The Reusable Module Pattern

Lars Verspohl
VISUALIZATION DESIGNER AND DEVELOPER
@lars_vers www.datamake.io

Overview

Reusable module pattern

Convention for modular code

Reusable chart modules

`bespokeChart()`

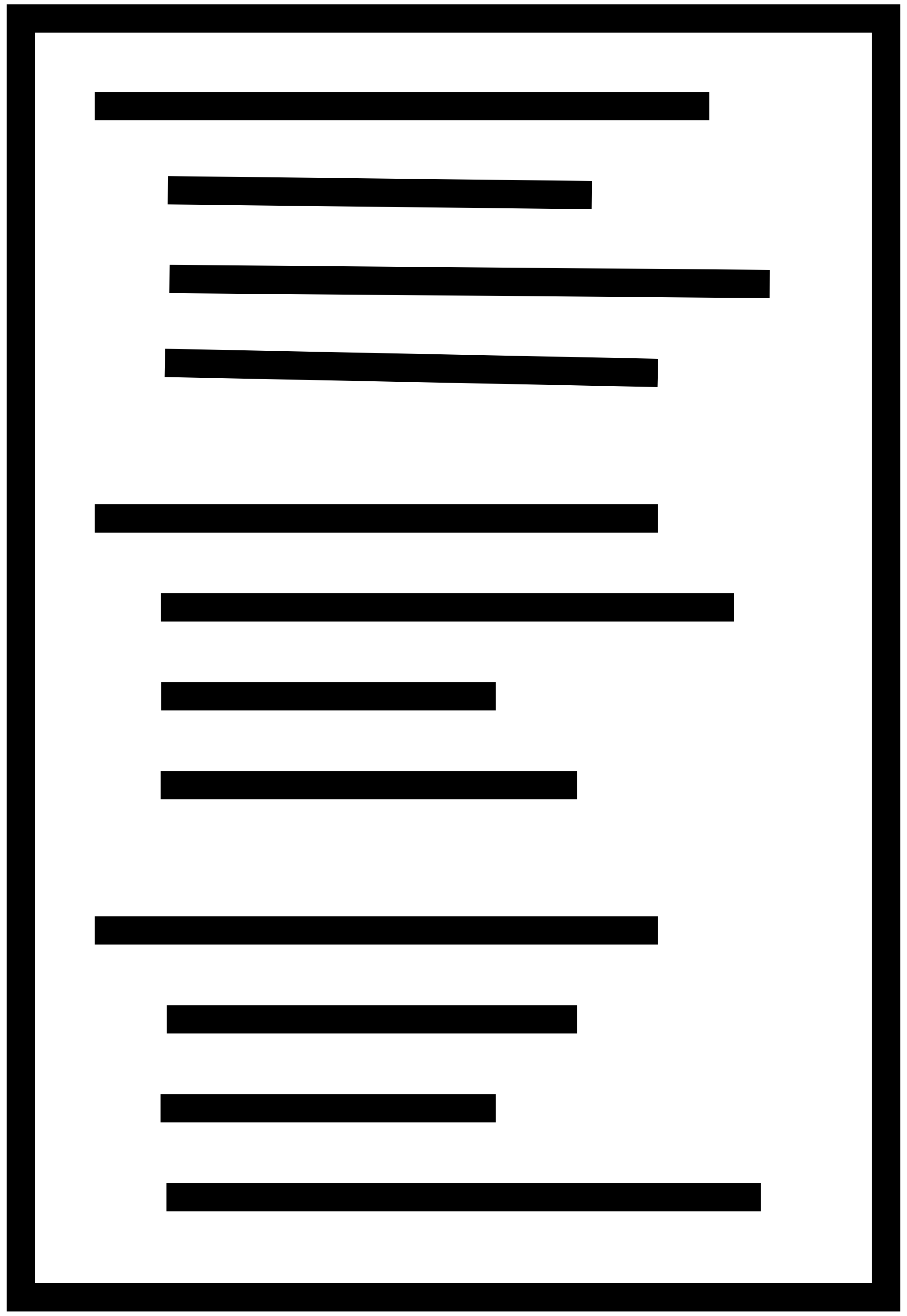
Definition

Mechanics

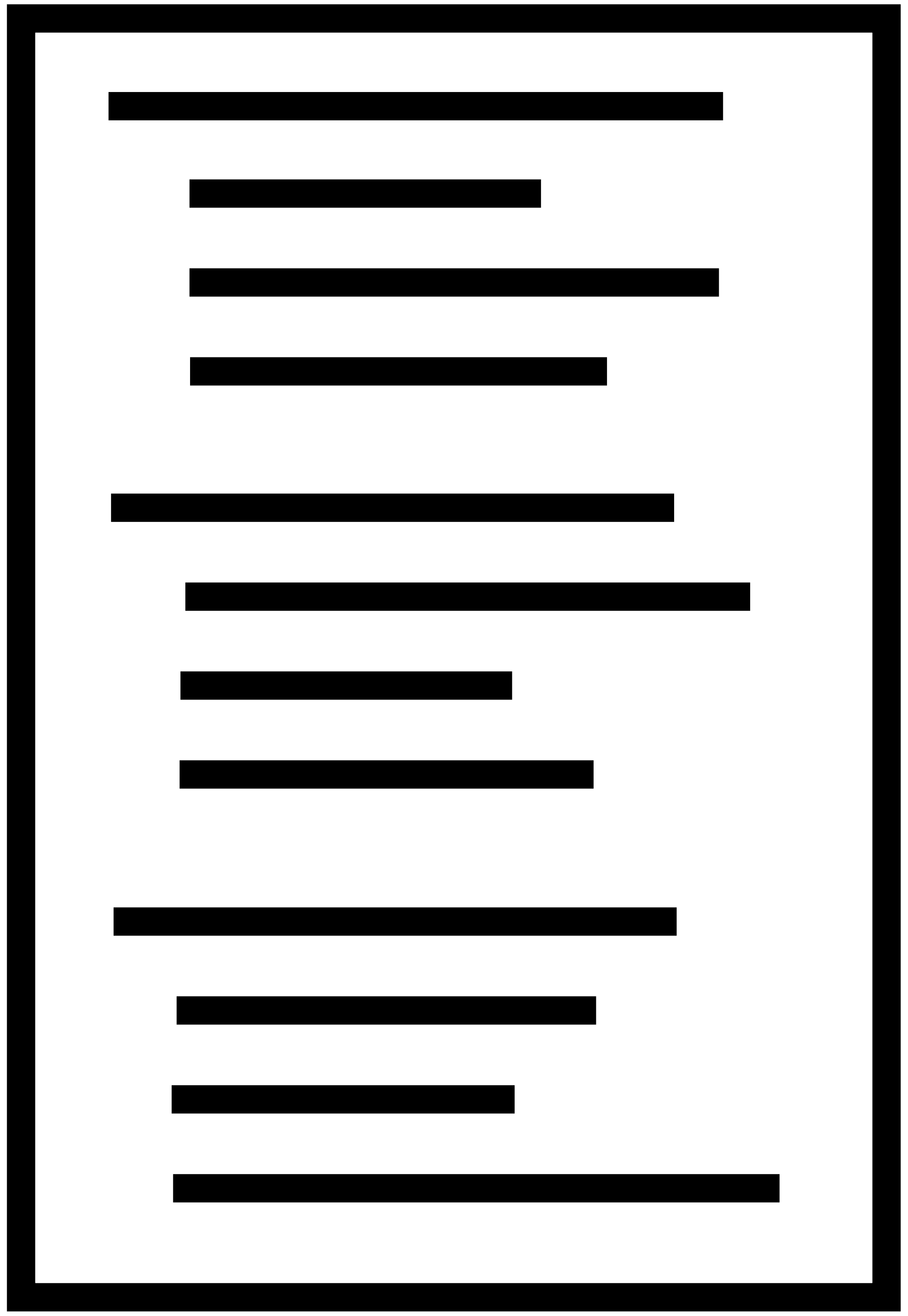
`barChart()` **module**

Small multiples

`barChart()` module



`myLayout()` module



The reusable module pattern
is a function that builds a chart

Using a Module

```
const myBarChart = barChart()
   .width(400)
   .color(‘pink’);
```

Create a module instance

Configure your instance

```
d3.select(‘.container’)
  .datum(myData)
  .call(myBarChart);
```

Use it

Passing in container + data

Typically with `.call()`

Updating Your Module

```
myBarChart.color(‘tomato’);
```

Reconfigure your instance

```
d3.select(‘.container’)
  .datum(myData)
  .call(myBarChart);
```

Call it again to update

Using a Module

```
myBarChart(
  400,
  'pink',
  myData,
  d3.select('.container')
);
```

Why not just call a single function?

