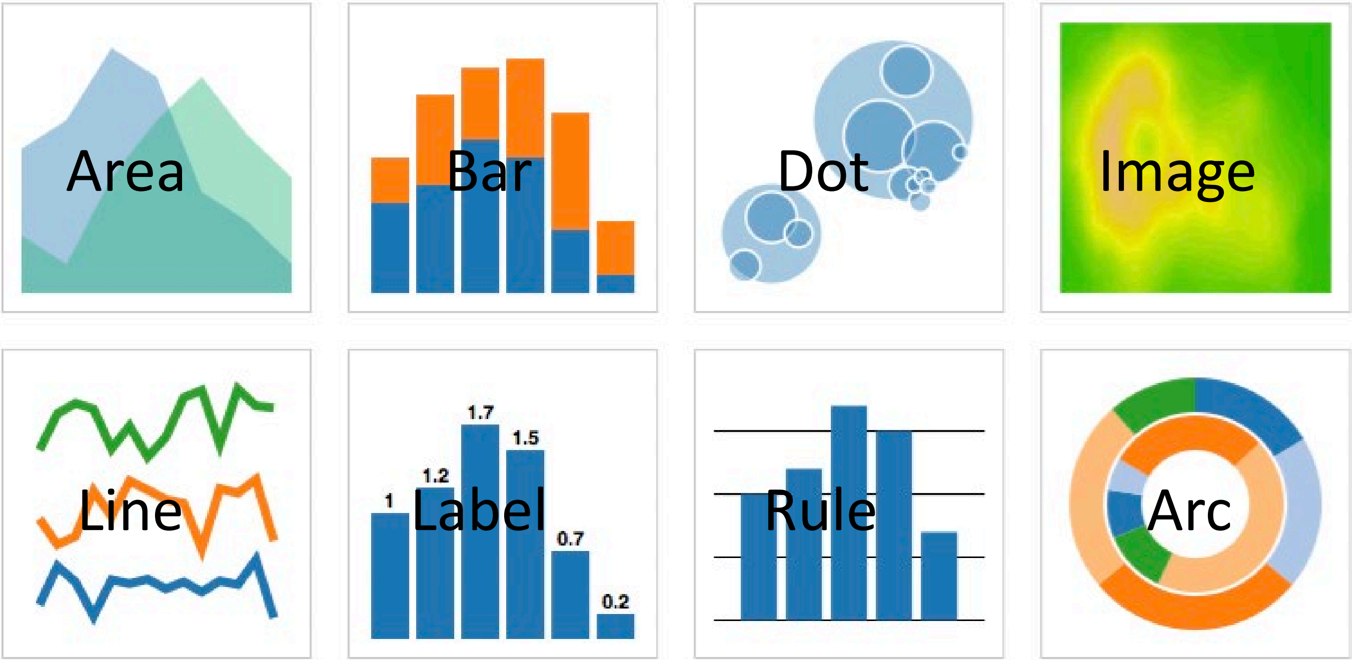
- Goal: “create an *expressive* (high ceiling) yet *concise* (low threshold) declarative language for specifying visualizations”

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

Vega-Lite

- Grammar of graphics
  - Data: input data to visualize
  - Mark: Data-representative graphics
  - Transform: whether to filter, aggregate, bin, etc.
  - Encoding: mapping between data and mark properties
  - Scale: map between data values and visual values
  - Guides: axes & legends that visualize scales

