DATA DRIVEN DOCUMENTS

```
d3.select(‘.container')
   .selectAll('div')
   .data(BIG_DATA)
   .enter()
   .append('div')
   .attr('class', 'bar')
   .style('width', function(d) {
     return d + 'px';
   })
   .text(function(d) {
     return d;
   })
   .exit();
```

Выборки в D3

```
01: d3.select(‘ul')
02: d3.selectAll(‘li')
```

Выборки в D3

```
01: d3.select('ul')
02: d3.selectAll('li')
```

- select - выбирает первый элемент соответствующий селектору
- selectAll - выбирает все элементы соответствующие селектору

Выборки в D3

```
01: d3.select(document.body);
02: d3.selectAll(
      document.querySelectorAll('li'));
```

- select - выбирает первый элемент соответствующий селектору
- selectAll - выбирает все элементы соответствующие селектору

Можно в качестве аргумента передавать сами элементы DOM дерева.

Выборки в D3

```
01: d3.select(‘ul')
02: d3.selectAll(‘li')
03:
04:
05: d3.select(‘ul’).selectAll(‘li')
```

Можно также выделять под-элементы уже выделенных элементов.

.append

```
01: d3.select('ul').append('li')
```

Append - добавляет новый элемент в выделение и возвращает ссылку на него.

.append

```
01: d3.select(‘ul’).append(‘li’)
02:     .style(‘color’, ‘red’)
03:     .html(‘Hello!’)
```

Append - добавляет новый элемент в выделение и возвращает ссылку на него.

Можно модифицировать атрибуты встроенного элемента прямо в цепочке вызовов.

Связывание данных в D3

Data Binding

```
01: d3.select('body')
02:     .selectAll('div')
03:         .data([1, 2, 3]).enter()
04:     .append('div')
```

Связывание данных в D3

Data Binding

```
01: d3.select('body')
02:     .selectAll('div')
03:         .data([1, 2, 3]).enter()
04:     .append('div')
```

вся магия происходит здесь

Связывание данных в D3

Data Binding

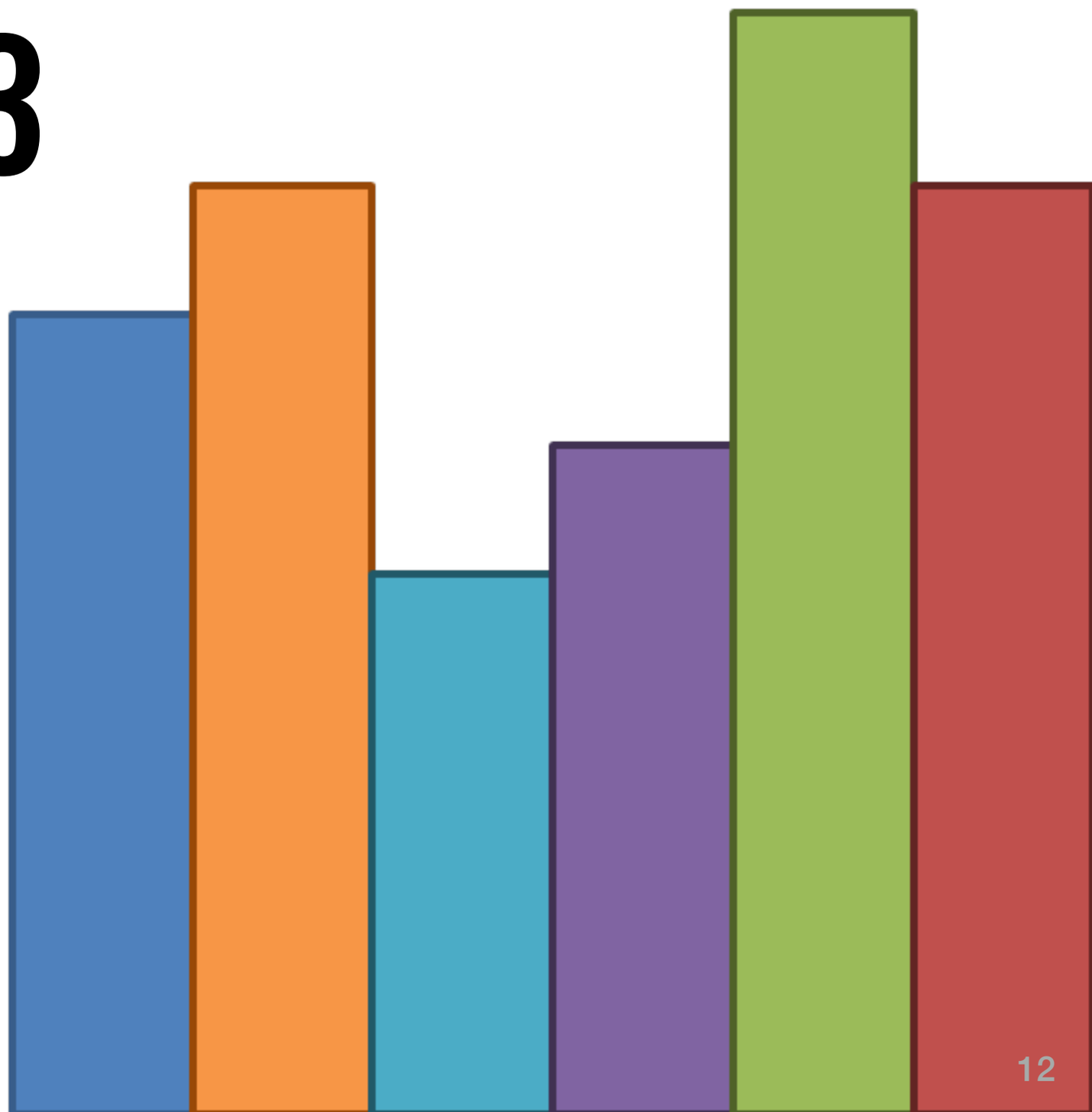
```
01: d3.select('body')
02:     .selectAll('div')
03:         .data([1, 2, 3]).enter()
04:     .append('div')
```

вся магия происходит здесь

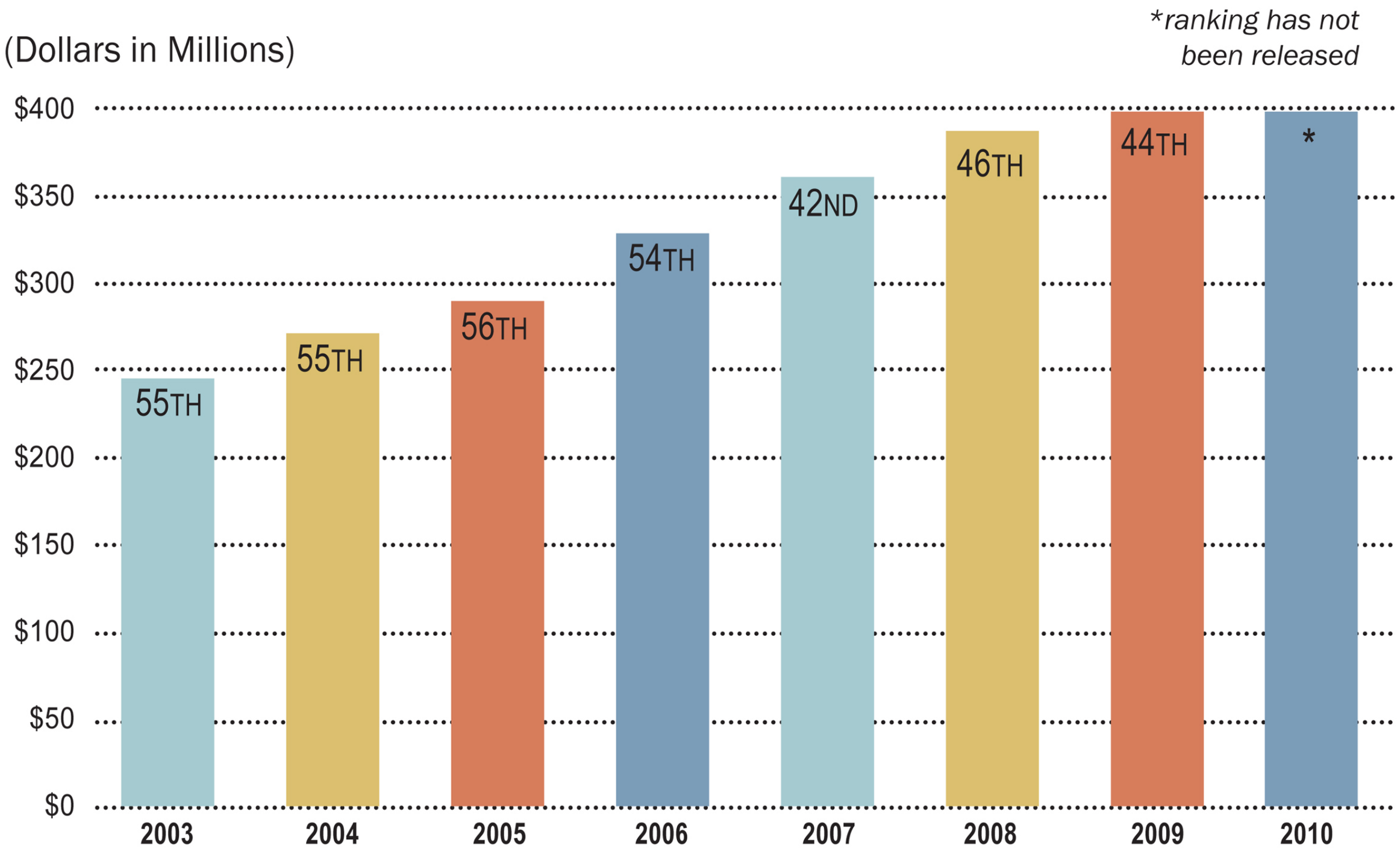
.data резервирует места для данных, которые мы связываем с представлением. Он возвращает три состояния выборки enter, update и exit.