Developer needs to keep track of the config's

Using a Module

```
const myBarChart = barChart()
   .width(400)
   .color(‘pink’);


d3.select(‘.container’)
  .datum(myData)
  .call(myBarChart);
```

Re-usable

Re-configurable

`barChart()` stores config

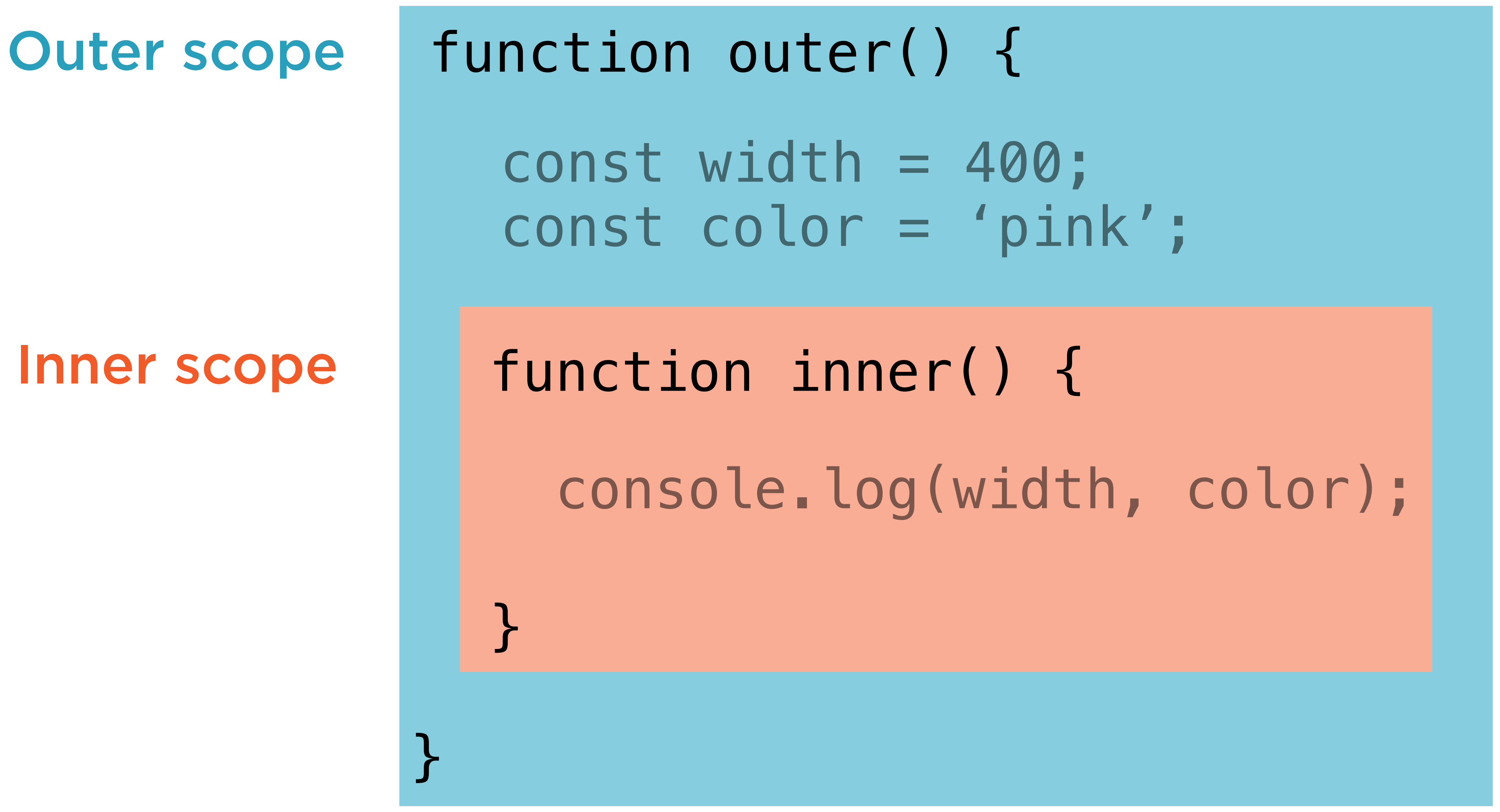
✓

Closures

Closures

```
 function outer() {

   const width = 400;
   const color = ‘pink’;

   function inner() {

     console.log(width, color);

   }

}
```

Closures



```
function outer() {

  const width = 400;
  const color = ‘pink’;

  function inner() {

    console.log(width, color);

  }

}
```

Usage

```
const myBarChart = barChart()
  .width(400);
```

Implementation

```
function barChart() {
  let width = 200;

  function chart() {

    // Build the chart

  }



  return chart;
}
```

Object !

Usage

```
const myBarChart = barChart()
  .width(400);
```

Implementation

```
function barChart() {

  let width = 200;

  function chart() {

    // Build the chart

  }

  chart.width = function(value) {
    // set or get the width
  }

  return chart;

}
```

Usage

```
const myBarChart = barChart()
  .width(400);
```

Implementation

```
function barChart() {

  let width = 200;

  function chart() {

    // Build the chart

  }

  chart.width = function(value) {
    // set or get the width
  }

  return chart;

}
```

Usage

```
const myBarChart = barChart()
  .width(400);
```

Implementation

```
function barChart() {

  let width = 200;

  function chart() {

    // Build the chart

  }

  chart.width = function(value) {
    // set or get the width
  }

  return chart;

}
```

Usage

```
const myBarChart = barChart()
  .width(400);


d3.select(‘.container’)
  .datum(myData)
  .call(myBarChart);
```

Implementation

```
function barChart() {

  let width = 200;

  function chart(container, data) {

    // Build the chart

  }

  chart.width = function(value) {
    // set or get the width
  }

  return chart;

}
```

`.data`

Performs a data join

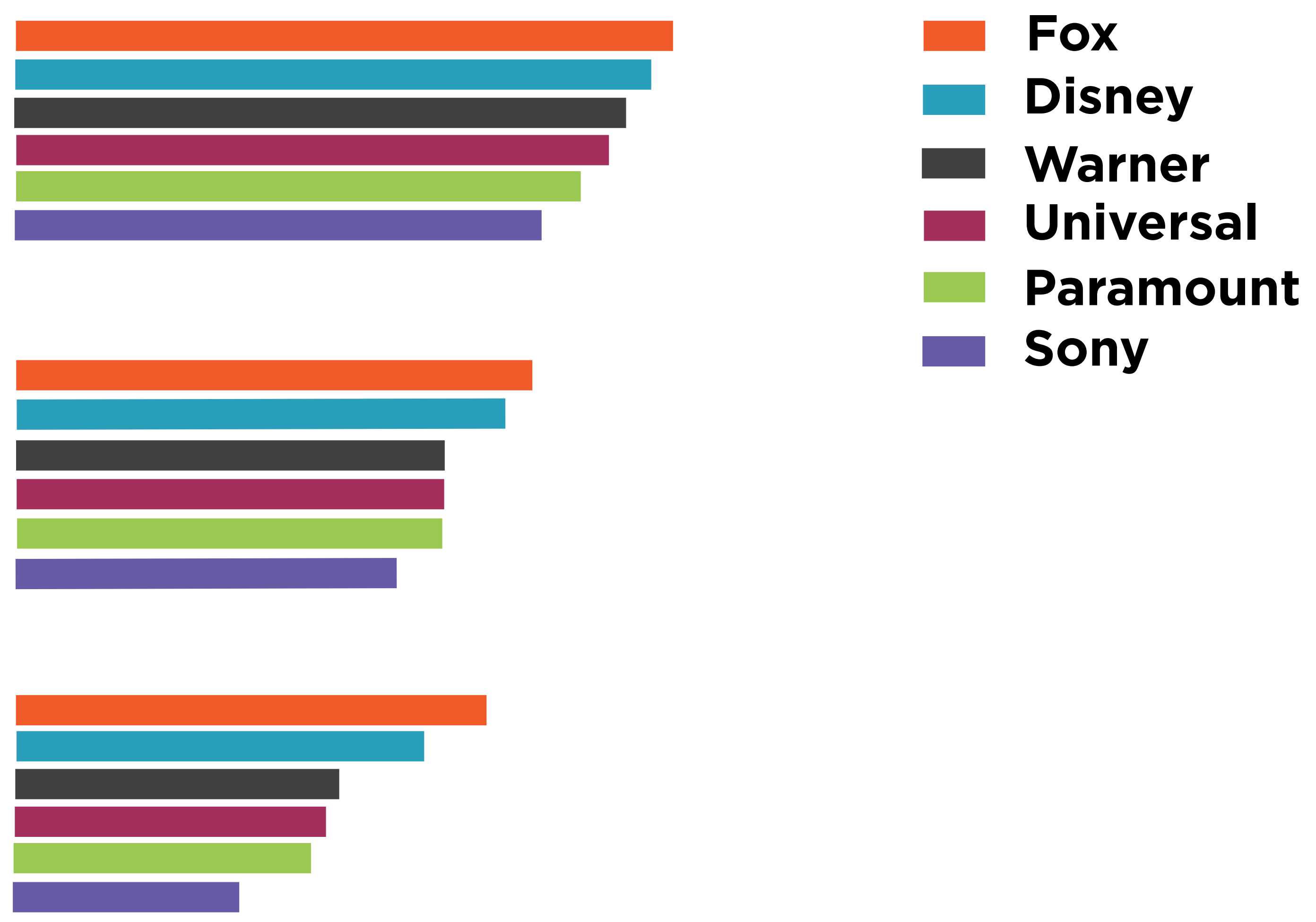
1. As many elements as there are data items
2. Binds data to elements