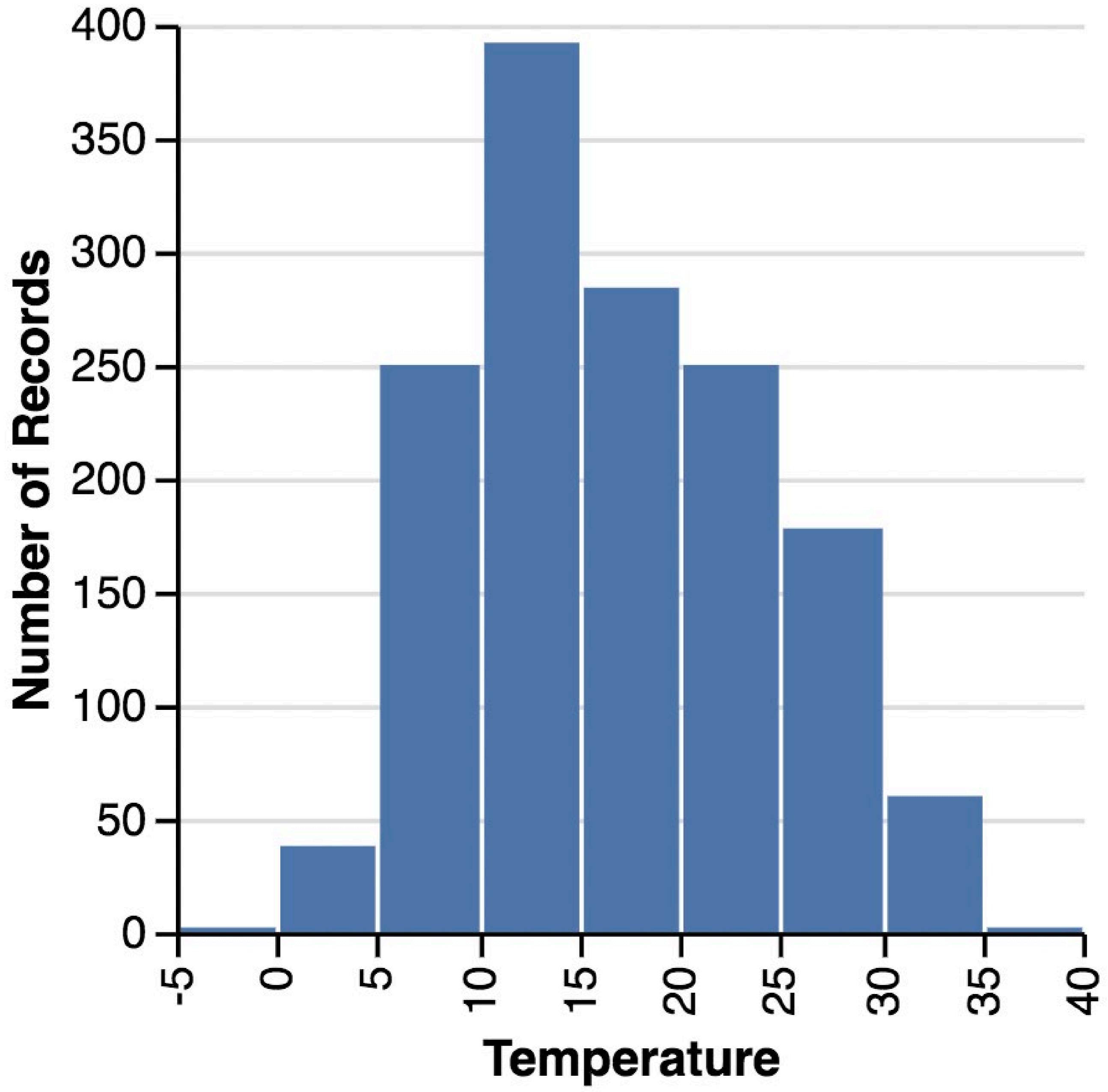


Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Making a histogram



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

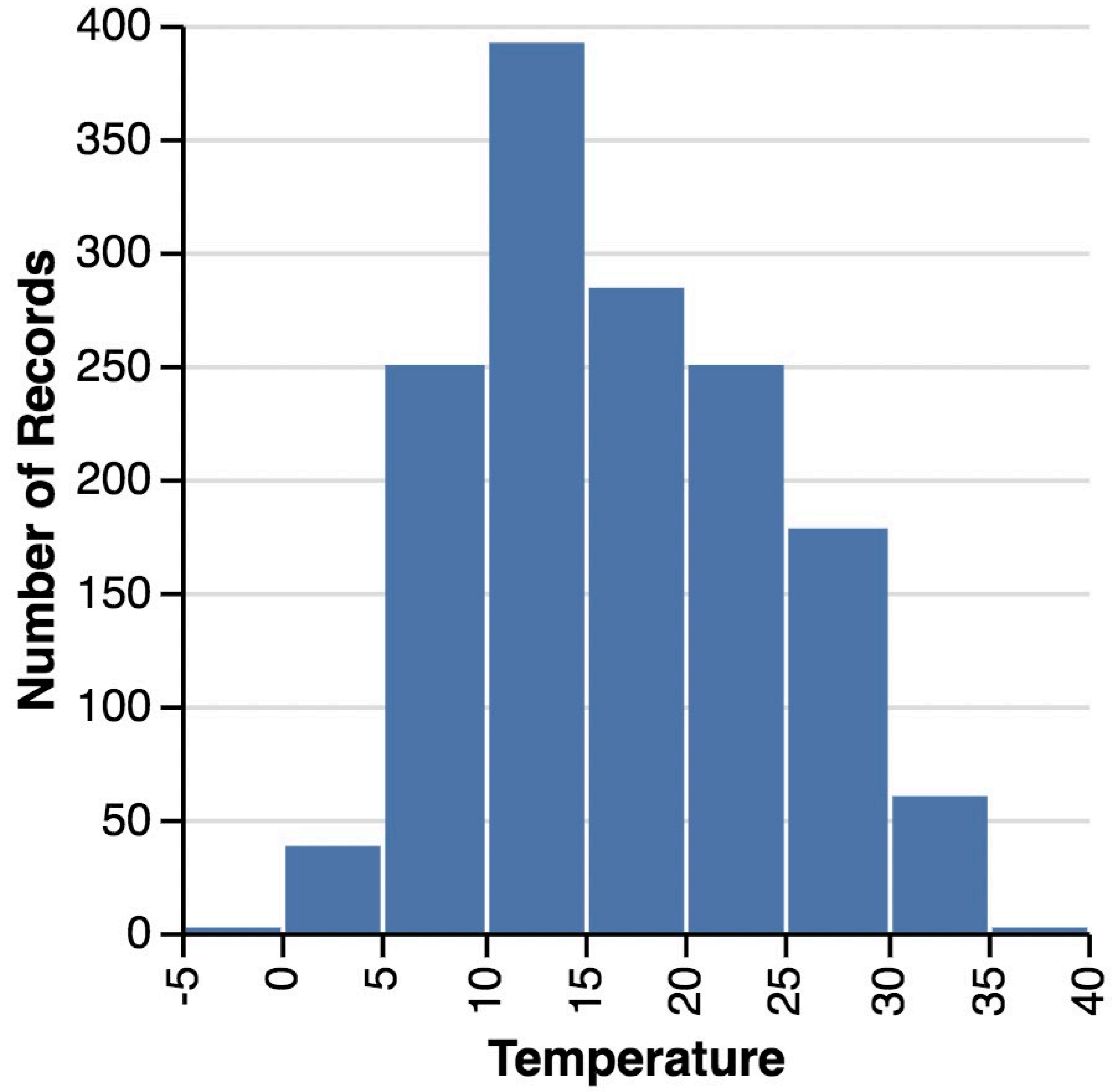
JSON file

```
[
    {
        "date": "2015/01/01",
        "weather": "sun",
        "temperature": 1.1999999999999997
    },
    {
        "date": "2015/01/02",
        "weather": "fog",
        "temperature": 2.8
    },
    {
        "date": "2015/01/03",
        "weather": "fog",
        "temperature": 3.35
    },
    {
        "date": "2015/01/04",
        "weather": "fog",
        "temperature": 6.949999999999999
    },
    {
        "date": "2015/01/05",
        "weather": "fog",
        "temperature": 10.8
    },
...
```

Vega-lite

Histogram = (Bar with x=binned field, y=count)

- Bin records by their temperature
- Count how many records fall into each bin
- Render those bins as vertical bars



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

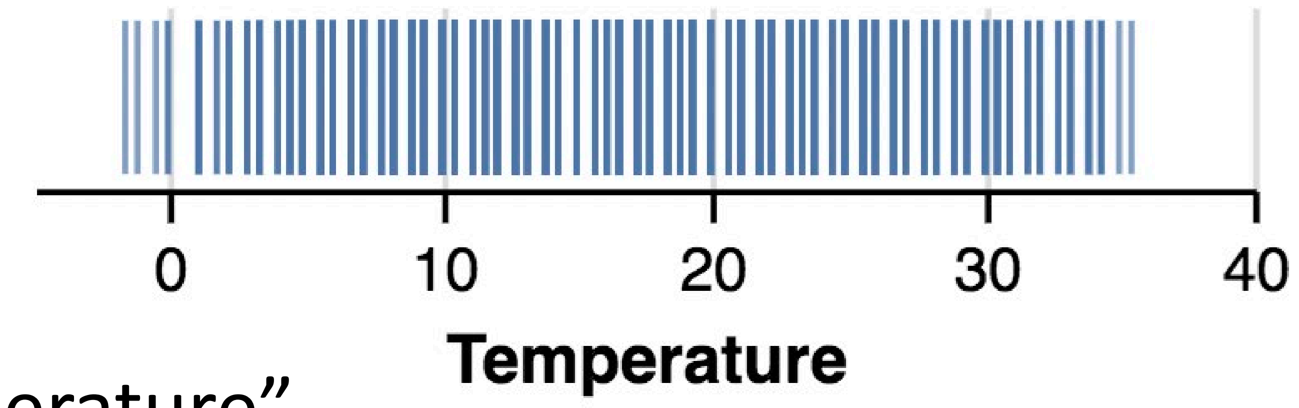
Vega-lite

Histogram = (Bar with x=binned field, y=count)

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      field: "temperature",
      type: "quantitative"
    }
}
```

Set mark as a tick

Encode x according to the “temperature” field



Four types:
quantitative (numerical)
temporal (time)
ordinal (ordered)
nominal (categorical)

Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

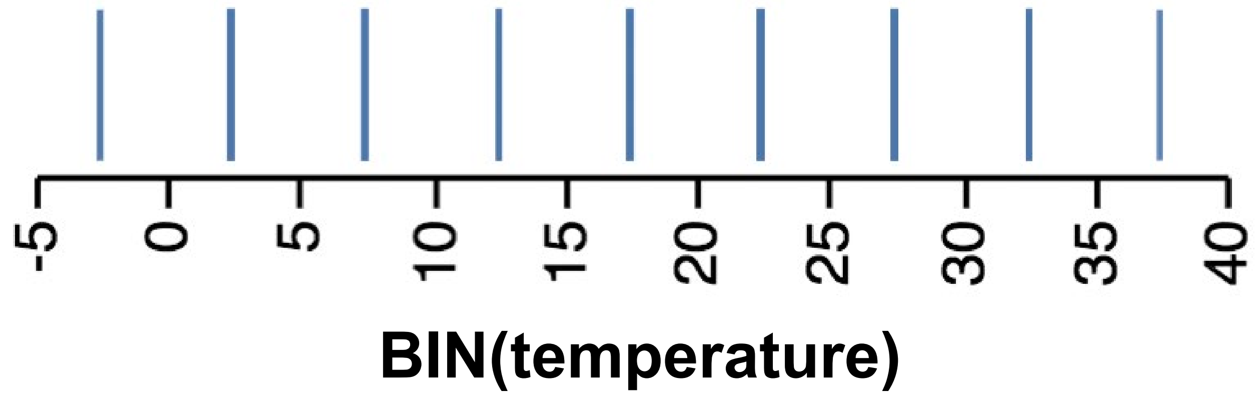
https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Histogram = (Bar with x=binned field, y=count)

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    }
}
```

Bin values by x dimension



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

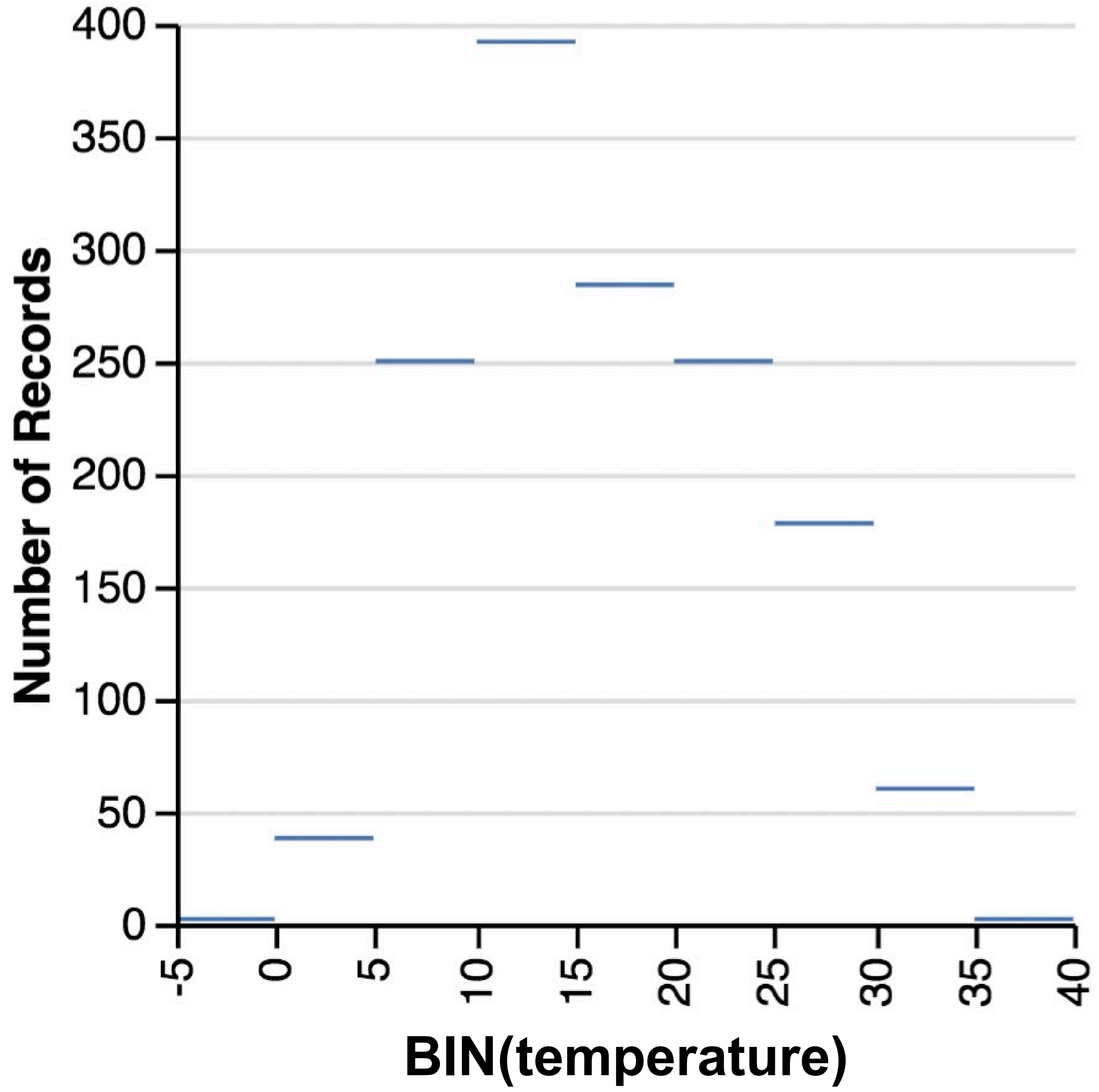
https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Histogram = (Bar with x=binned field, y=count)

```
{
  data: {url: "weather-seattle.json"},
  mark: "tick",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
}
```