.enter - возвращает зарезервированные места для выборки, с которыми мы можем начать творить нашу магию

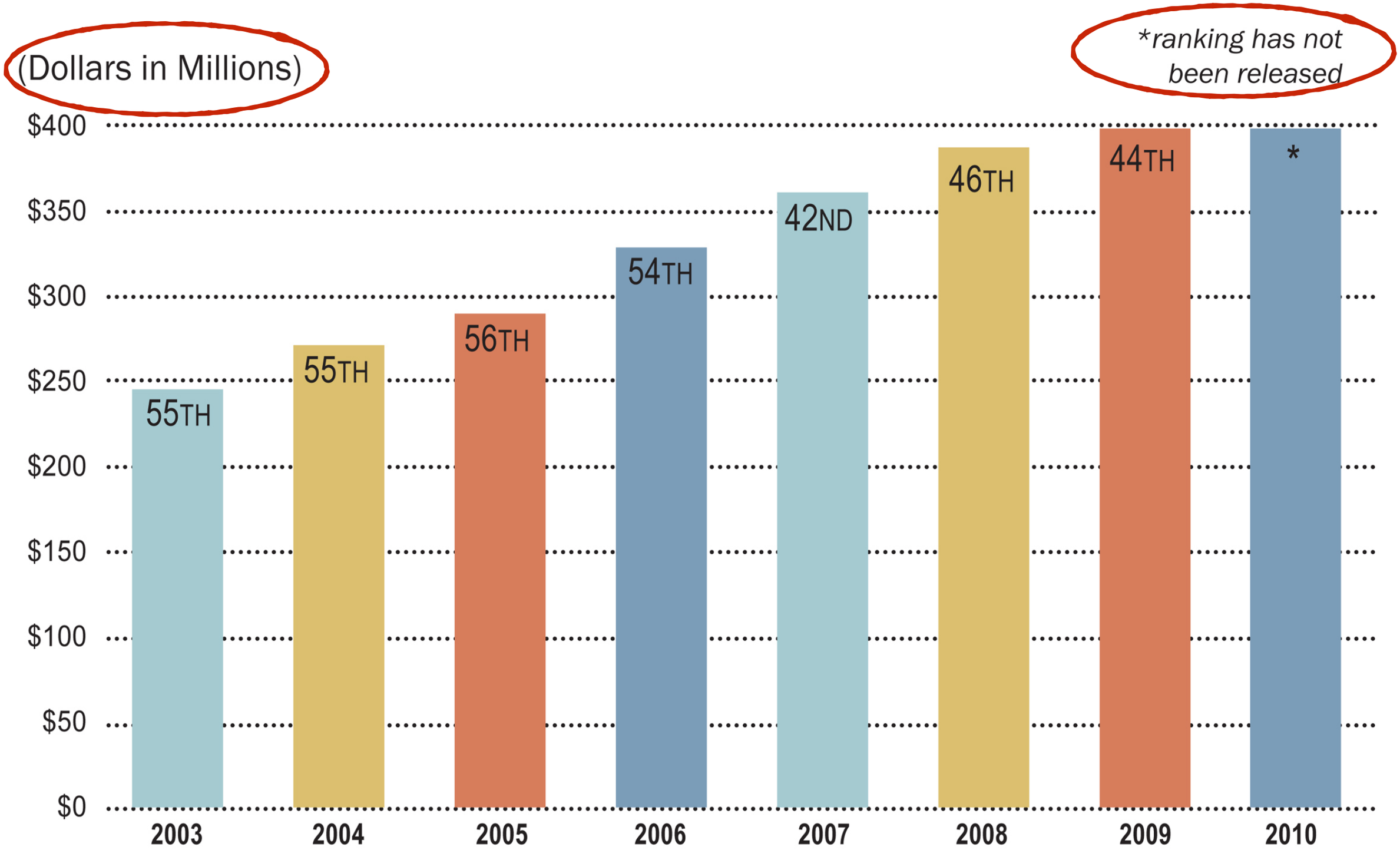
ГИСТОГРАММА SVG И D3



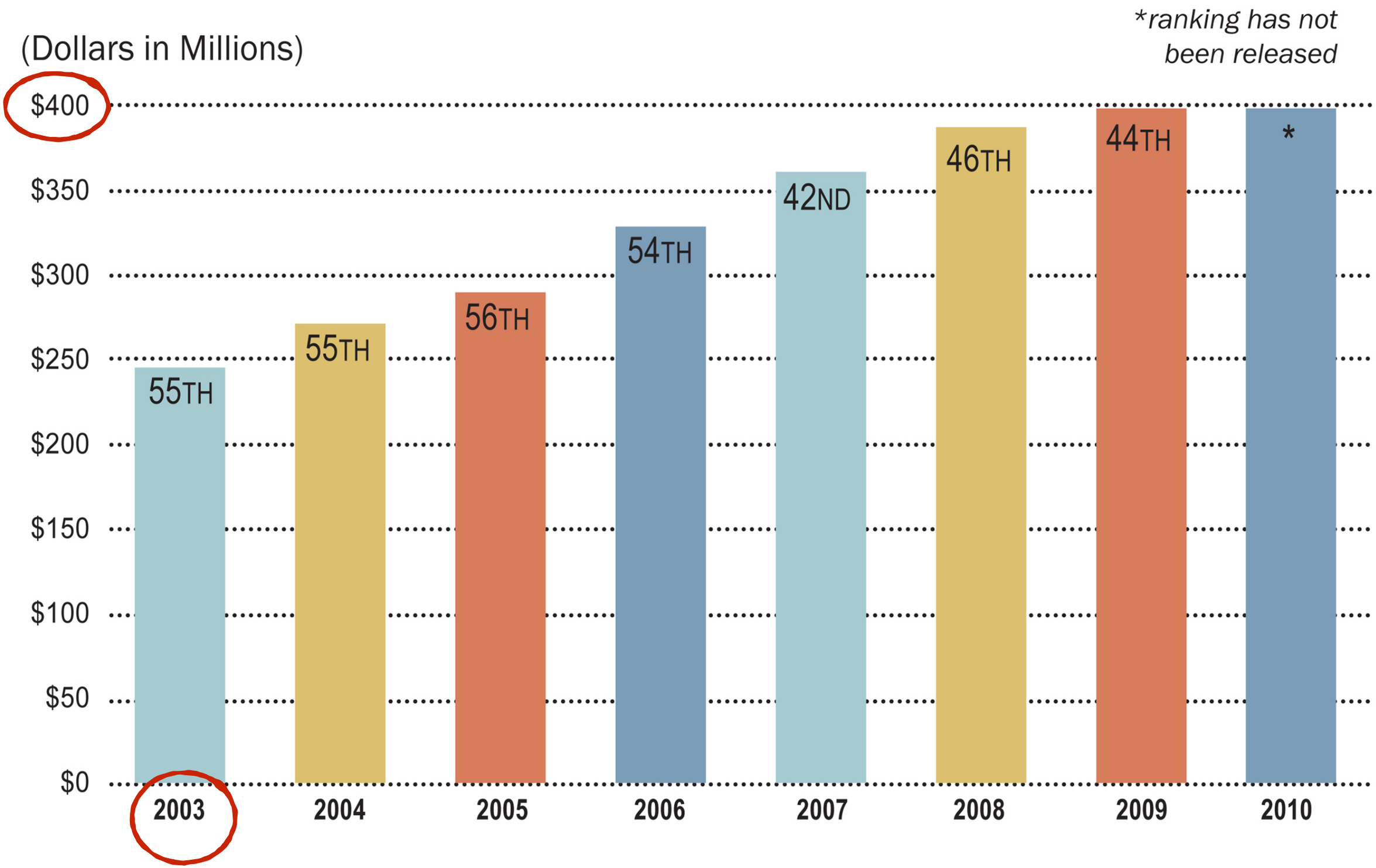
Barchart


*ranking has not been released
(Dollars in Millions)
$400
$350
$300
$250
$200
$150
$100
$50
$0
55TH
55TH
56TH
54TH
42ND
46TH
44TH
*
2003
2004
2005
2006
2007
2008
2009
2010