`.datum`

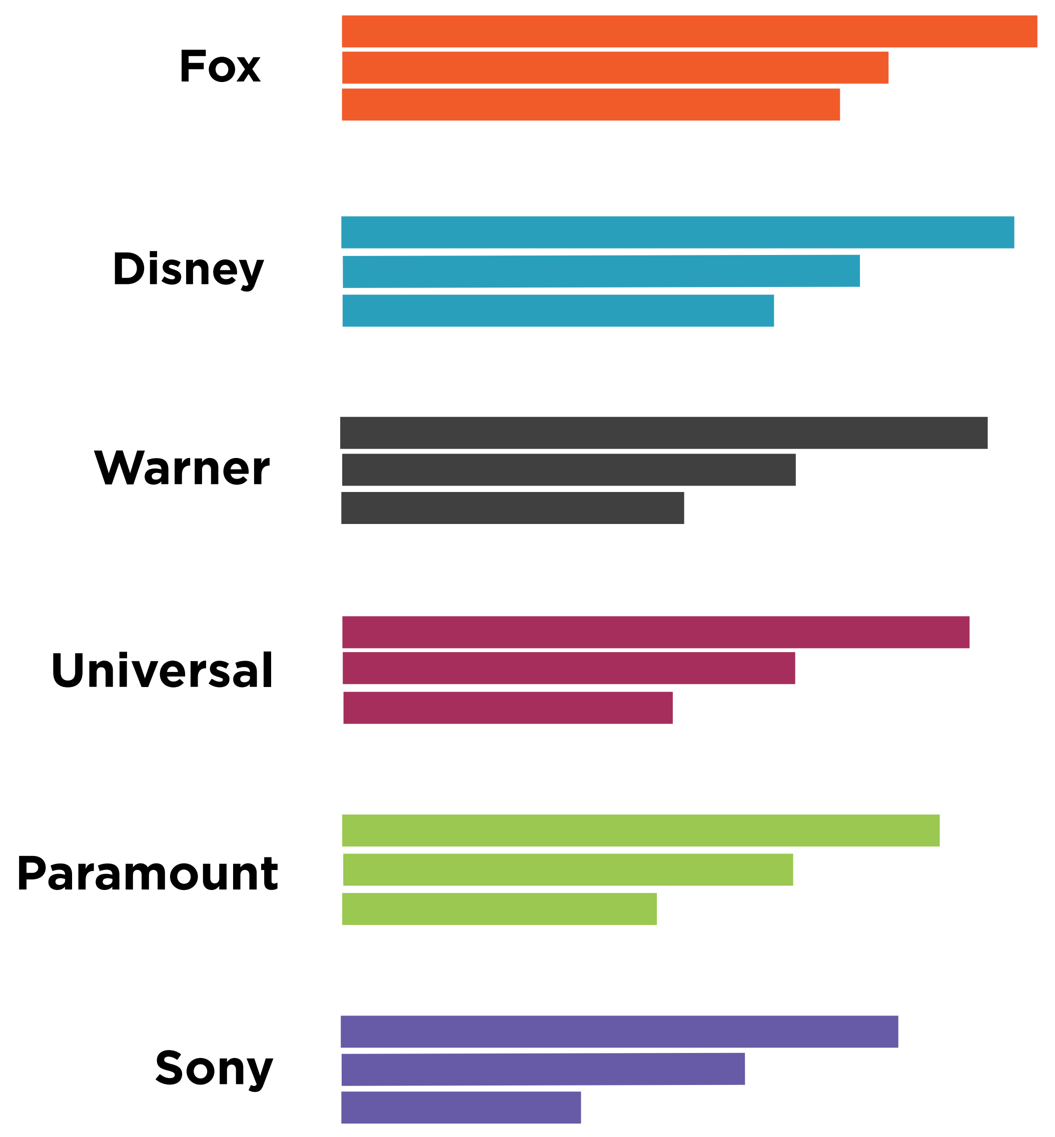
Doesn’t perform a data join

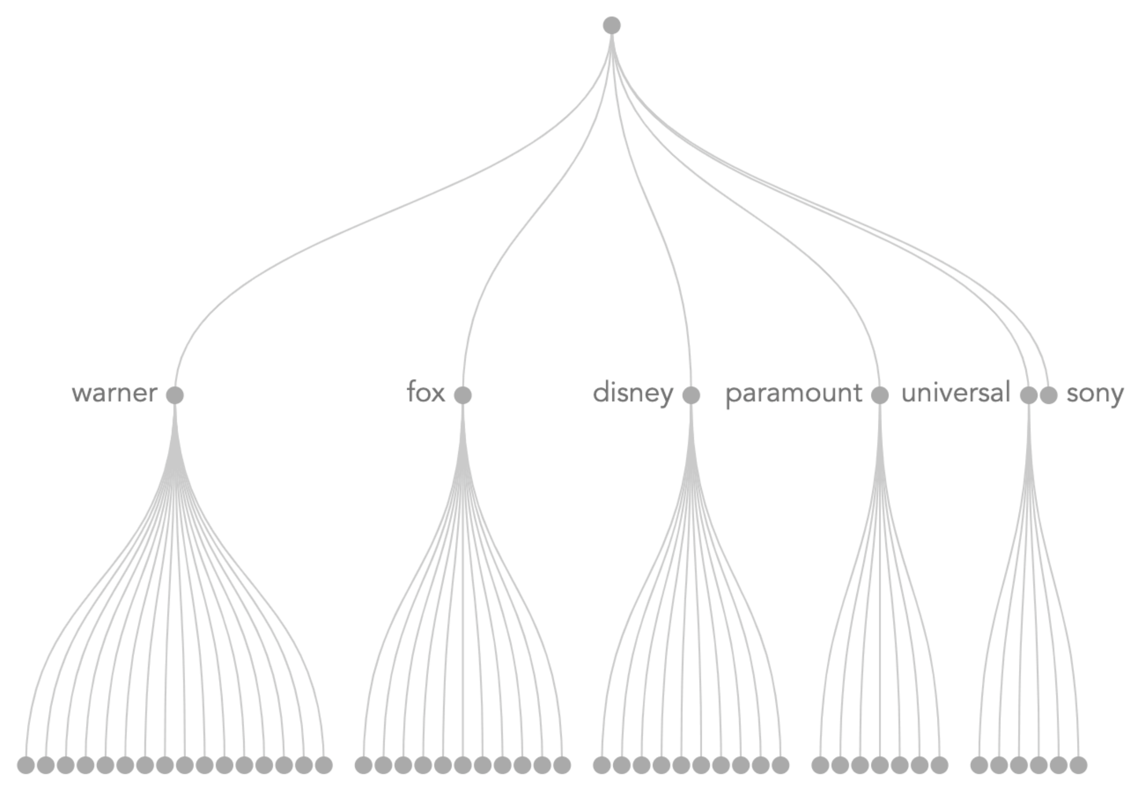
Just binds data to the element

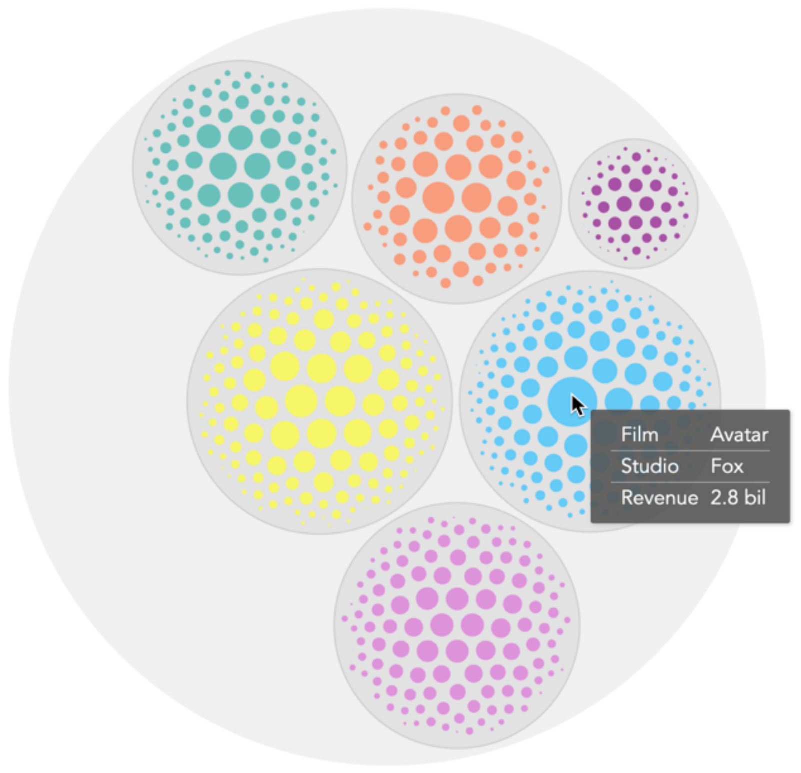
Grouped Bar Chart



Small Multiples

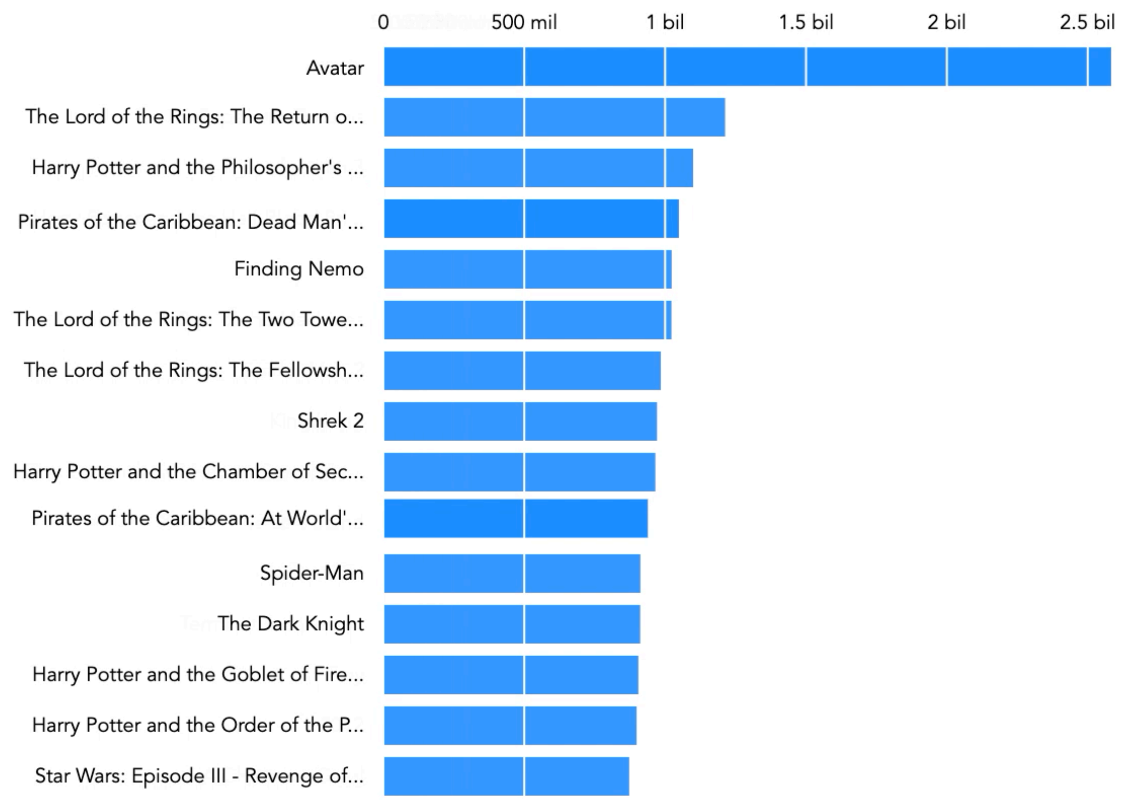