y should aggregate the bins by counting how many values are in them



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

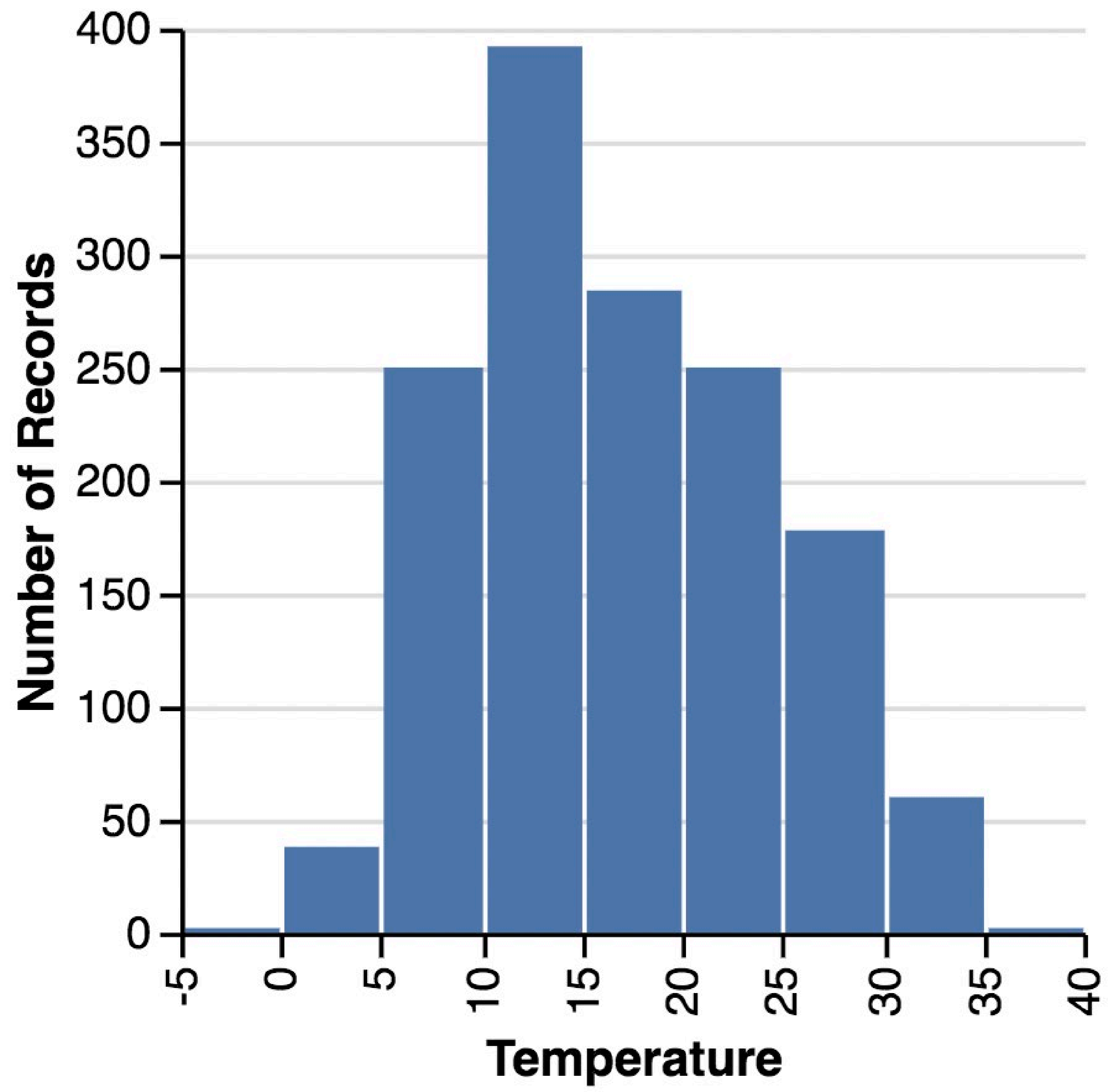
https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Histogram = (Bar with x=binned field, y=count)

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    }
}
```

← Change the mark to a bar



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

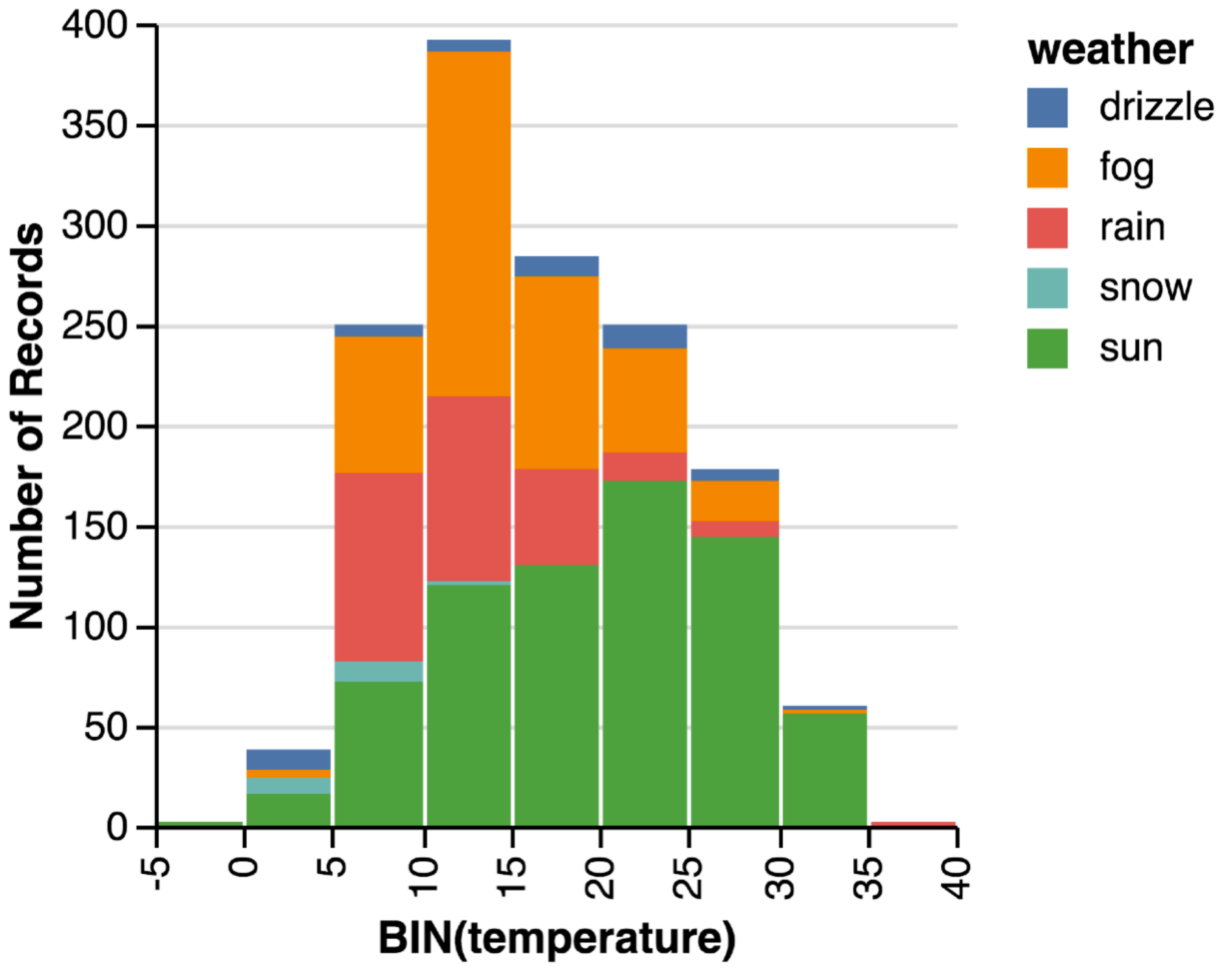
https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Histogram + Color

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
     field: "weather",
     type: "nominal"
    }
}
```