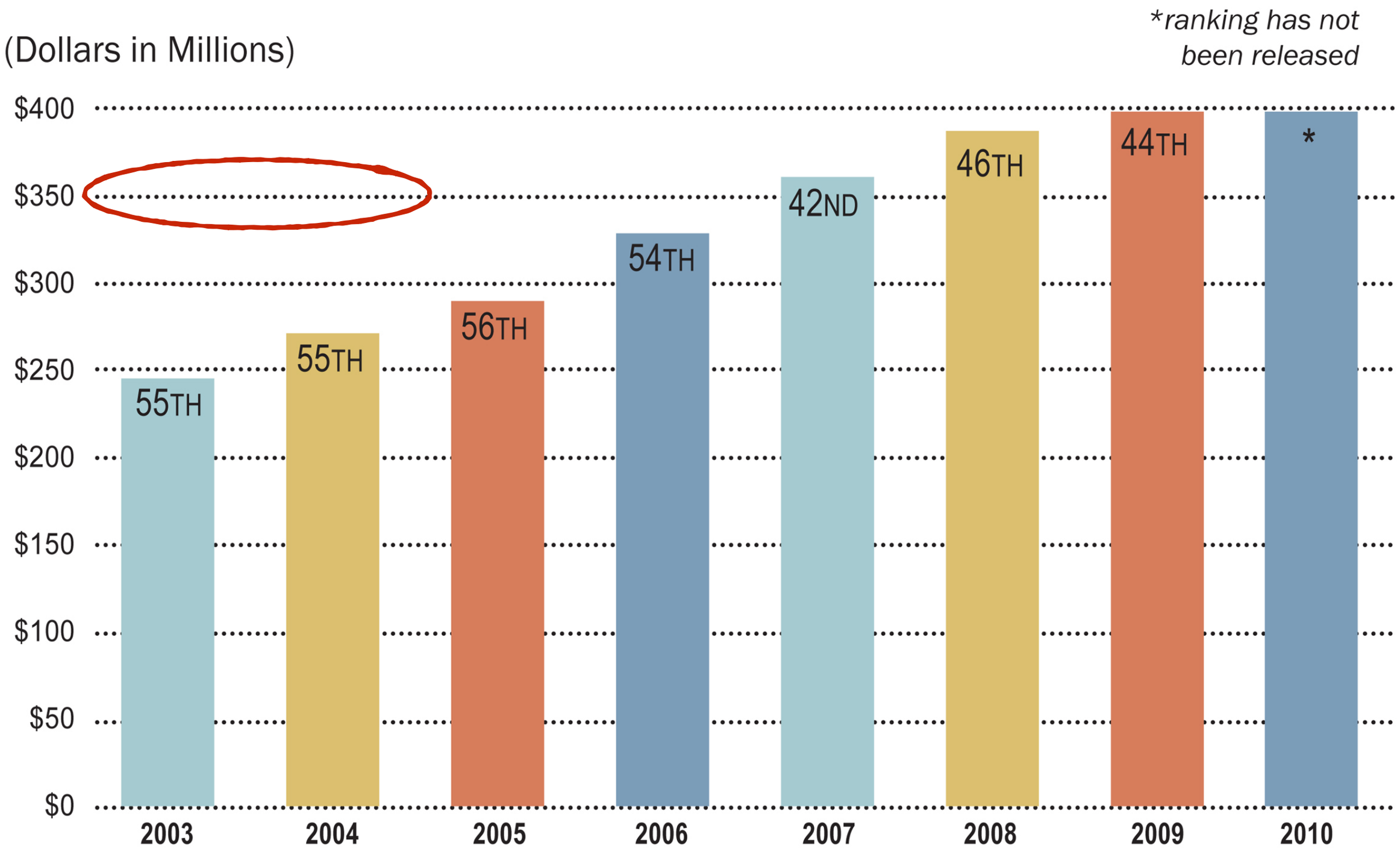
Легенда


(Dollars in Millions)
*ranking has not been released
$400
$350
$300
$250
$200
$150
$100
$50
$0
55TH
55TH
56TH
54TH
42ND
46TH
44TH
*
2003
2004
2005
2006
2007
2008
2009
2010