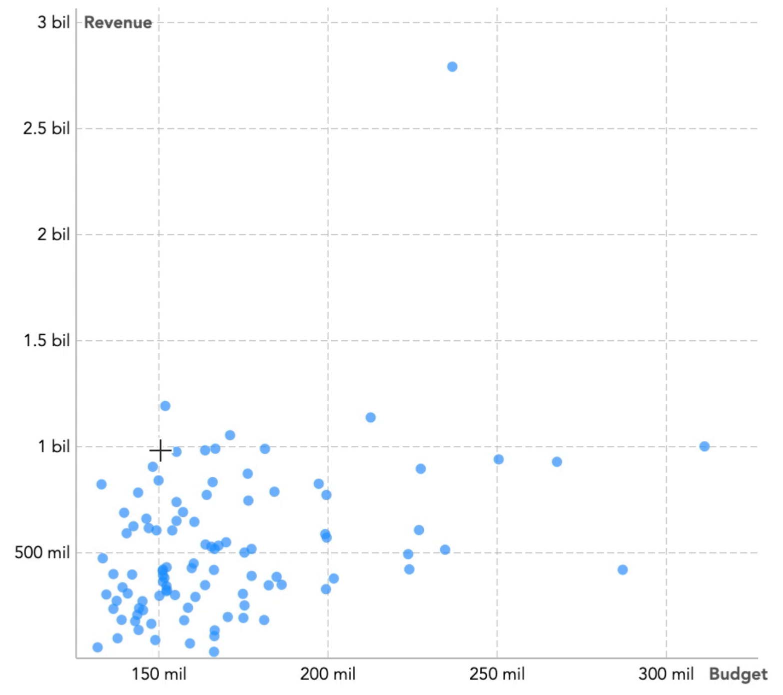
Total budget by title in $US

Top 15 films, 2000-2009



Budget vs. Revenue in $US

Top 100 films by budget, 2000-2009



Selected Elements

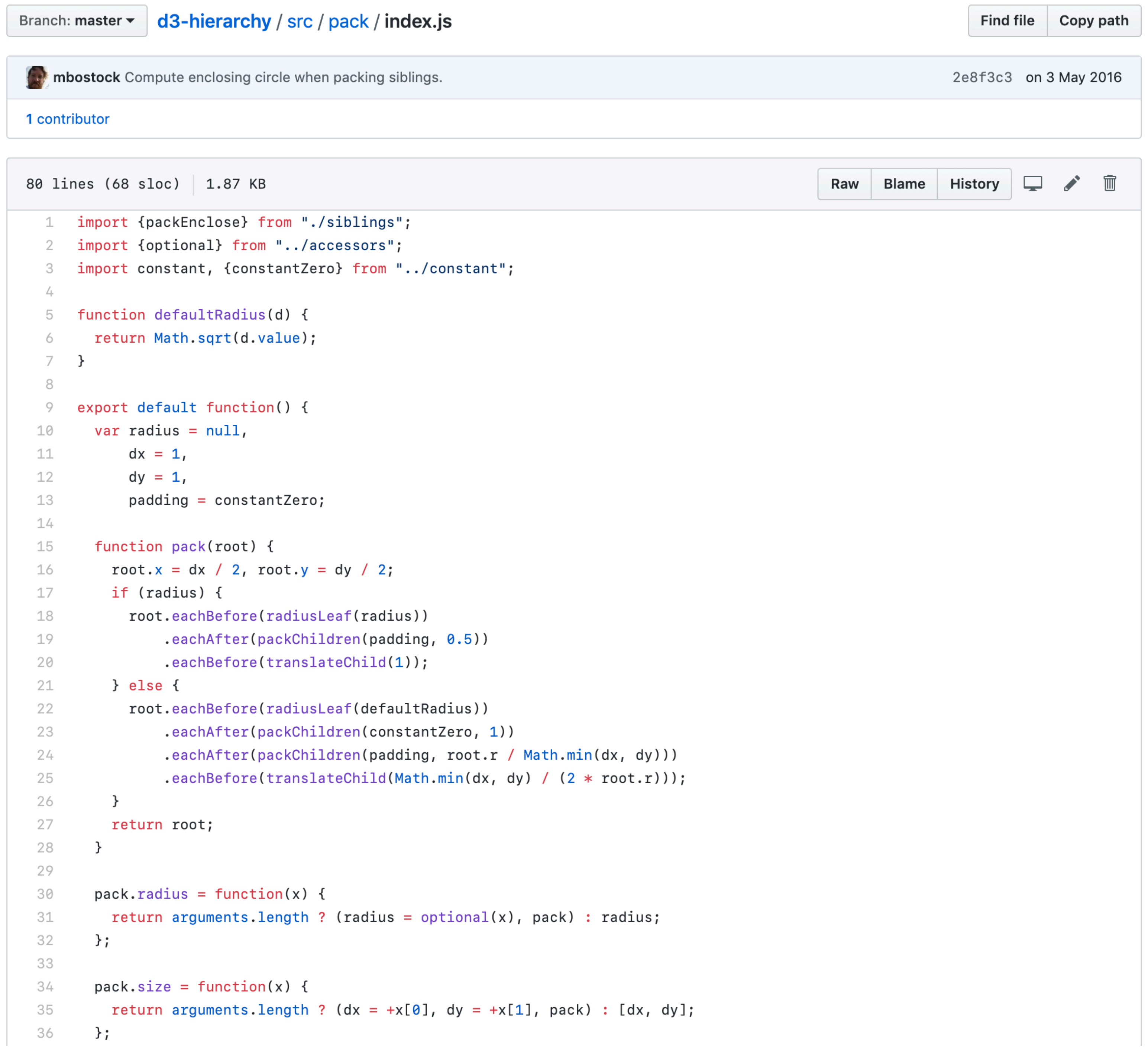
Reusable Module Pattern

Technical