Set the color to follow the weather field



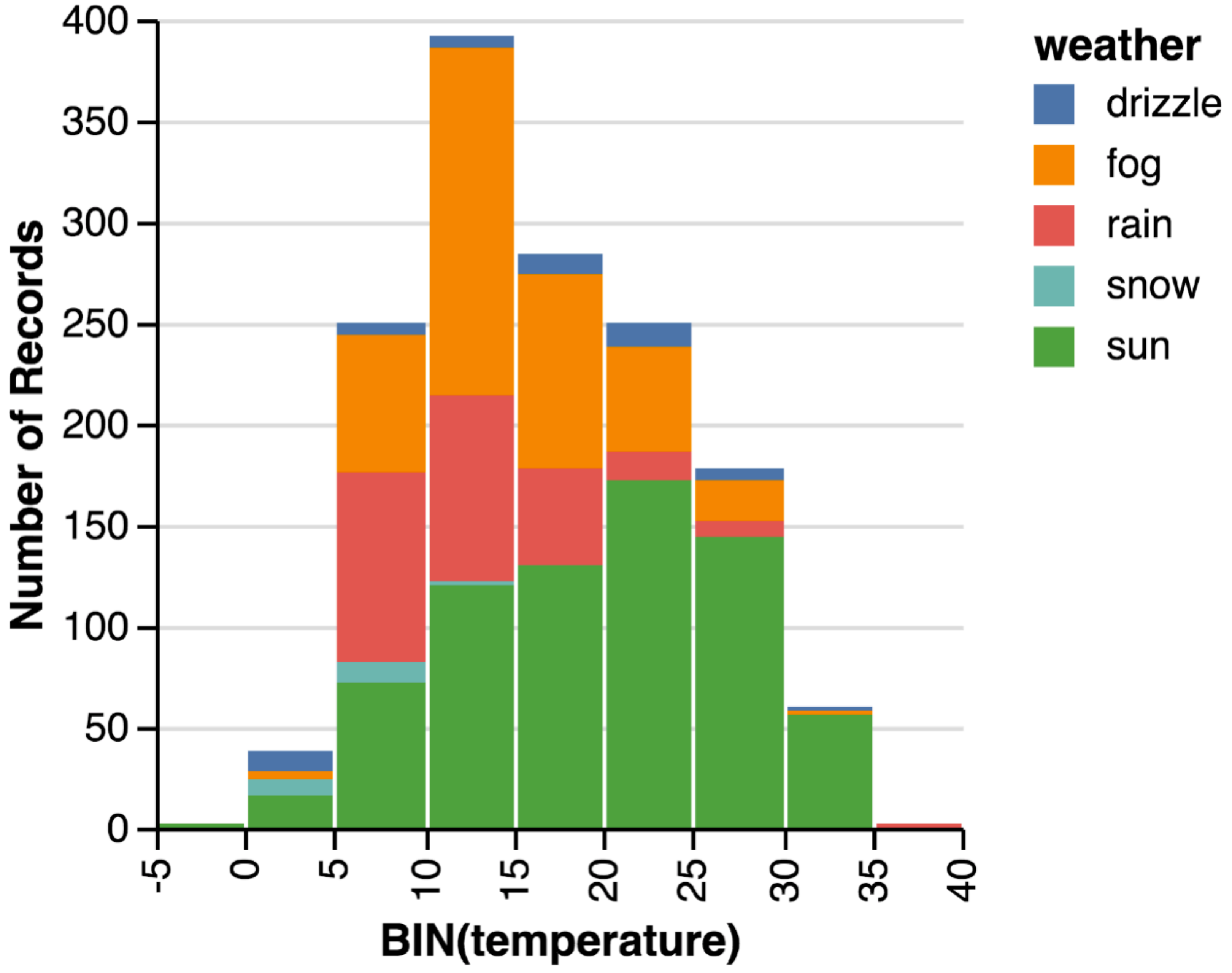
Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

“Sensible defaults”

- The field chose reasonable defaults for presenting the data
  - We didn’t specify what colors to use
  - Or how wide bins should be
  - Or how to label the axes
  - Or that the bars should be stacked
  - …



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

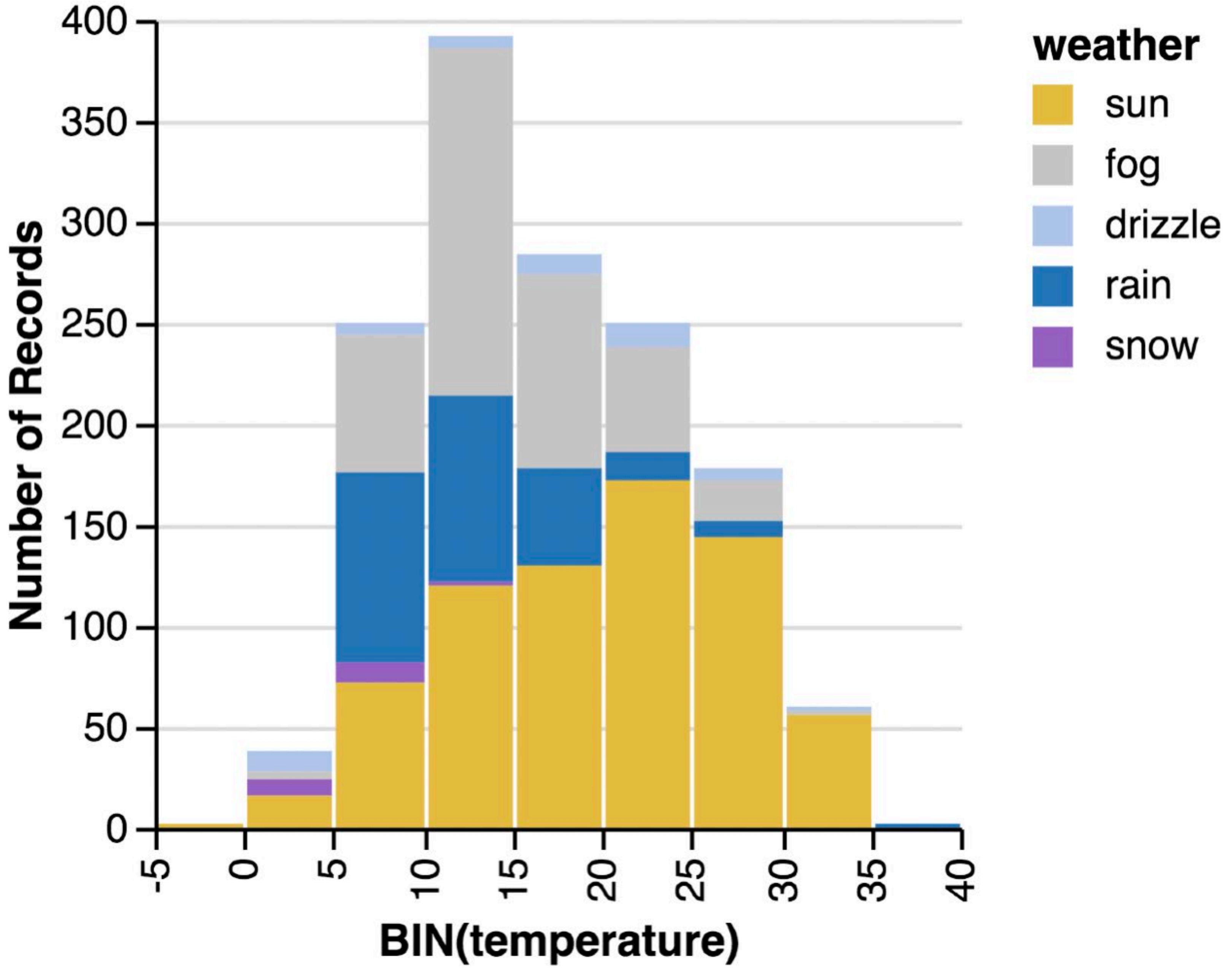
https://doi.org/10.1109/TVCG.2016.2599030