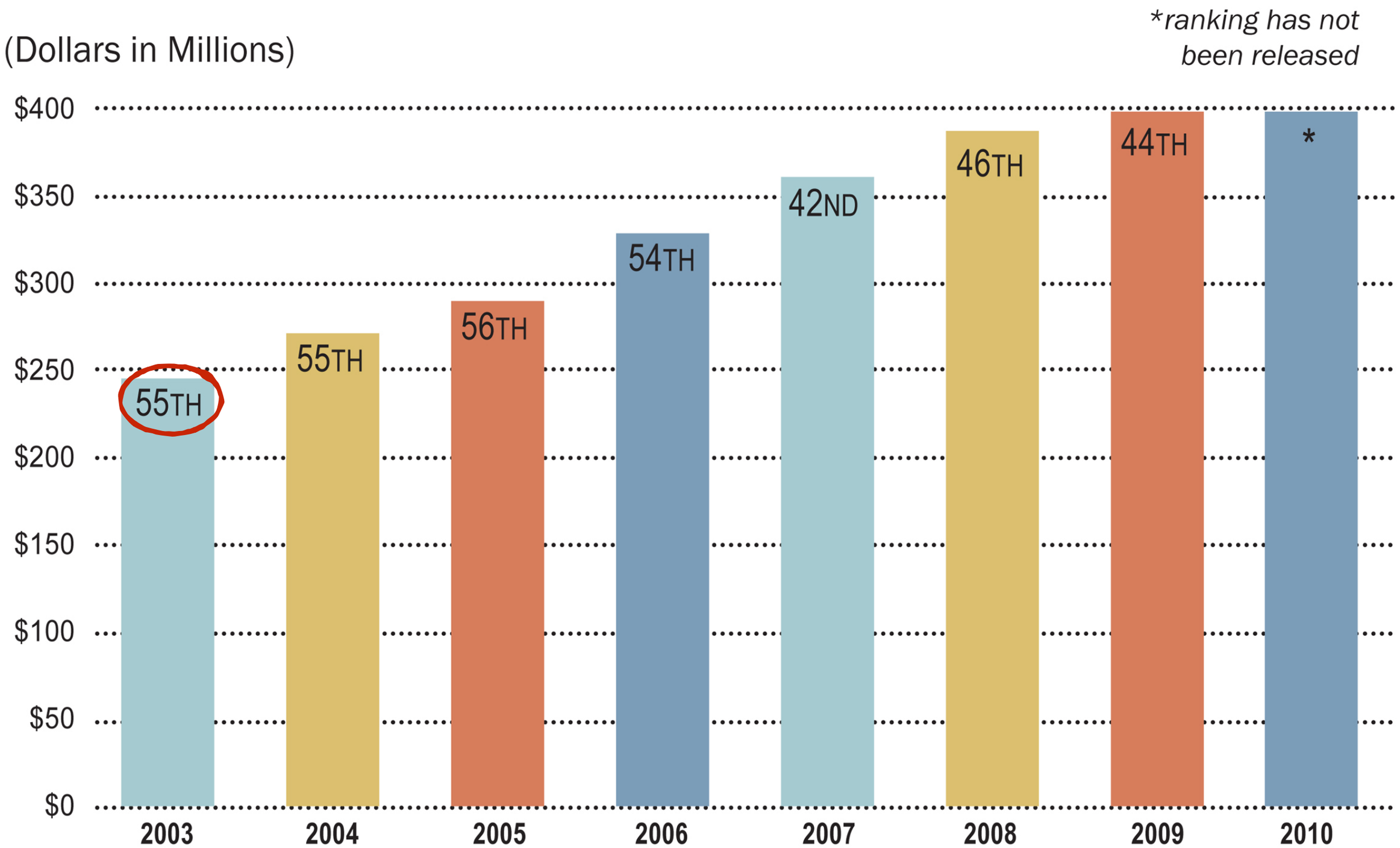
Отметки на шкалах



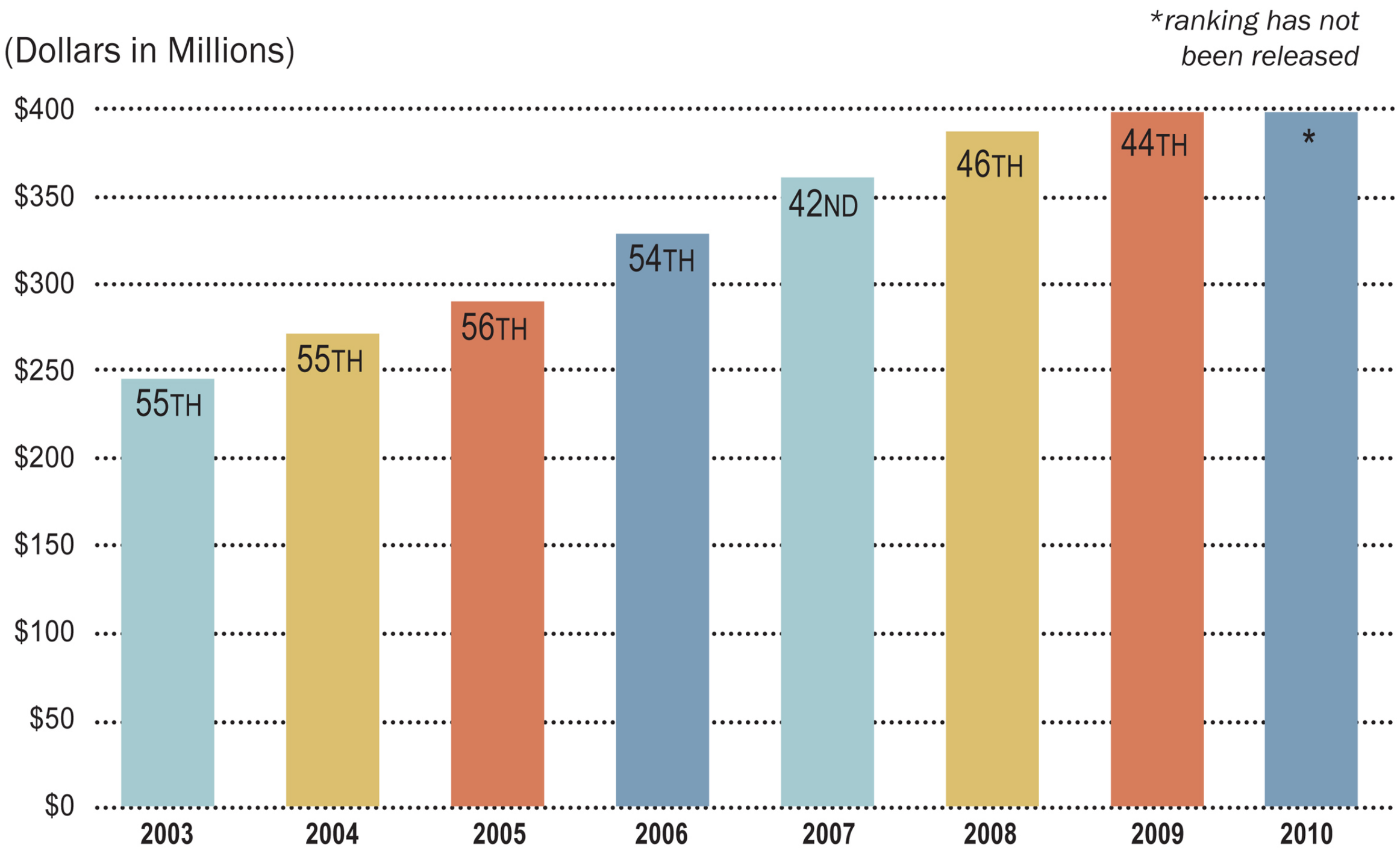
Разметка для удобства

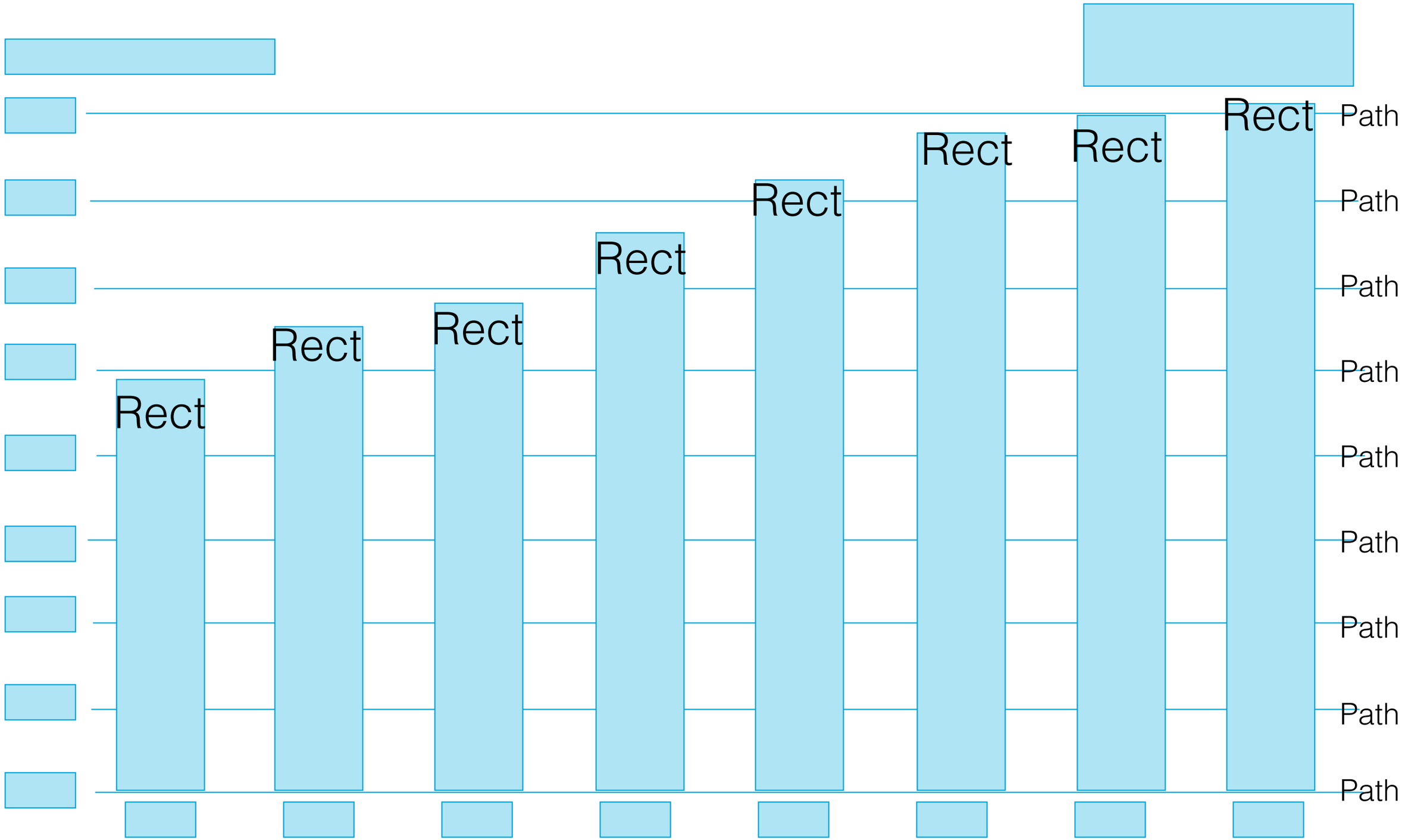


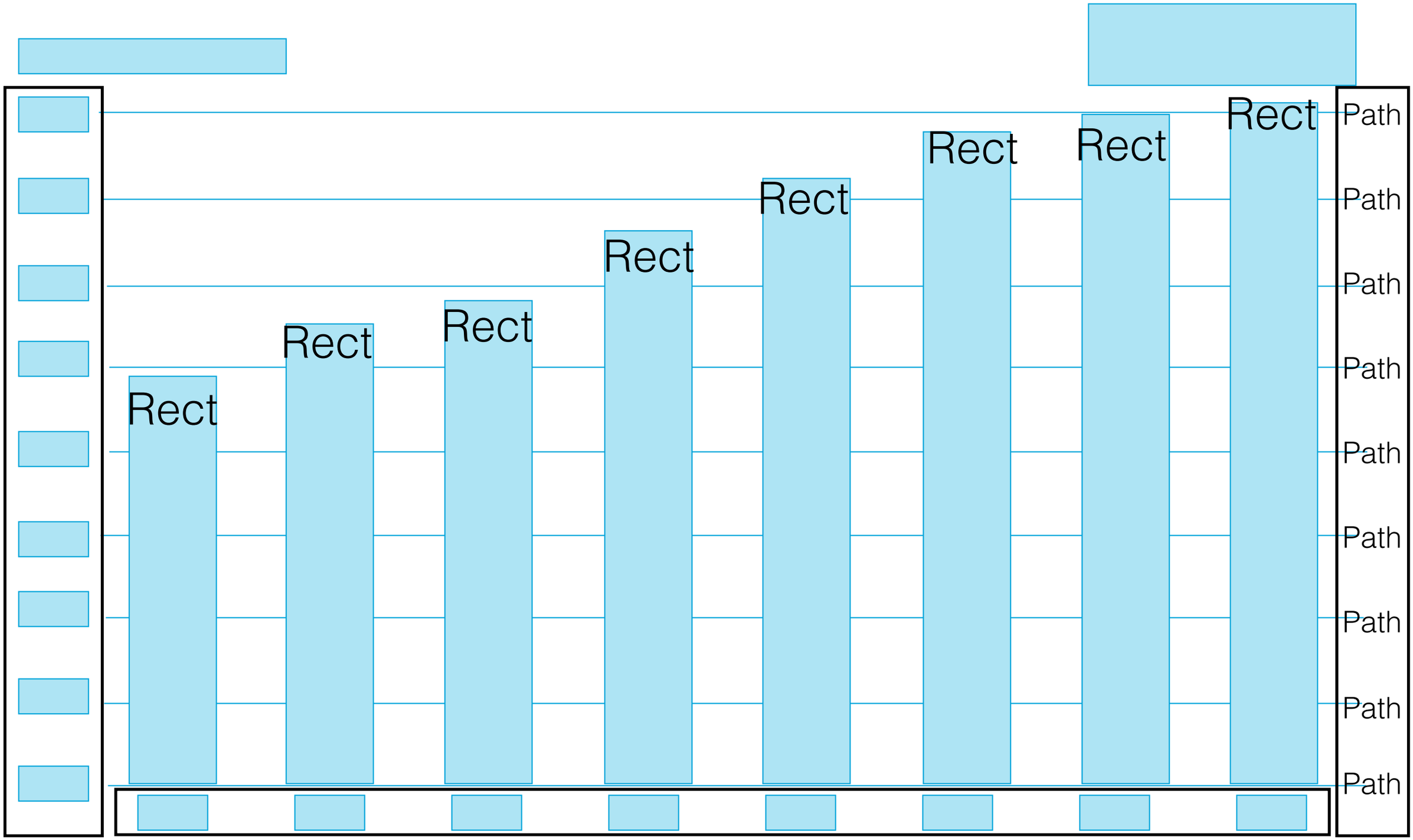
Какие-то еще данные



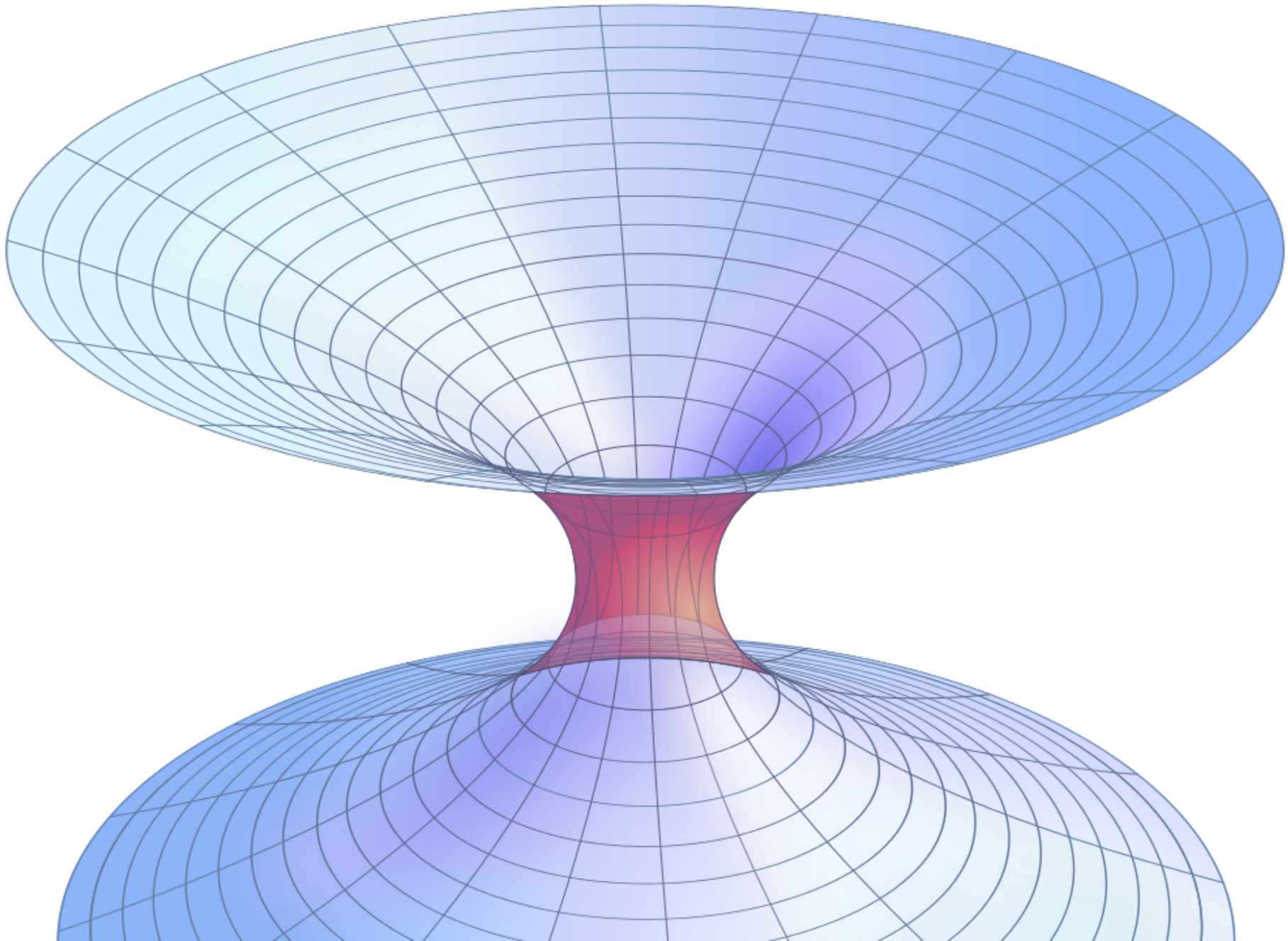
Используем для этого SVG



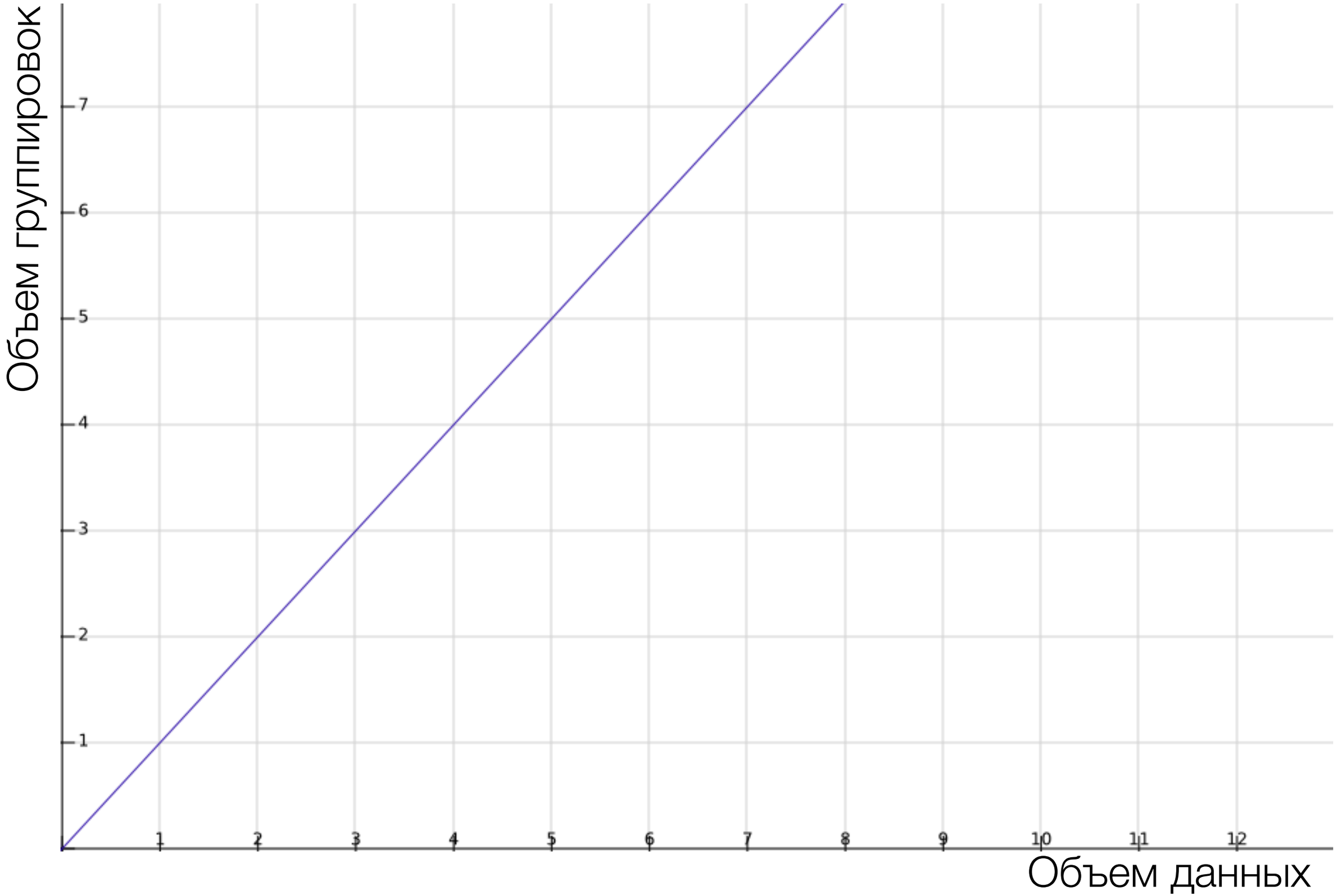

Rect
Rect
Rect
Rect
Rect
Rect