Stepping up

D3 internals

Reusable Module Pattern



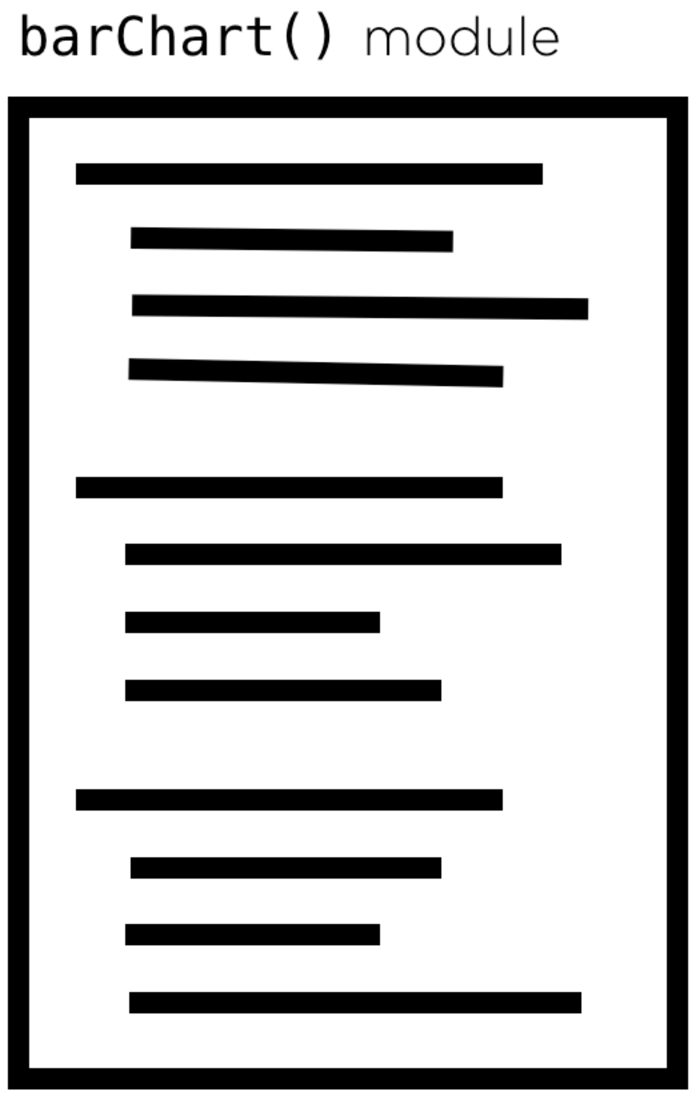
Branch: master ▾ d3-hierarchy / src / pack / index.js

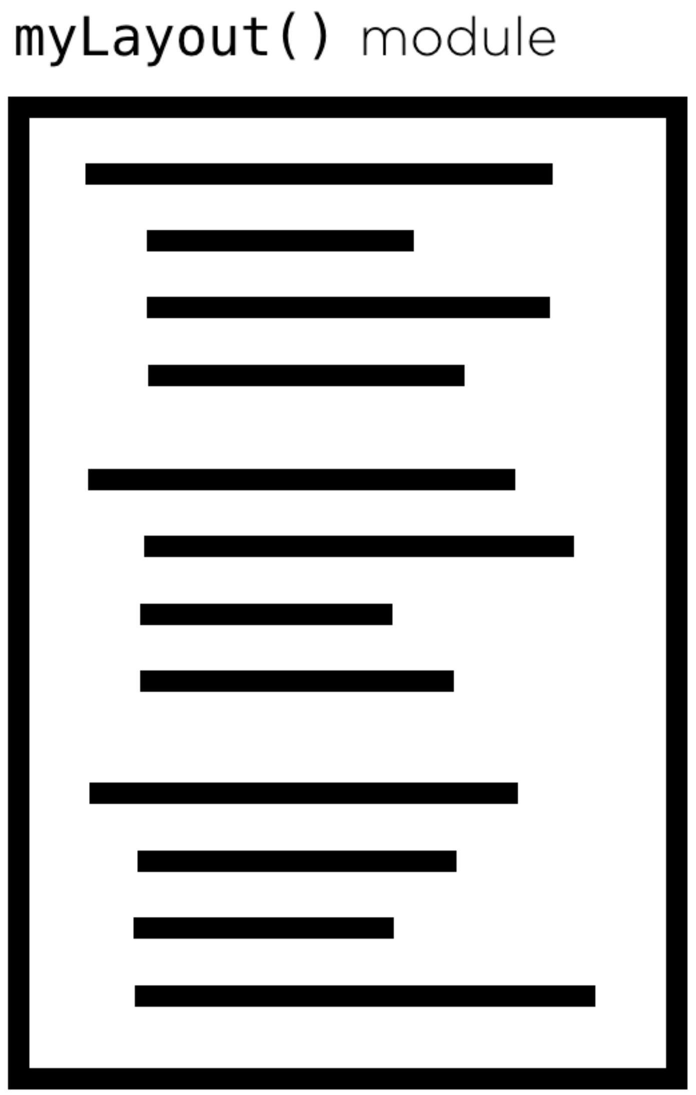
mbostock Compute enclosing circle when packing siblings. 2e8f3c3 on 3 May 2016