Vega-lite

Overriding the sensible defaults

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      bin: true,
      field: "temperature",
      type: "quantitative"
    },
    y: {
      aggregate: "count",
      type: "quantitative"
    },
    color: {
     field: "weather",
     type: "nominal"
    },
    scale: {
        domain: ["sun","fog","drizzle", "rain","snow"],
        range: ["#e7ba52","#c7c7c7","#aec7e8",
               “#1f77b4","#9467bd"]
  }
}
```

Set our own color scale



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

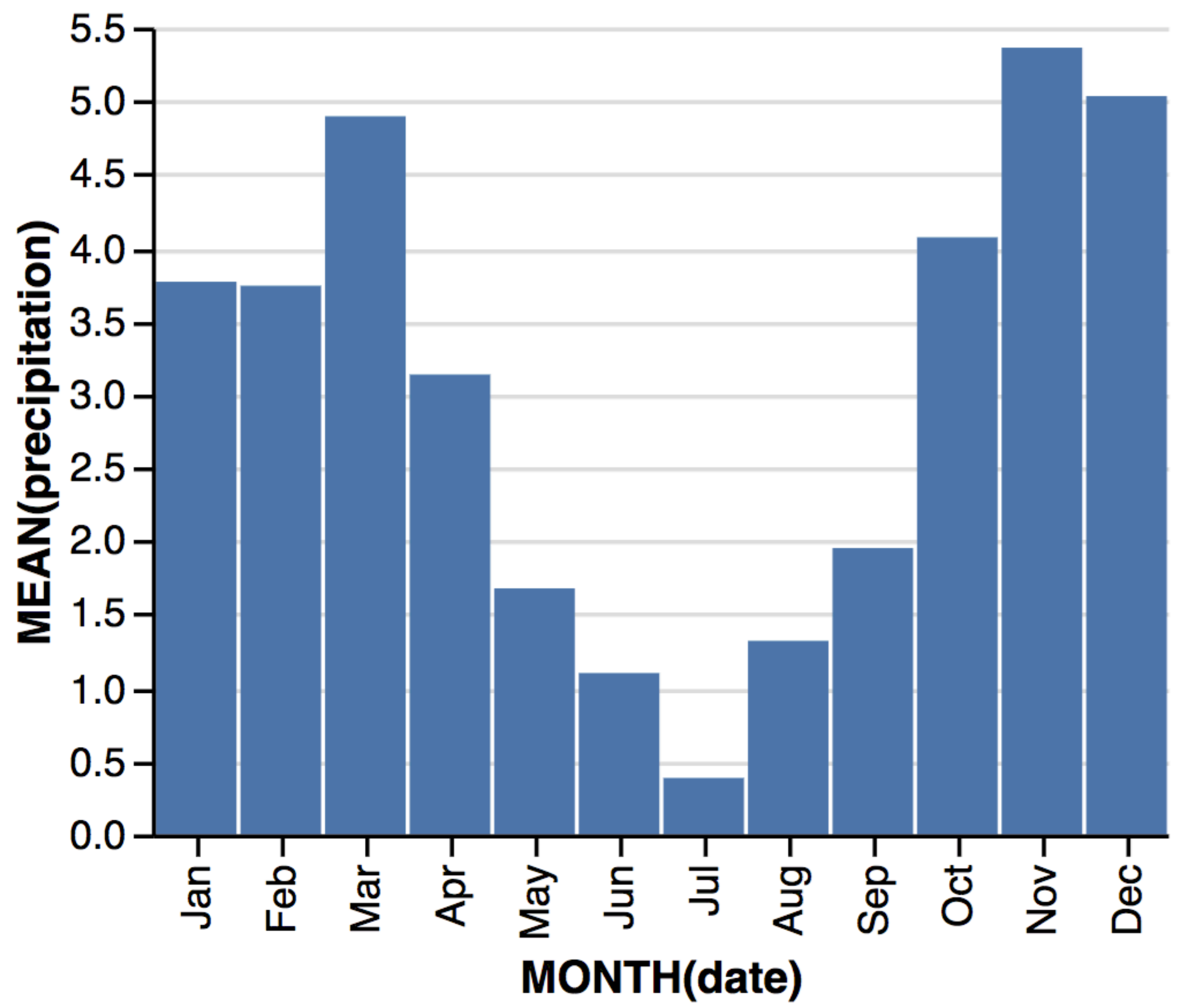
Vega-lite

Monthly precipitation