Rect
Rect
Path
Path
Path
Path
Path
Path


Rect
Rect
Rect
Rect
Rect
Rect
Rect
Rect
Path
Path
Path
Path
Path
Path
Path

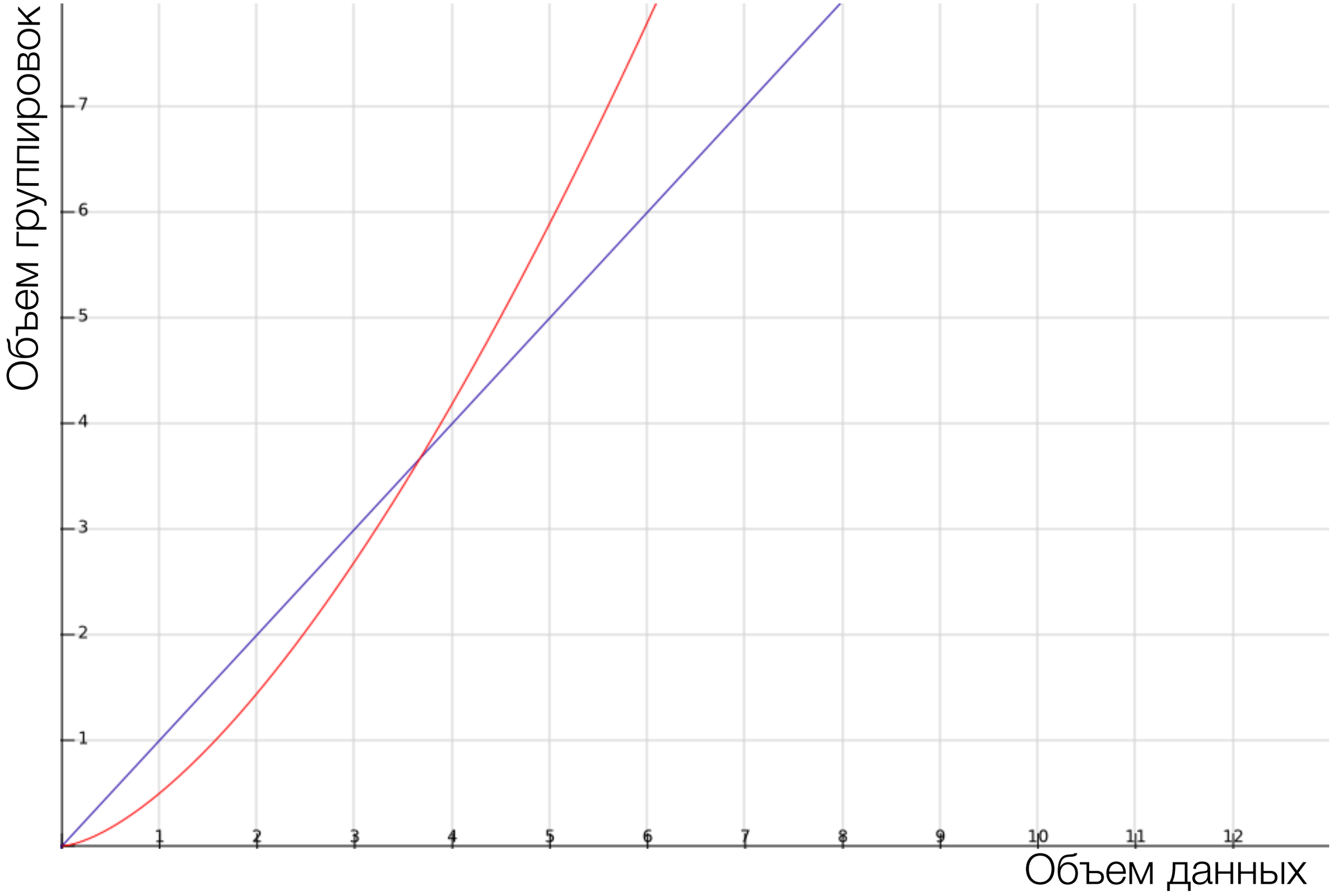
Масштабирование



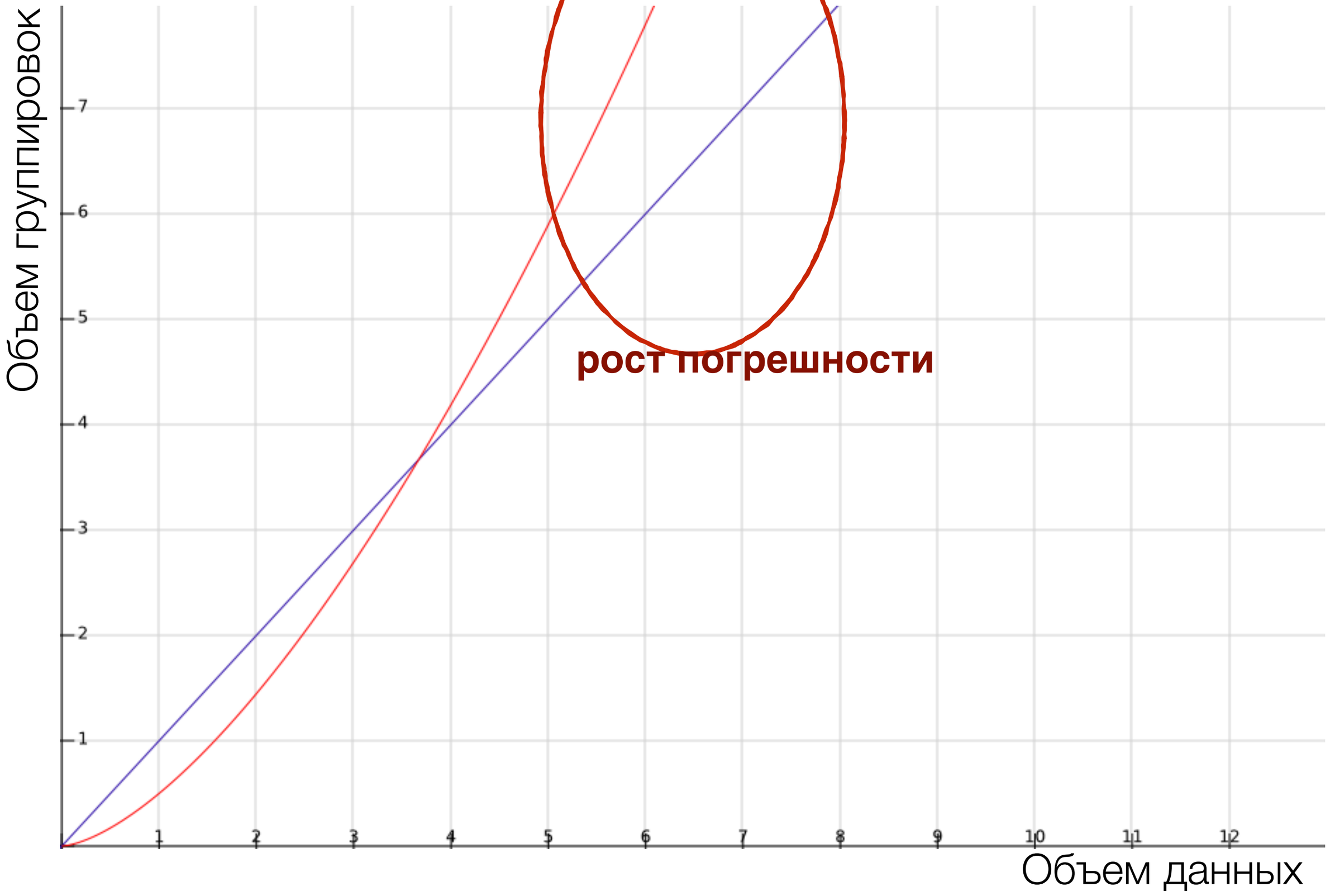
Общий линейный случай



Общий линейный случай

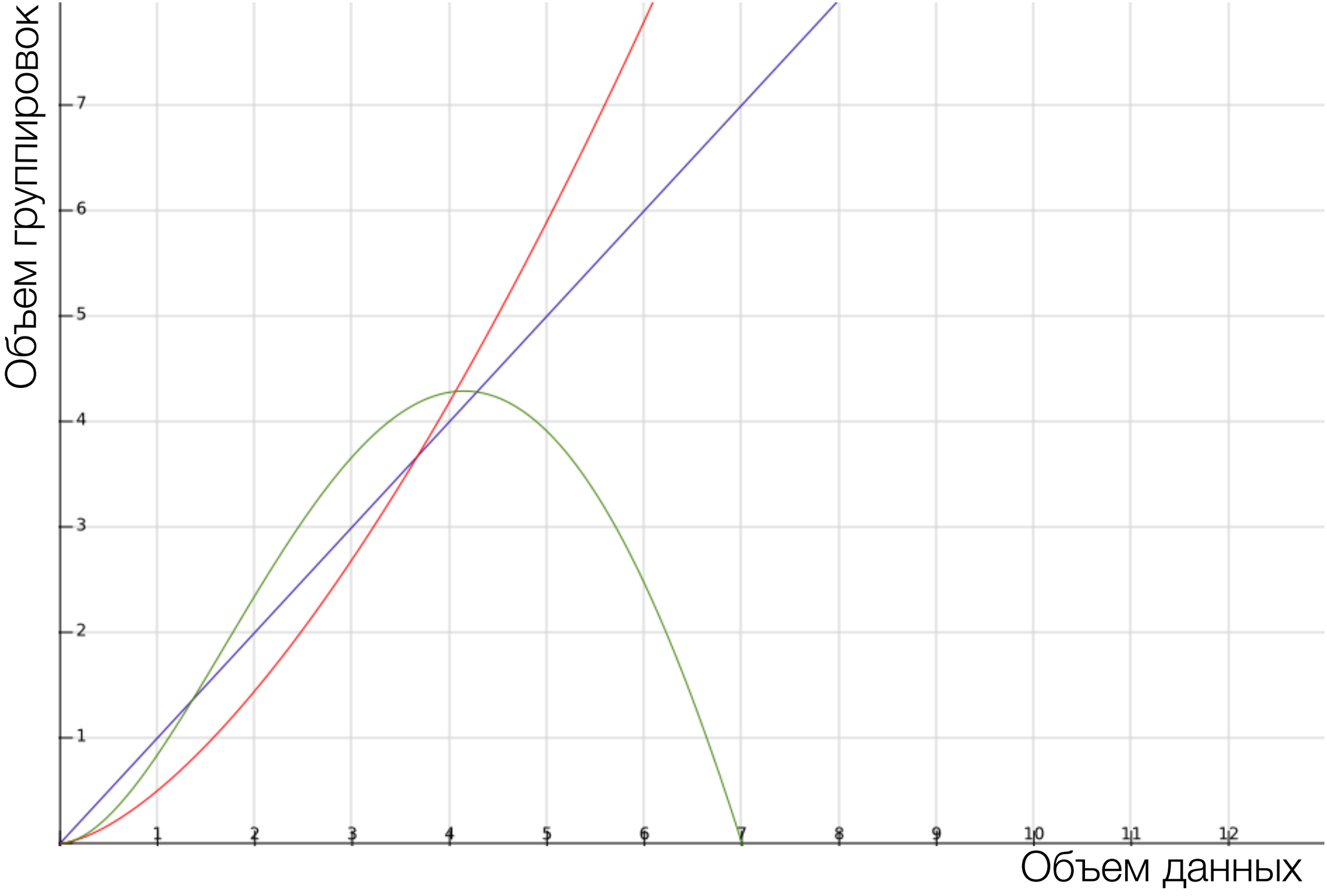


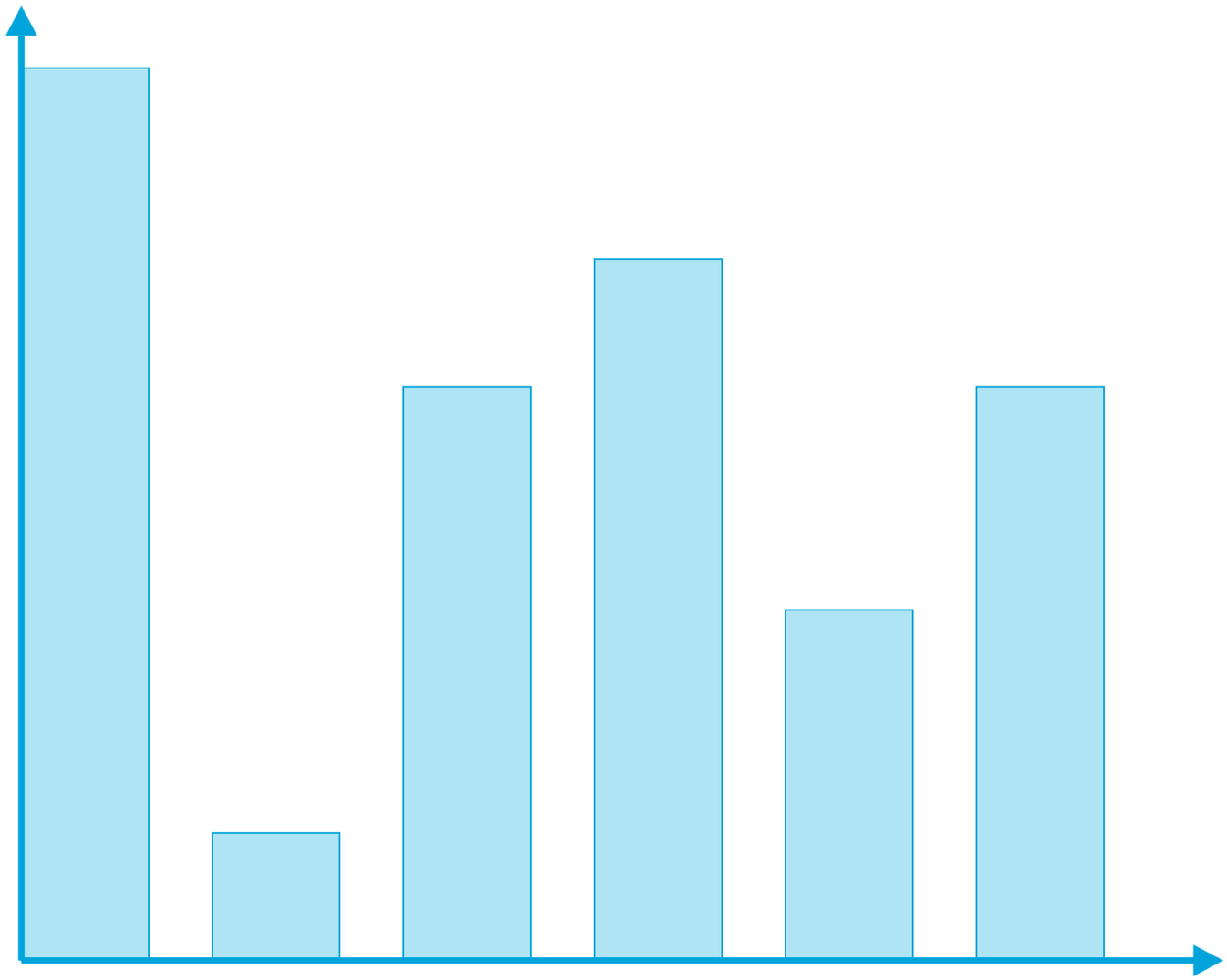
Расхождение данных