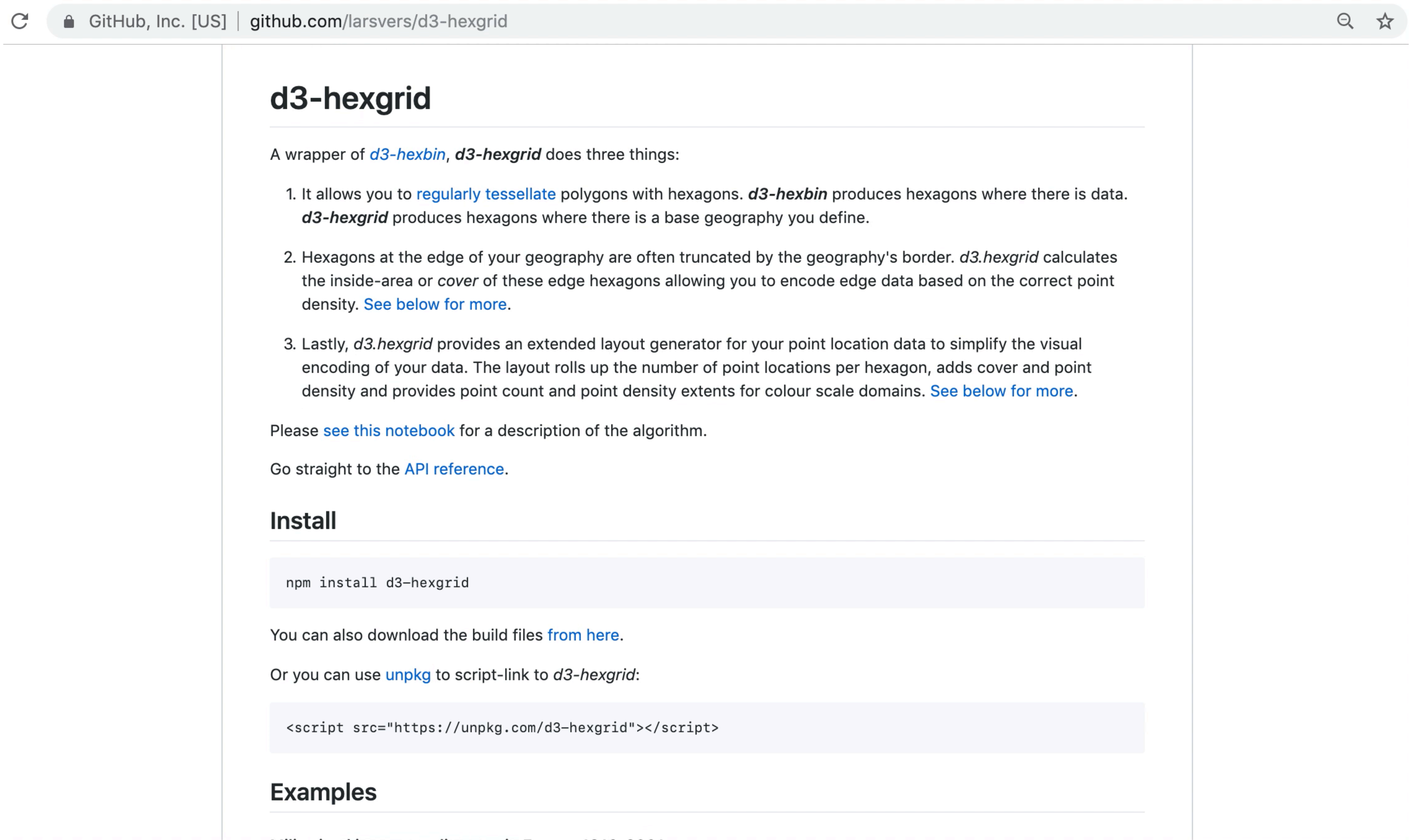
1 contributor

80 lines (68 sloc) | 1.87 KB

```
import {packEnclose} from "./siblings";
import {optional} from "../accessors";
import constant, {constantZero} from "../constant";

function defaultRadius(d) {
  return Math.sqrt(d.value);
}

export default function() {
  var radius = null,
      dx = 1,
      dy = 1,
      padding = constantZero;

  function pack(root) {
    root.x = dx / 2, root.y = dy / 2;
    if (radius) {
      root.eachBefore(radiusLeaf(radius))
          .eachAfter(packChildren(padding, 0.5))
          .eachBefore(translateChild(1));
    } else {
      root.eachBefore(radiusLeaf(defaultRadius))
          .eachAfter(packChildren(constantZero, 1))
          .eachAfter(packChildren(padding, root.r / Math.min(dx, dy)))
          .eachBefore(translateChild(Math.min(dx, dy) / (2 * root.r)));
    }
    return root;
  }

  pack.radius = function(x) {
    return arguments.length ? (radius = optional(x), pack) : radius;
  };

  pack.size = function(x) {
    return arguments.length ? (dx = +x[0], dy = +x[1], pack) : [dx, dy];
  };
```







GitHub, Inc. [US] | github.com/larsvers/d3-hexgrid

d3-hexgrid

A wrapper of *d3-hexbin*, ***d3-hexgrid*** does three things:

1. It allows you to regularly tessellate polygons with hexagons. ***d3-hexbin*** produces hexagons where there is data. ***d3-hexgrid*** produces hexagons where there is a base geography you define.
2. Hexagons at the edge of your geography are often truncated by the geography's border. *d3.hexgrid* calculates the inside-area or *cover* of these edge hexagons allowing you to encode edge data based on the correct point density. See below for more.
3. Lastly, *d3.hexgrid* provides an extended layout generator for your point location data to simplify the visual encoding of your data. The layout rolls up the number of point locations per hexagon, adds cover and point density and provides point count and point density extents for colour scale domains. See below for more.

Please see this notebook for a description of the algorithm.

Go straight to the API reference.

Install

```
npm install d3-hexgrid
```

You can also download the build files from here.

Or you can use unpkg to script-link to *d3-hexgrid*:

```
<script src="https://unpkg.com/d3-hexgrid"></script>
```