```
{
  data: {url: "weather-seattle.json"},
  mark: "bar",
  encoding: {
    x: {
      timeUnit: "month", field: "date",
      type: "quantitative"
    },
    y: {
      aggregate: "mean",
      field: "precipitation",
      type: "quantitative"
    }
  }
}
```

Field is a date string (“2018-10-17”), but display and bin it as a month

Aggregate by the mean



Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer. IEEE Vis, 2017. Vega-Lite: A Grammar of Interactive Graphics

https://doi.org/10.1109/TVCG.2016.2599030

Sensible defaults: Vega-lite’s secret

- Threshold is lower
  - More concise definitions, less to understand up front
- Ceiling remains the same
  - The sensible defaults can be overridden
- A downside: visualizations made with Vega-Lite look similar
  - The path of least resistance Vega-Lite provides influences what visualizations people make and what they look like

Downside: path of least resistance

- The path of least resistance: tools influence what is created
- Sensible defaults make Vega-Lite visualizations look similar to one another
  - These defaults *can* be overwritten, but are they in practice?
- Similar concern to the widespread adoption of grid frameworks

For A2, you will be using Vega-Lite alongside JavaScript to create interactivity into a web page.

Language Roles

HTML

Markup language

CSS

Styling language

JS

Programming language

Language Roles

Specify how content is rendered

Visually style content

Dynamically manipulate content

Why JavaScript?

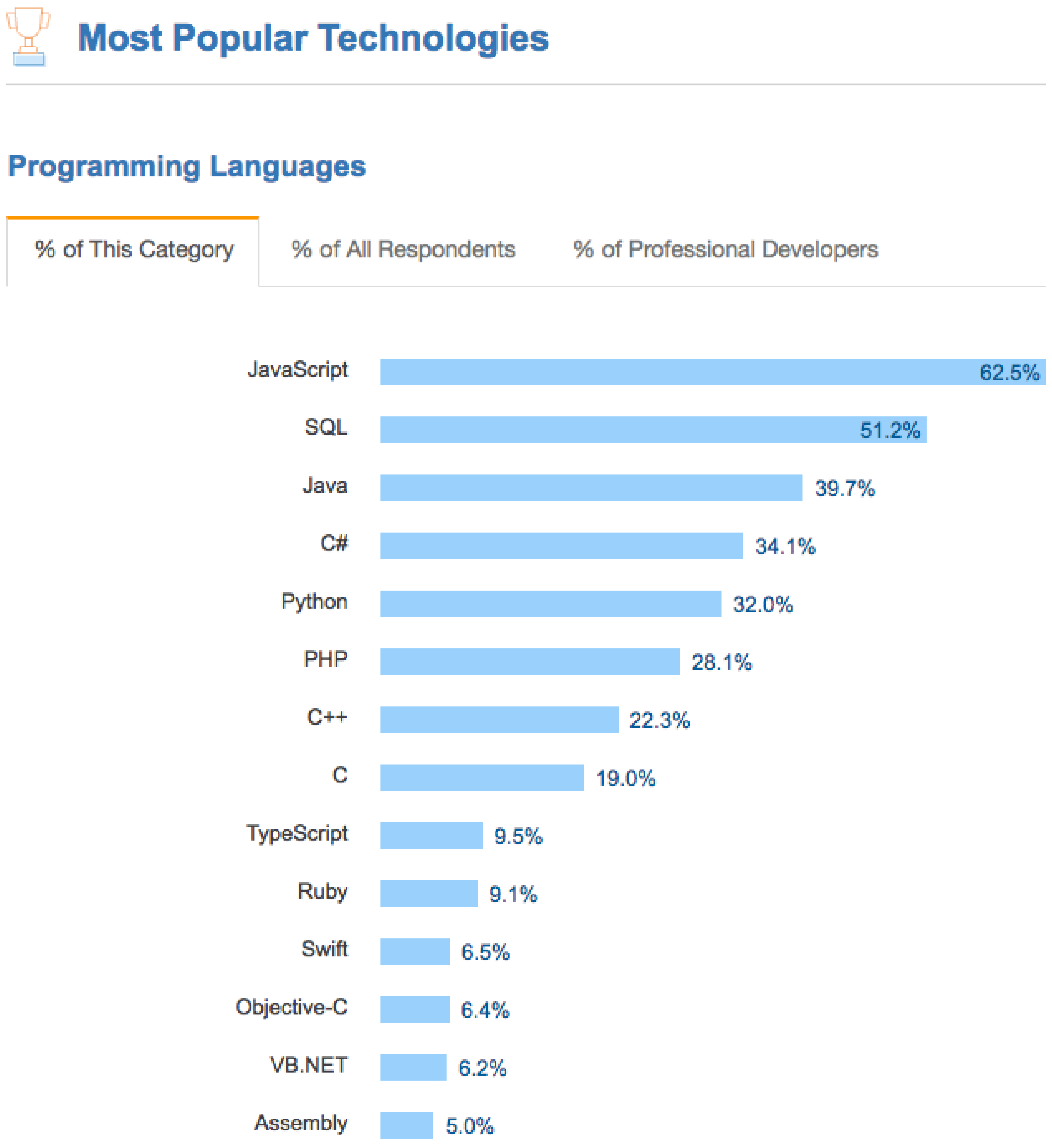
- Make pages dynamic
- Make pages personalized
- Make pages interact with other sources, like databases and APIs

Other web programming languages

- Ruby, via Ruby on Rails
- Python, via Django or web2py
- These days, you can create a dynamic website in almost any language

Other web programming languages

- Some languages transpile to JavaScript
- TypeScript, by Microsoft, introduces types
  - More on TypeScript later
- Kotlin, by Google, runs on the Java virtual machine and compiles to JavaScript
  - Links all of Google's platforms



rity

https://insights.stackoverflow.com/survey/2017#technology-programming-languages

How did JavaScript become the most popular language for web development?

History of JavaScript

- *“Developed under the name Mocha, the language was officially called **LiveScript** when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it later was renamed JavaScript”*
- Java’s popularity was on the rise
  - Marketing ploy
  - Intended to be the “web” language to Java’s “desktop”

https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17

History of JavaScript

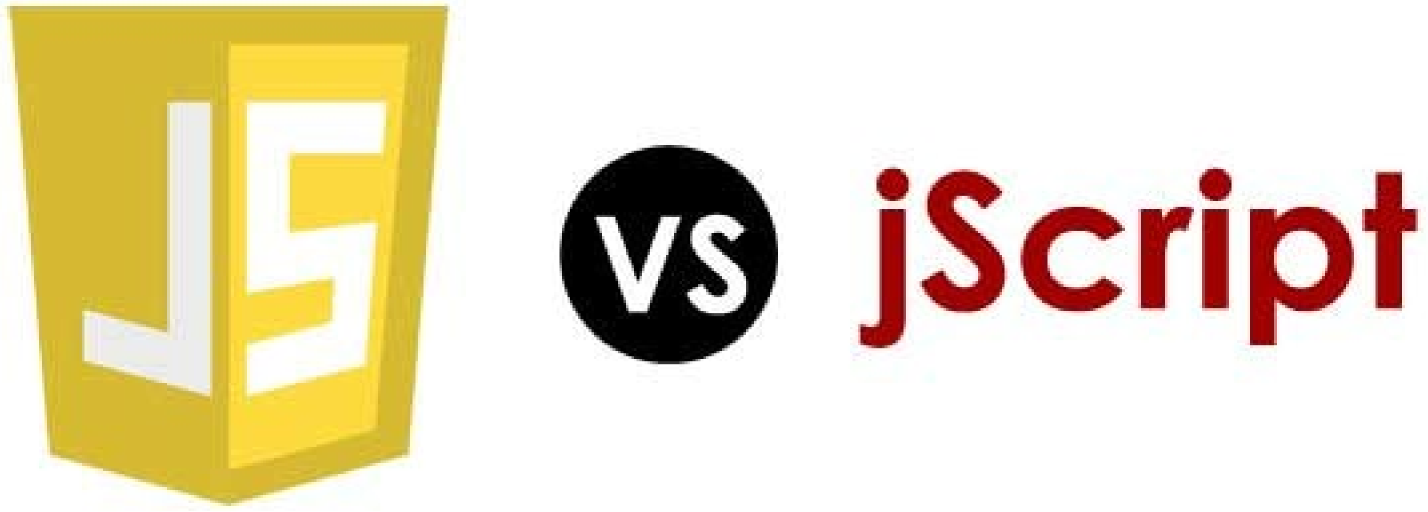
- Netscape submitted JavaScript to ECMA International for consideration as an industry standard
- Subsequent versions were standardized as “ECMAScript”
- Today, ECMAScript and JavaScript are more or less two different names for the same language

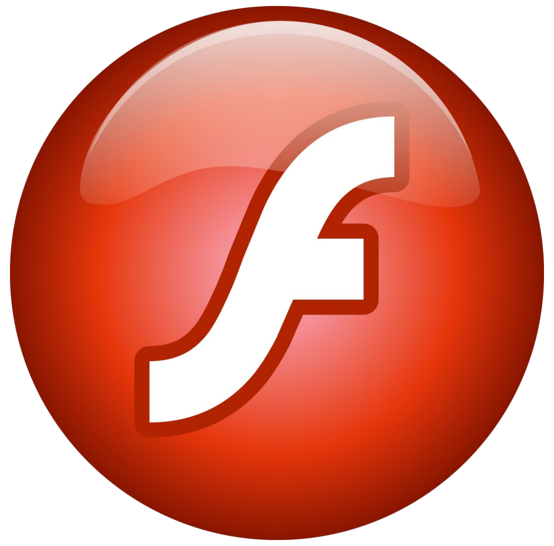
ecma
INTERNATIONAL

European Computer Manufacturers Association

History of JavaScript

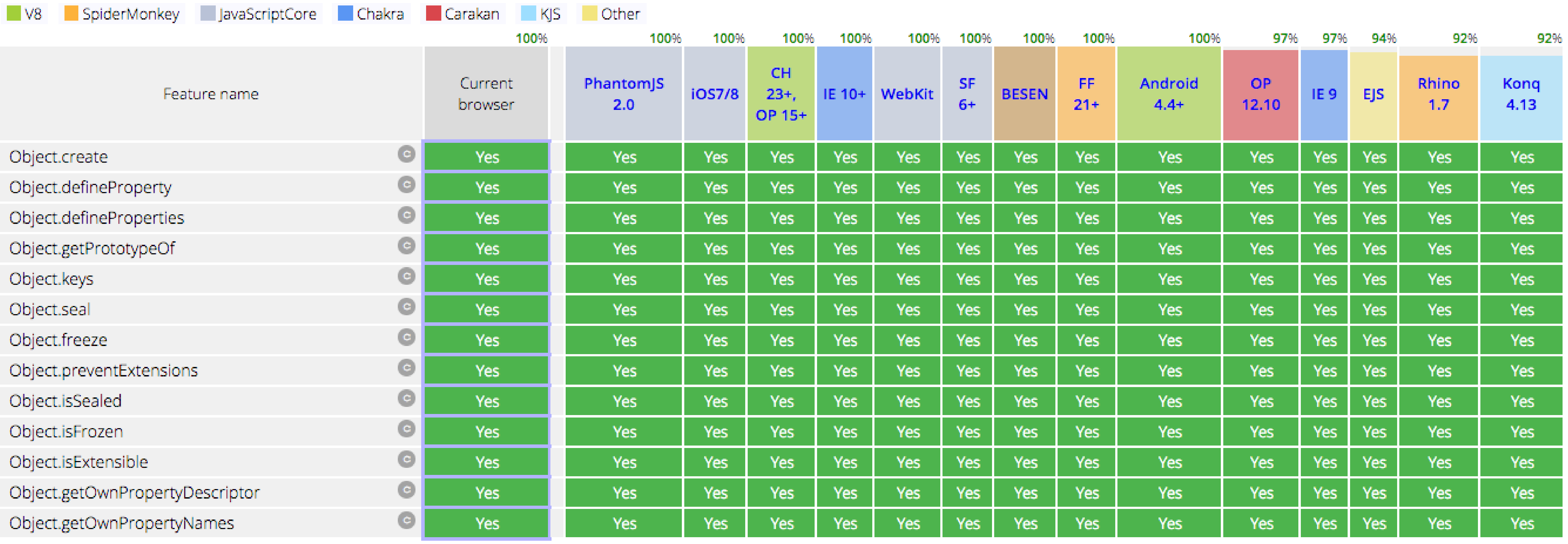
- Alternatives started springing up in the late 1990s and early 2000's
  - Microsoft introduced JScript engine
  - Macromedia Flash's ActionScript
- Both were vaguely JavaScript-like, but standards differed





History of JavaScript

- Standards later converged
  - Firefox came out in 2005
  - Adobe bought Flash
  - JScript followed the standards
- But browser's implementations of the language still vary

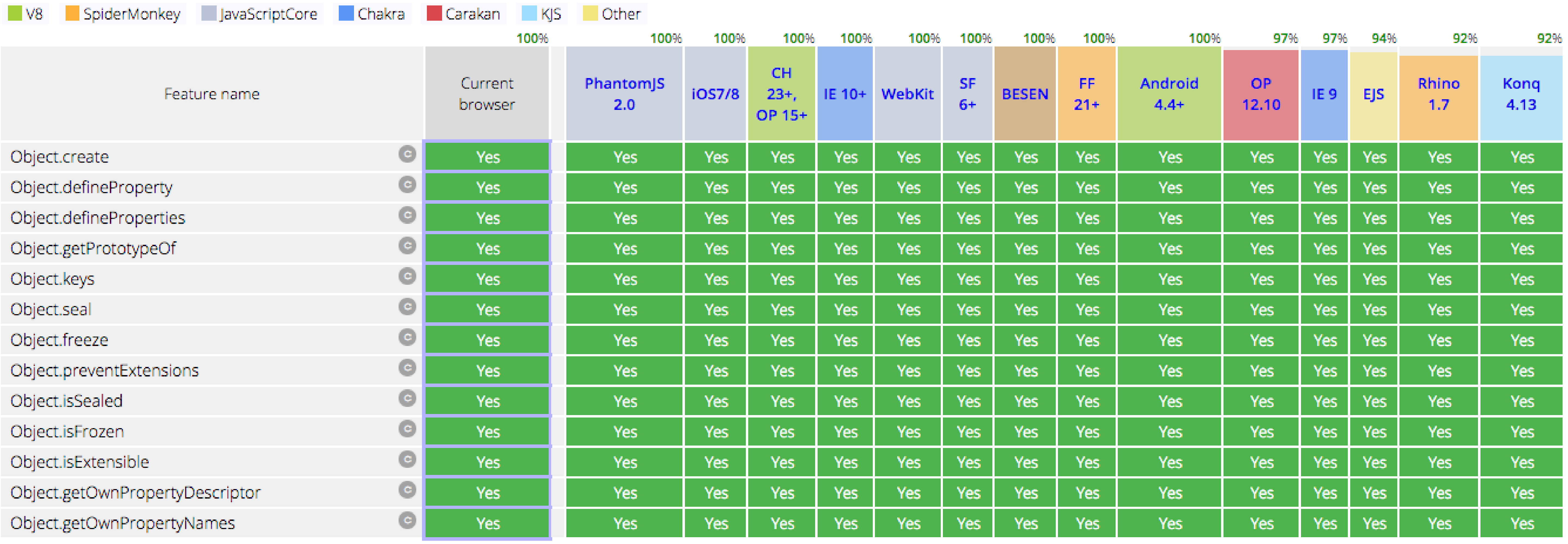


🟩 V8 🟧 SpiderMonkey ⬜ JavaScriptCore 🟦 Chakra 🟥 Carakan 🟦 KJS 🟨 Other

	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	97%	97%	94%	92%	92%