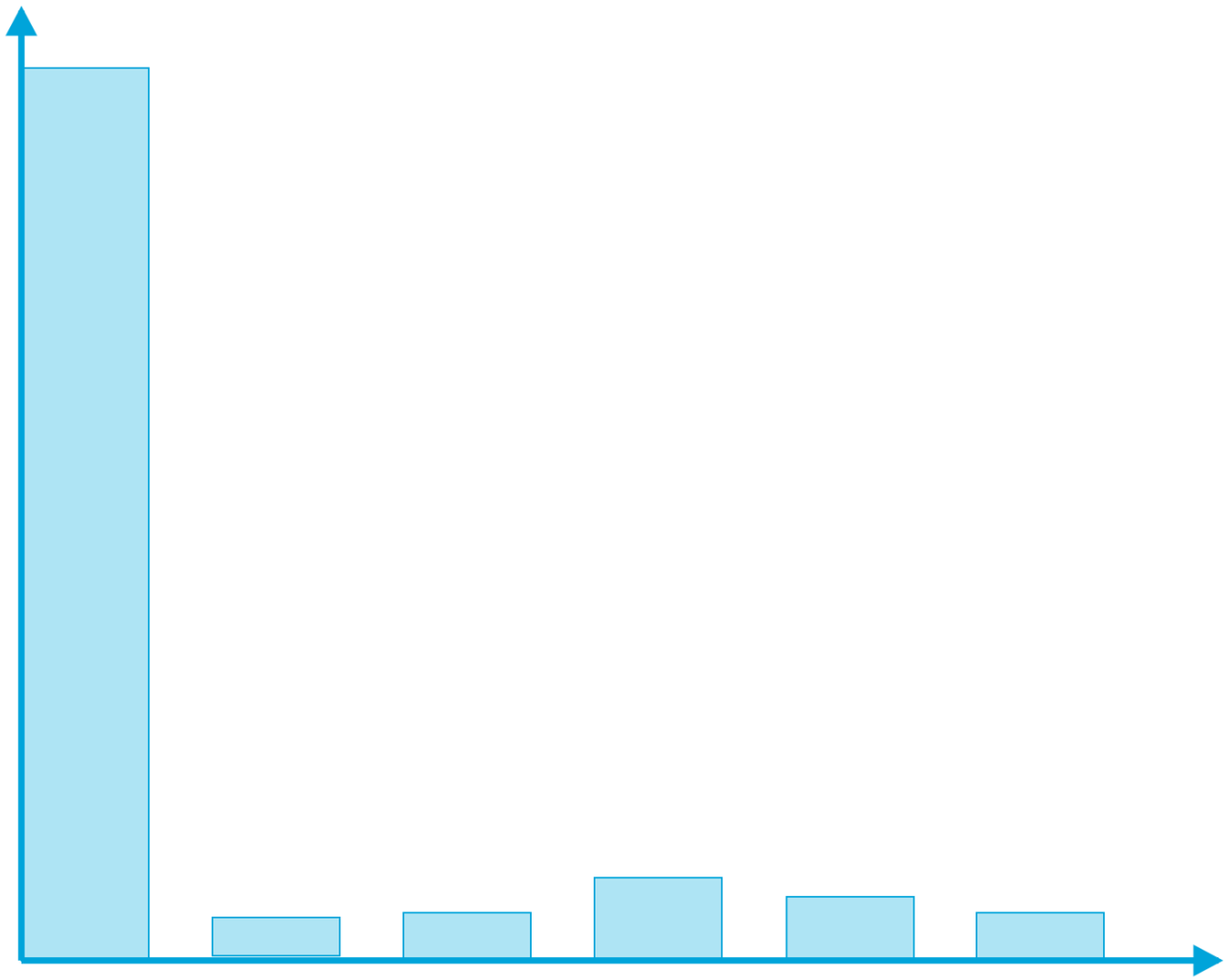
рост погрешности

Расхождение данных

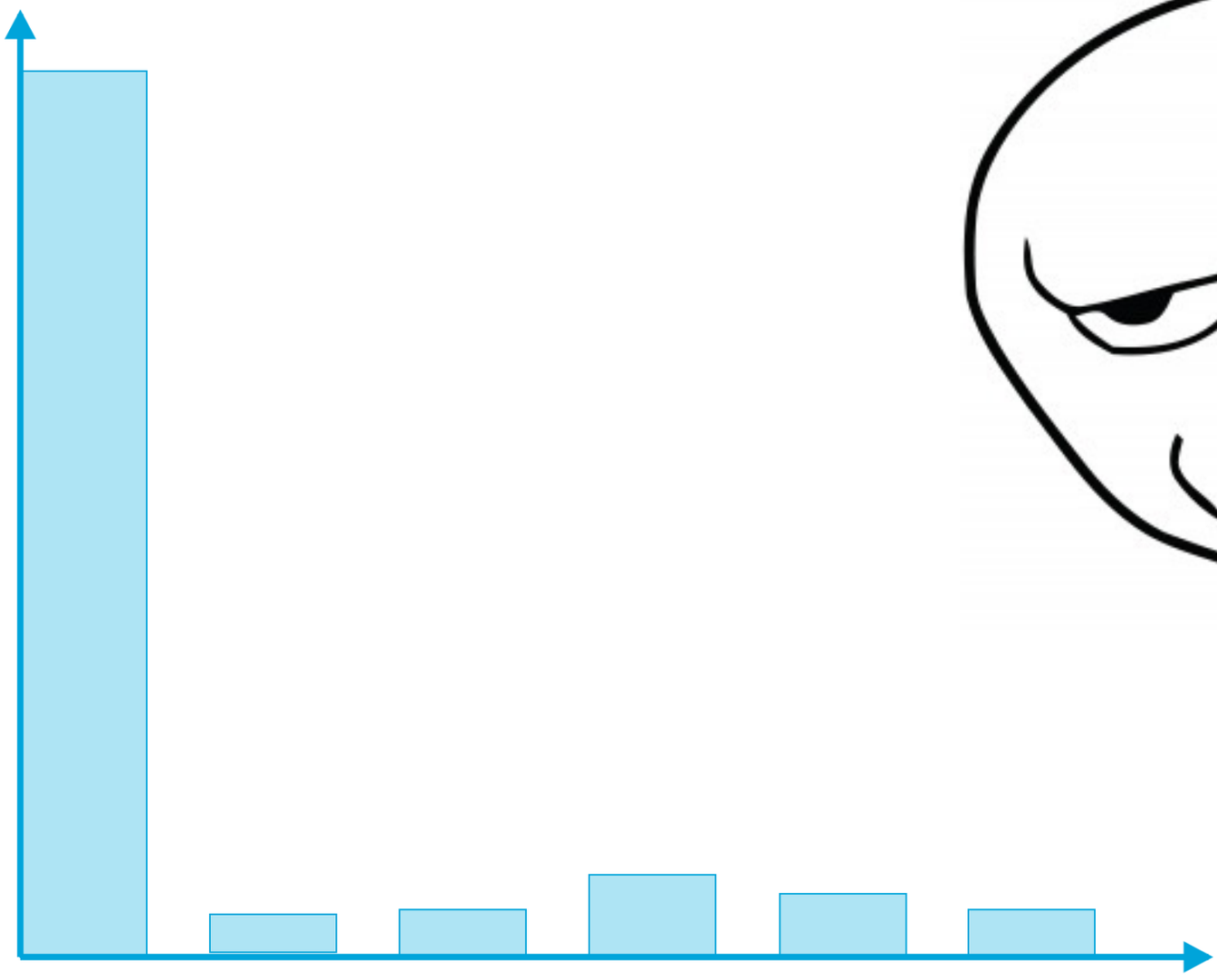




Равномерно распределенные данные



Неравномерно распределенные данные

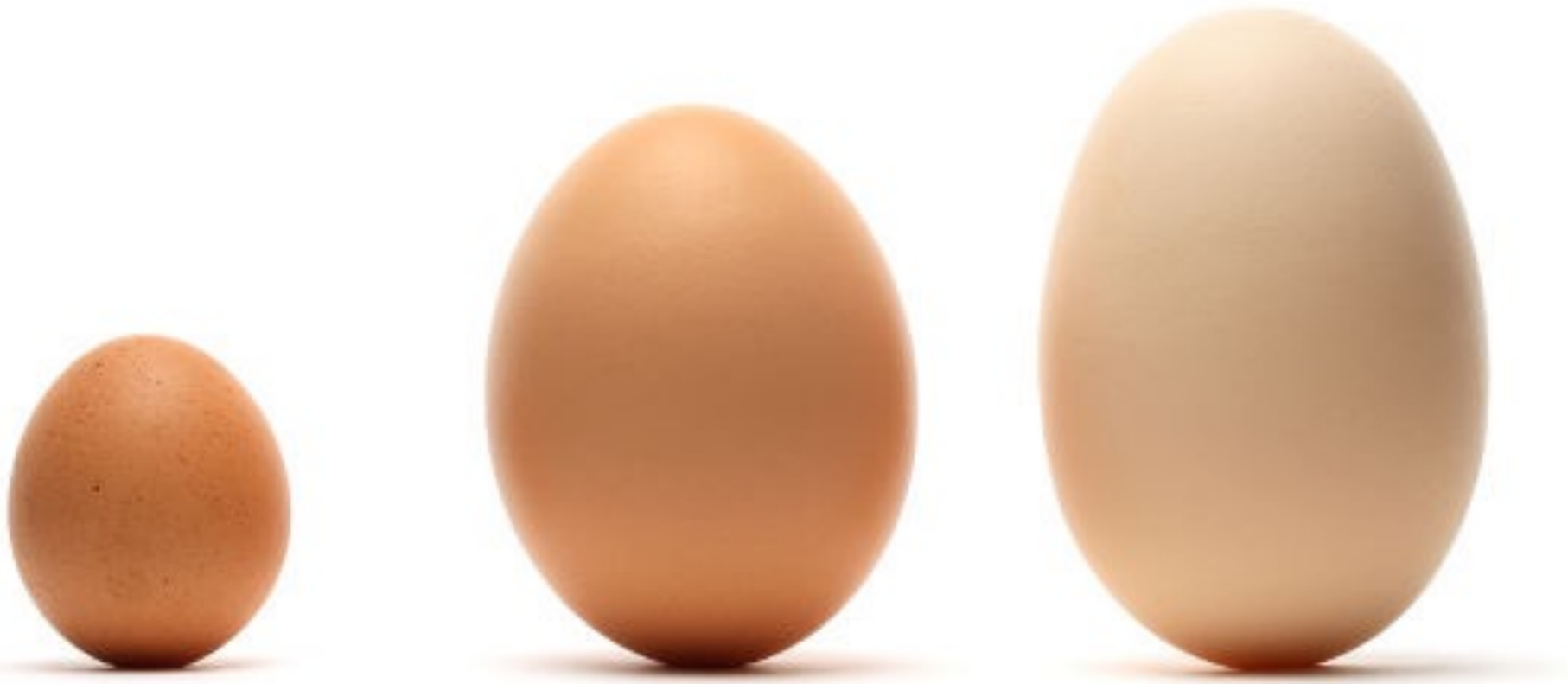


Неравномерно распределенные данные

Проблемы

- Данные бывают неоднородные
- Пространство визуализации динамическое
- Данные могут изменятся