Examples

Summary

Reusable module pattern concept

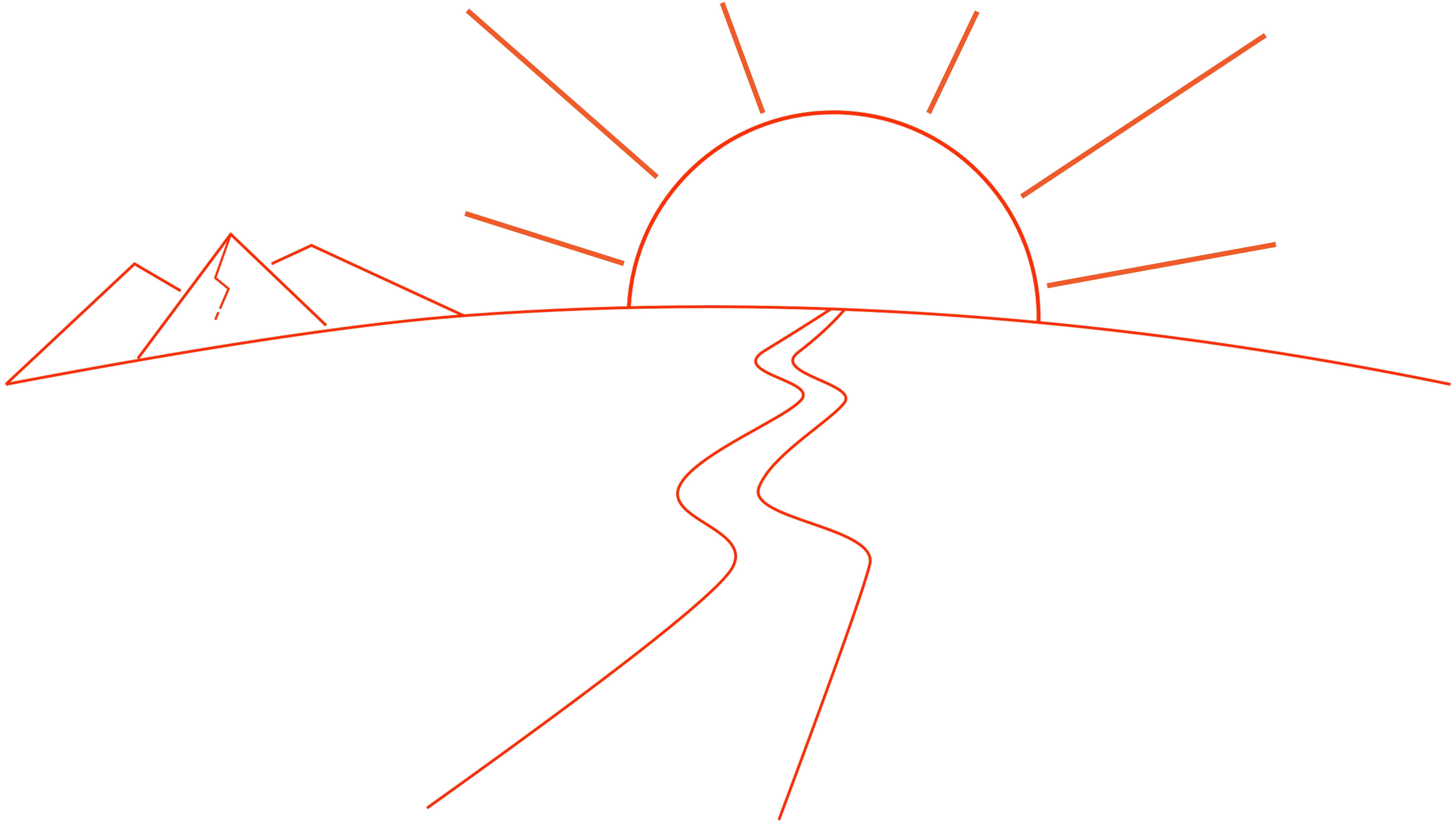
Configure and reuse charts and more

Closures

Getters and setters

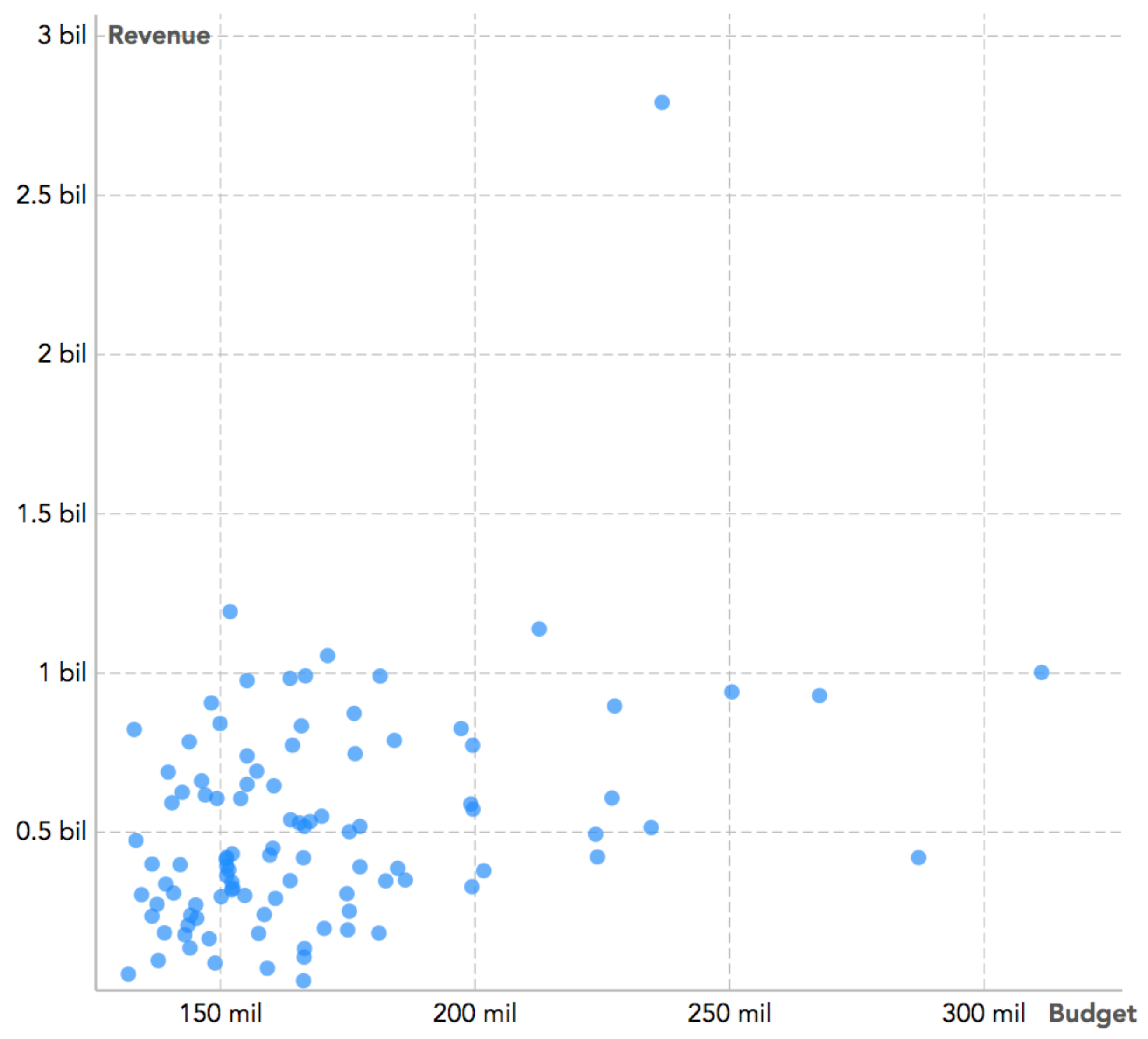
`barChart()` **module**

Small multiples



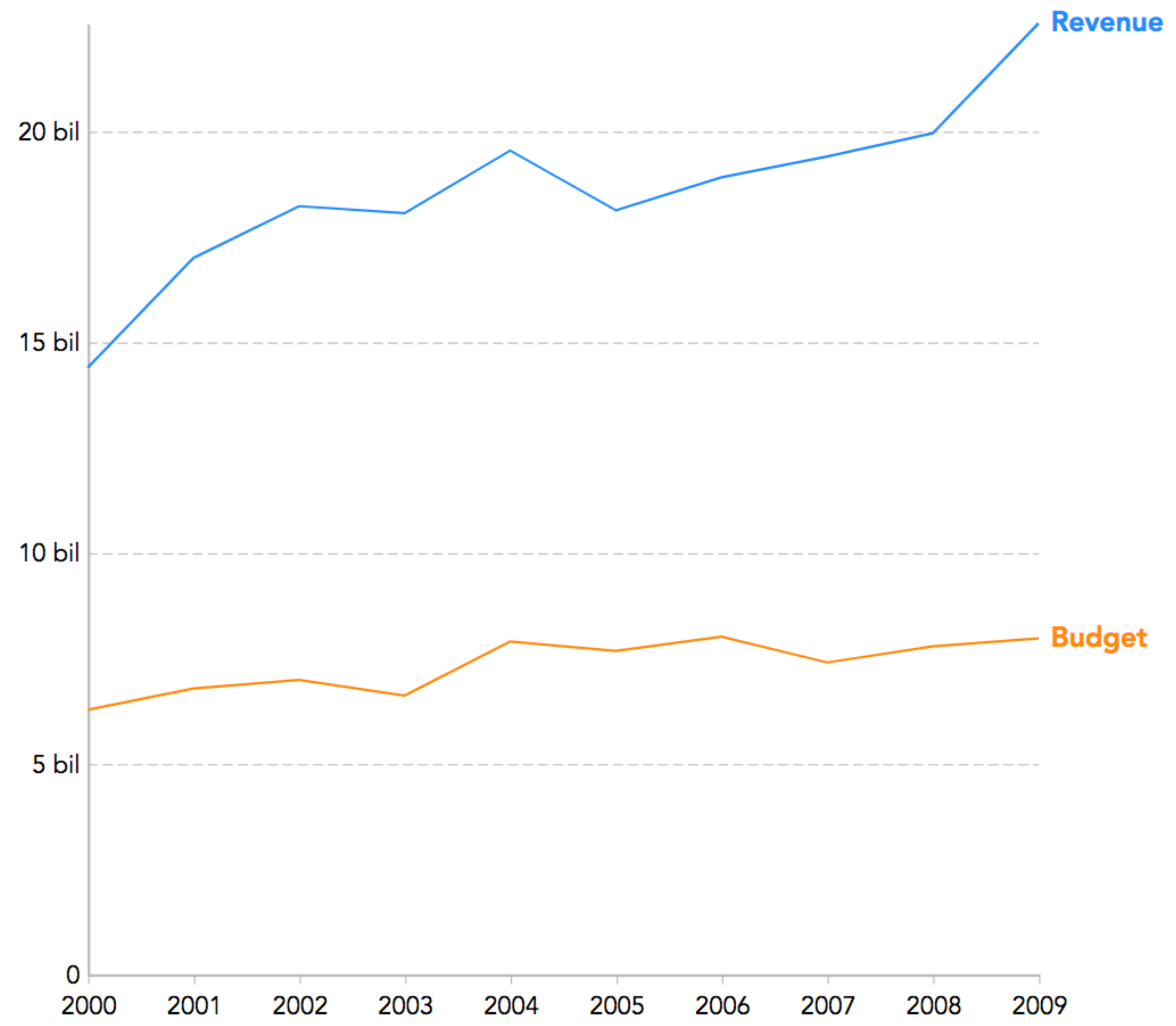
Budget vs. Revenue in $US

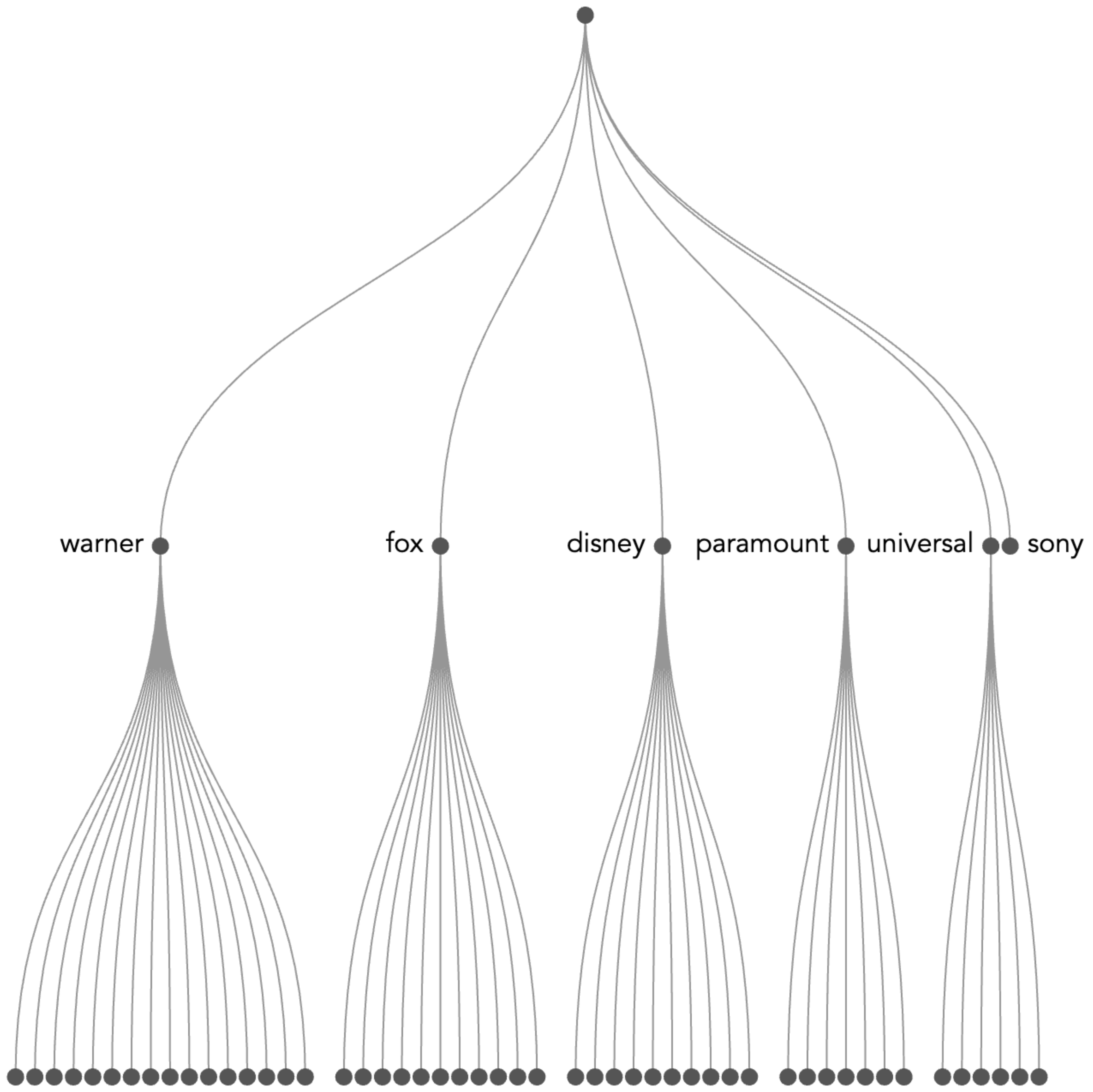