Feature name	Current browser	PhantomJS 2.0	iOS7/8	CH 23+, OP 15+	IE 10+	WebKit	SF 6+	BESEN	FF 21+	Android 4.4+	OP 12.10	IE 9	EJS	Rhino 1.7	Konq 4.13
Object.create	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

History of JavaScript

- JavaScript Engines
  - SpiderMonkey (Firefox)
  - V8 (Chrome)
  - JavaScriptCore (Safari)
  - Carakan (Opera)
  - Chakra (IE & Edge)



🟩 V8 🟧 SpiderMonkey ▨ JavaScriptCore 🟦 Chakra 🟥 Carakan 🟦 KJS 🟨 Other

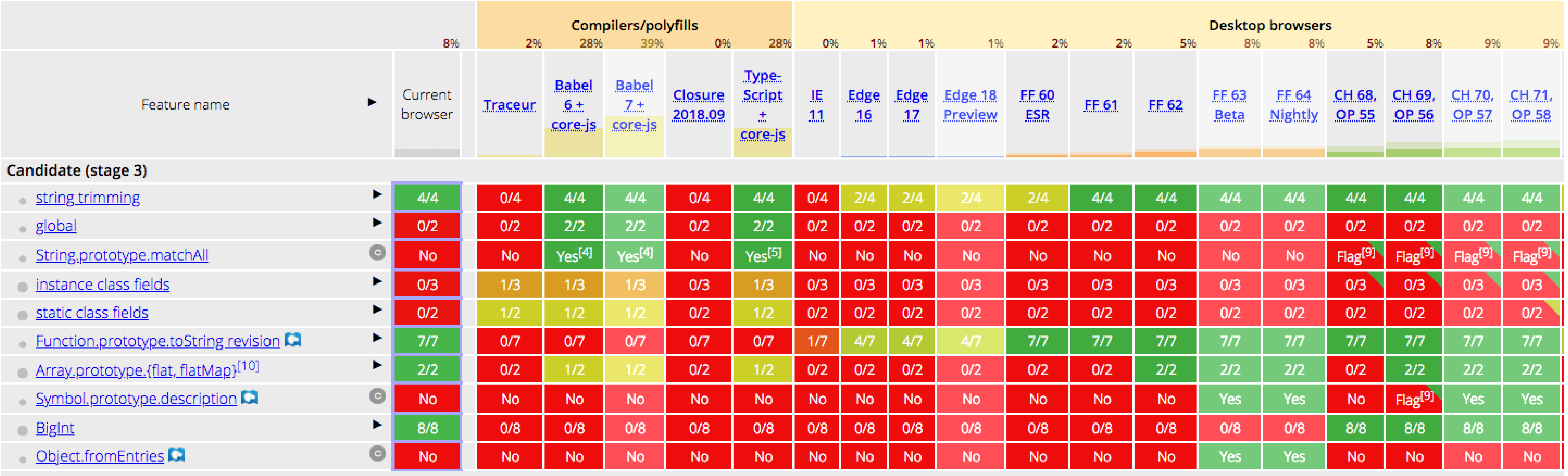
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	97%	97%	94%	92%	92%
Feature name	Current browser	PhantomJS 2.0	iOS7/8	CH 23+, OP 15+	IE 10+	WebKit	SF 6+	BESEN	FF 21+	Android 4.4+	OP 12.10	IE 9	EJS	Rhino 1.7	Konq 4.13
Object.create	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperty	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.defineProperties	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getPrototypeOf	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.keys	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.seal	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.freeze	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.preventExtensions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isSealed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isFrozen	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.isExtensible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyDescriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Object.getOwnPropertyNames	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Versions of JavaScript

- You may see references to ECMAScript
- ECMAScript is just the standard for JavaScript
  - The last “major” release was ECMAScript 6, or ES6, or ECMAScript 2015, or ES2015
  - The latest is ECMAScript 2020, or ES11, or ES2020 (released in June 2020)

Versions of JavaScript

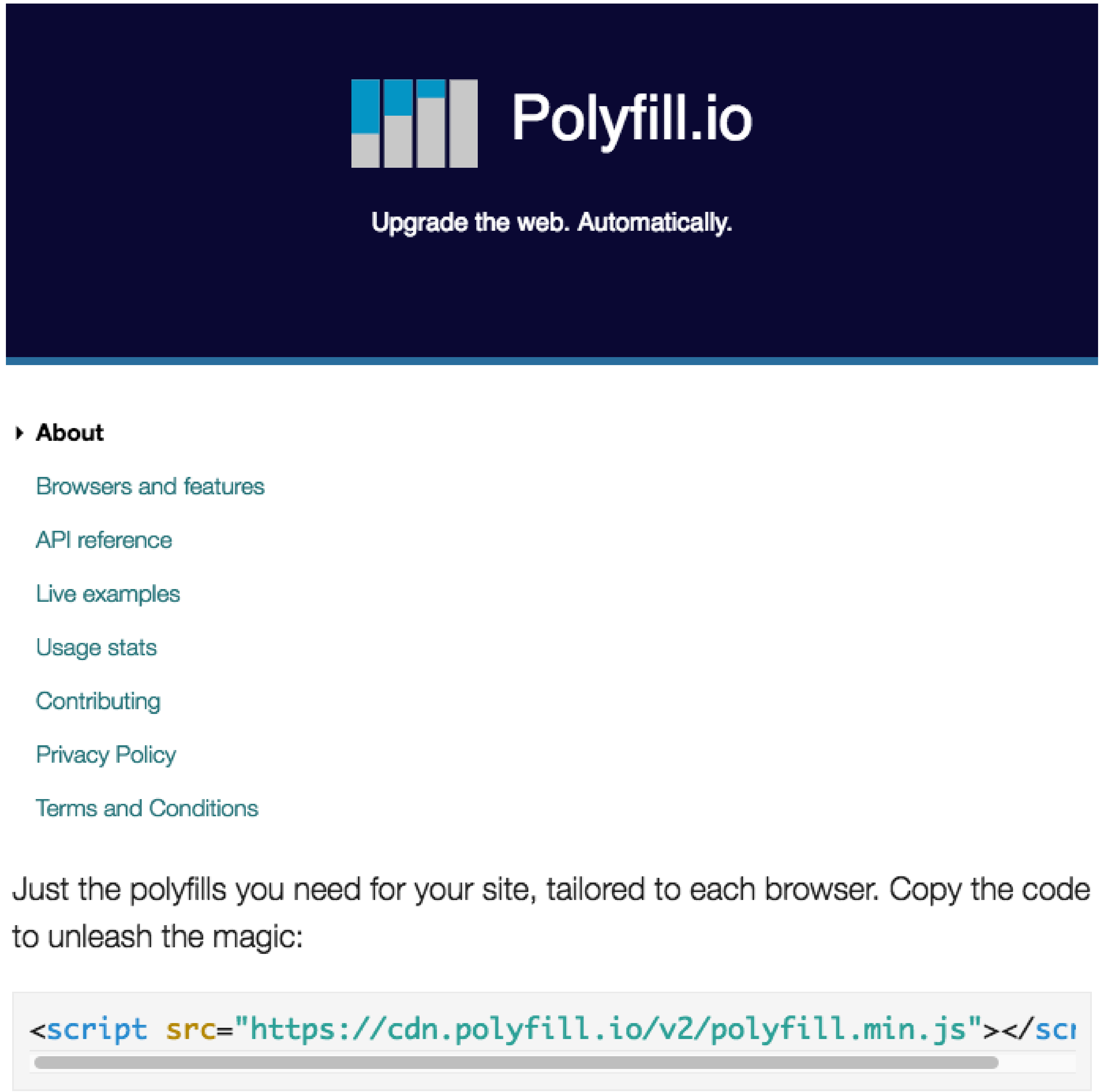
- Engines/Browsers continually play catch-up,
  so many tools support slightly older versions of the standard



		Compilers/polyfills					Desktop browsers												
	8%	2%	28%	39%	0%	28%	0%	1%	1%	1%	2%	2%	5%	8%	8%	5%	8%	9%	9%
Feature name ▶	Current browser	Traceur	Babel 6 + core-js	Babel 7 + core-js	Closure 2018.09	Type-Script + core-js	IE 11	Edge 16	Edge 17	Edge 18 Preview	FF 60 ESR	FF 61	FF 62	FF 63 Beta	FF 64 Nightly	CH 68, OP 55	CH 69, OP 56	CH 70, OP 57	CH 71, OP 58
Candidate (stage 3)																			
string trimming ▶	4/4	0/4	4/4	4/4	0/4	4/4	0/4	2/4	2/4	2/4	2/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
global ▶	0/2	0/2	2/2	2/2	0/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
String.prototype.matchAll	No	No	Yes[4]	Yes[4]	No	Yes[5]	No	No	No	No	No	No	No	No	No	Flag[9]	Flag[9]	Flag[9]	Flag[9]
instance class fields ▶	0/3	1/3	1/3	1/3	0/3	1/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3
static class fields ▶	0/2	1/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
Function.prototype.toString revision ▶	7/7	0/7	0/7	0/7	0/7	0/7	1/7	4/7	4/7	4/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
Array.prototype.{flat, flatMap}[10] ▶	2/2	0/2	1/2	1/2	0/2	1/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2
Symbol.prototype.description	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	No	Flag[9]	Yes	Yes
BigInt ▶	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8
Object.fromEntries	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes	No	No	No	No

Versions of JavaScript

- Polyfills ensure a user's browser has the latest libraries
  - Downloads "fill" versions of added functions, re-written using existing functions
  - Recreates missing features for older browsers
- Sometimes called a "shim" or a "fallback"



Just the polyfills you need for your site, tailored to each browser. Copy the code to unleash the magic:

```
<script src="https://cdn.polyfill.io/v2/polyfill.min.js"></sc
```

JavaScript

- Interpreted language
- Executed by a JavaScript engine
- Engine runs the same code that a programmer writes

Java

- Compiled language (into bytecode)
- Run in a Java Virtual Machine (JVM)
- Bytecode is unreadable by people

JavaScript

- Standardized through ECMAScript, but discrepancies exist
- Debugging dependent on execution environment
- Prototype-based?
- Used in every browser without a plugin

Java

- “Write once, deploy anywhere”
- Bugs found at compile time
- Class-based
- Requires a plugin to be run in most browsers

JavaScript is just a programming language

Printing in JavaScript

```
console.log("Hello, world!");
```

- Won’t be visible in the browser
- Shows in the JavaScript Console

https://repl.it/@m5b/inf133-javascript-demo#index.html

JavaScript Syntax

- Has functions and objects
  - foo() bar.baz
  - They look like Java, but act differently

JavaScript Variables

- Variables are dynamically typed

```
var x = 'hello'; //value is a string
console.log(typeof x); //string

x = 42; //value is now a Number
console.log(typeof x); //number
```

- Unassigned variables have a value of `undefined`

```
var hoursSlept;
console.log(hoursSlept);
```

JavaScript types

```
console.log('40' + 2); //'402'
console.log('40' - 4); //36
```

Minus isn't defined for strings, so JavaScript knows to convert this

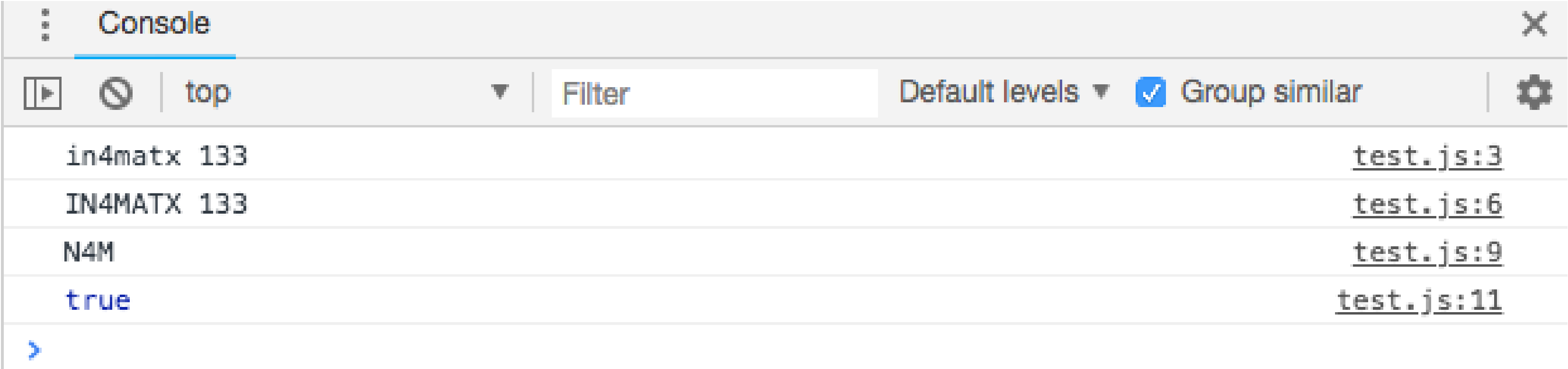
```
var num = 10;
var str = '10';

//comparisons: these will all be booleans (true/false)
console.log(num == str); //true
console.log(num === str); //false
console.log('' == 0); //true
```

=== "strict equality" same as "==" but without type conversion

JavaScript loops and conditionals

```
var i = 4.4;

if(i > 5) {
 console.log('i is bigger than 5');
} else if(i >= 3) {
 console.log('i is between 3 and 5');
} else {
 console.log('i is less than 3');
}

for(var x = 0; x < 5; x++) {
 console.log(x);
}
```

JavaScript methods



```
in4matx 133        test.js:3
IN4MATX 133        test.js:6
N4M                test.js:9
true               test.js:11
>
```

- Called with dot notation

```
var className = 'in4matx 133';
console.log(className);

className = className.toUpperCase();
console.log(className);

var part = className.substring(1, 4);
console.log(part);

console.log(className.indexOf('MATX') >= 0); //whether the substring appears
```

JavaScript arrays

- Similar to Java, but can be a mix of different types

```
var letters = ['a', 'b', 'c'];
var numbers = [1, 2, 3];
var things = ['raindrops', 2.5, true, [5, 9, 8]]; //arrays can be nested
var empty = [];
var blank5 = new Array(5); //empty array with 5 items

//access using [] notation like Java
console.log( letters[1] ); //=> "b"
console.log( things[3][2] ); //=> 8

//assign using [] notation like Java
letters[0] = 'z';
console.log( letters ); //=> ['z', 'b', 'c']

//assigning out of bounds automatically grows the array
letters[10] = 'g';
console.log( letters);
    //=> [ 'z', 'b', 'c', , , , , , , , 'g' ]
console.log( letters.length ); //=> 11
```

JavaScript arrays

- Arrays have their own methods

```
//Make a new array
var array = ['i','n','f','x'];

//add item to end of the array
array.push('133');
console.log(array); //=> ['i','n','f','x','133']

//combine elements into a string
var str = array.join('-');
console.log(str); //=> "i-n-f-x-133"

//get index of an element (first occurrence)
var oIndex = array.indexOf('x'); //=> 3

//remove 1 element starting at oIndex
array.splice(oIndex, 1);
console.log(array); //=> [‘i','n','f','133']
```

JavaScript objects

- An unordered set of key and value pairs
  - Like a HashMap in Java or a dictionary in Python
  - Sometimes called *associative arrays*

Quotes around keys are optional

```
ages = {alice:40, bob:35, charles:13}
extensions = {‘mark':1622, 'in4matx':9937}
num_words = {1:'one', 2:'two', 3:'three'}
things = {num:12, dog:'woof', list:[1,2,3]}
empty = {}
empty = new Object(); //empty object
```

Goals for this Lecture

By the end of this lecture, you should be able to…

- Explain the relative threshold and ceilings of visualization tools like Protovis, D3, and Vega-Lite
- Describe common visualization primitives like marks, axes, and scales
- Implement simple visualizations with Vega-Lite
- Explain the different roles HTML, CSS, and JavaScript play
- Describe how JavaScript standards evolved
- Follow JavaScript syntax for traditional programming concepts like typing, variable assignment, loops, and conditionals