- Слишком дорого обходятся математические вычисления

ШКАЛЫ



Линейные (linear)

```
01: var scale = d3.scale.linear();
02:
03: scale(10); // -> 10
```

Линейные (linear)

```
01: var scale = d3.scale.linear();
02:
03: scale(10); // -> 10
```

масштабирование не задано

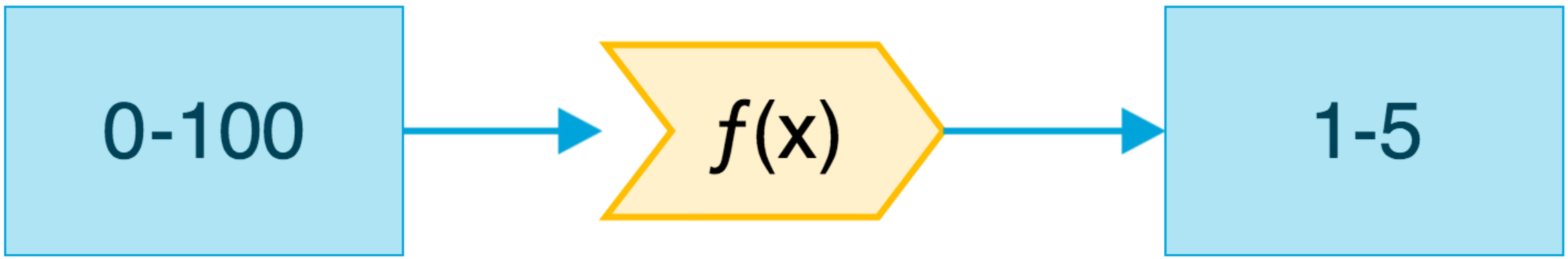
Линейные (linear)

```
01: var scale = d3.scale.linear();