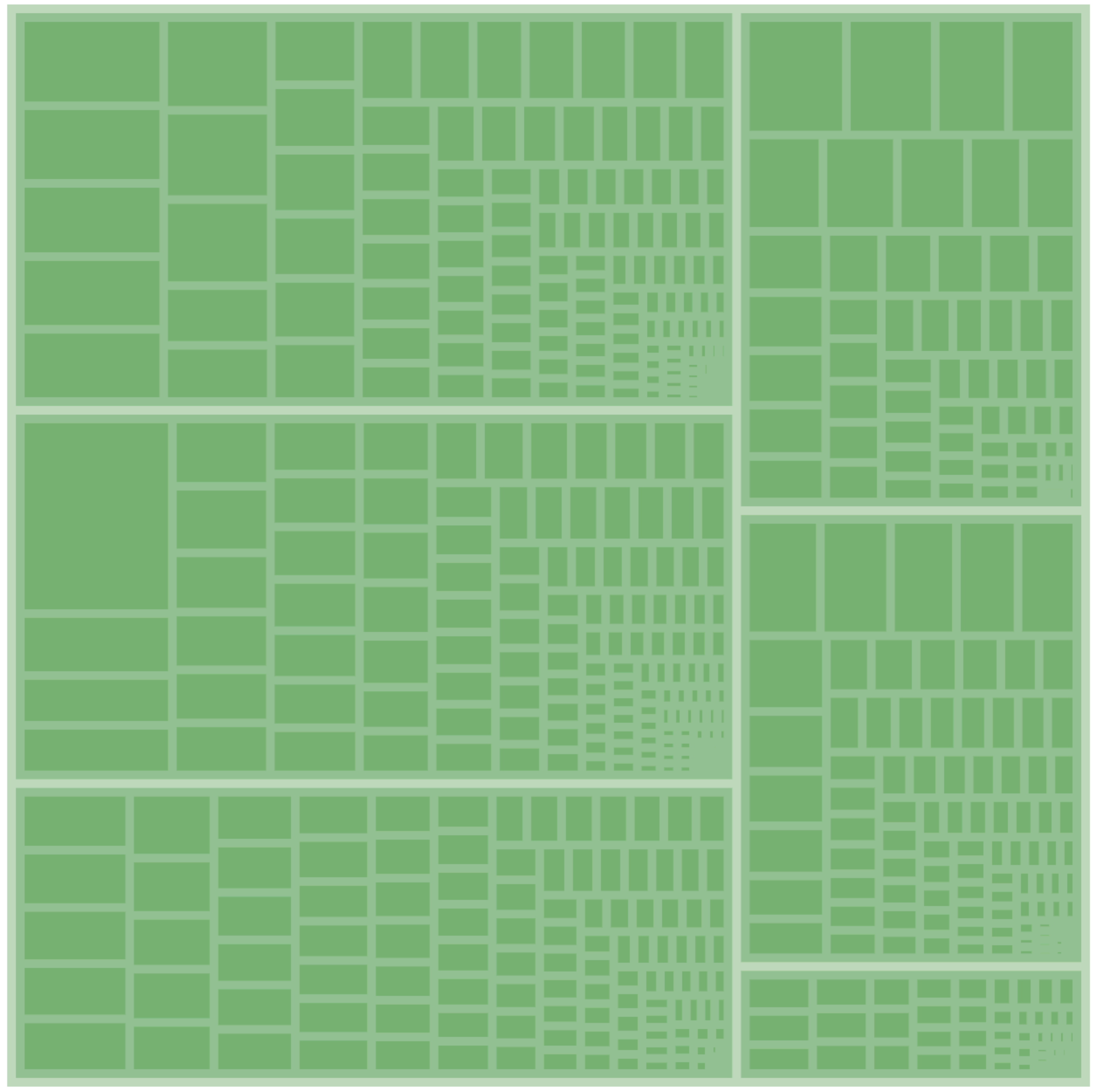
Top 100 films by budget, 2000-2009



Budget and Revenue over time in $US

Films w/ budget and revenue figures, 2000-2009




warner
fox
disney
paramount
universal
sony


D3.js Data Visualization Fundamentals

THANK YOU!

Lars Verspohl
VISUALIZATION DESIGNER AND DEVELOPER
@lars_vers www.datamake.io