02:
03: scale(10); // -> 10
04:
05: scale.domain([0, 100]);
06:
07:
08: scale.range([1, 5]);
```

Линейные (linear)

```
01: var scale = d3.scale.linear();
02:
03: scale(10); // -> 10
04:
05: scale.domain([0, 100]);
06:     область входных данных
07:
08: scale.range([1, 5]);
      область выходных данных
```

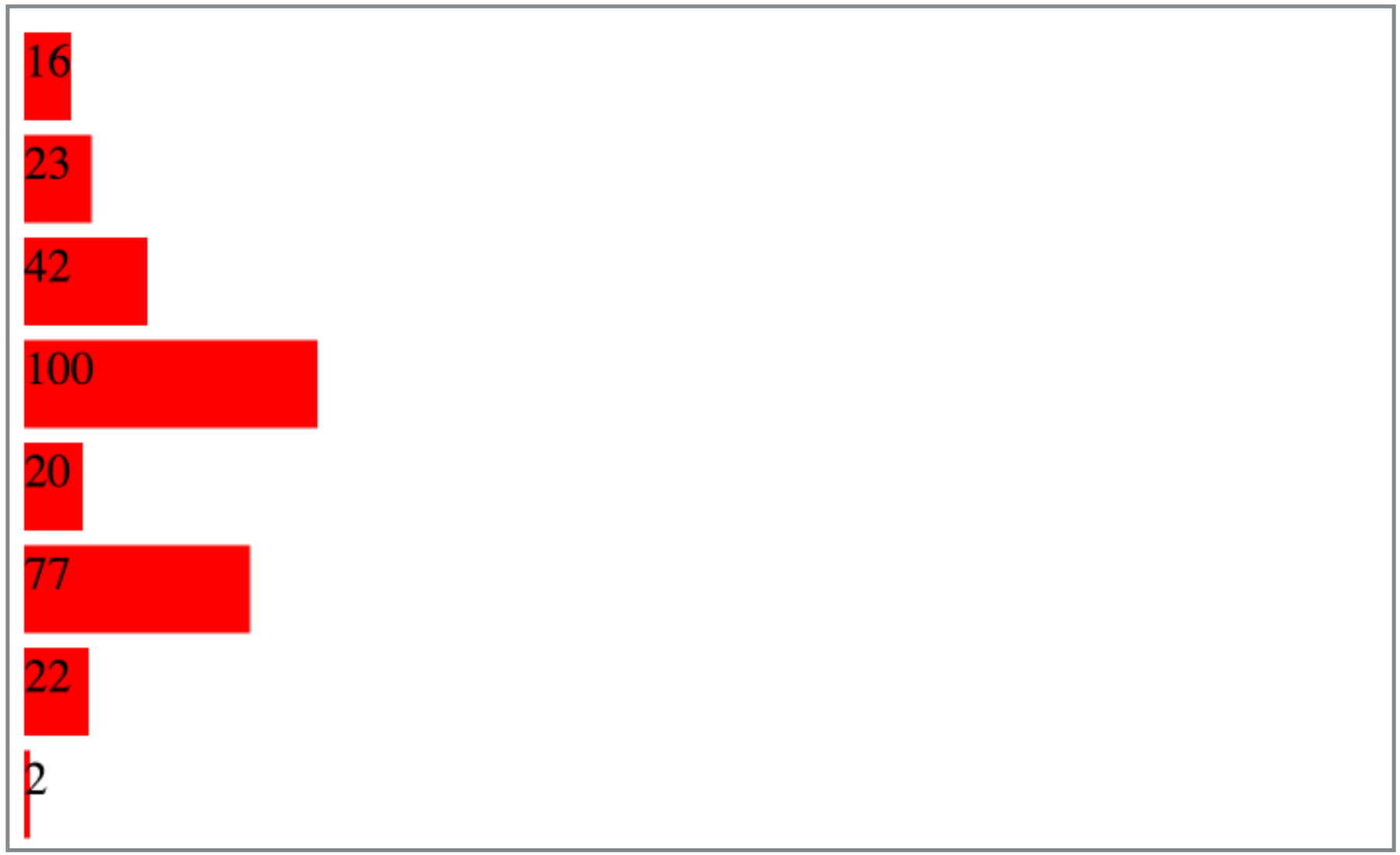
Линейные (linear)



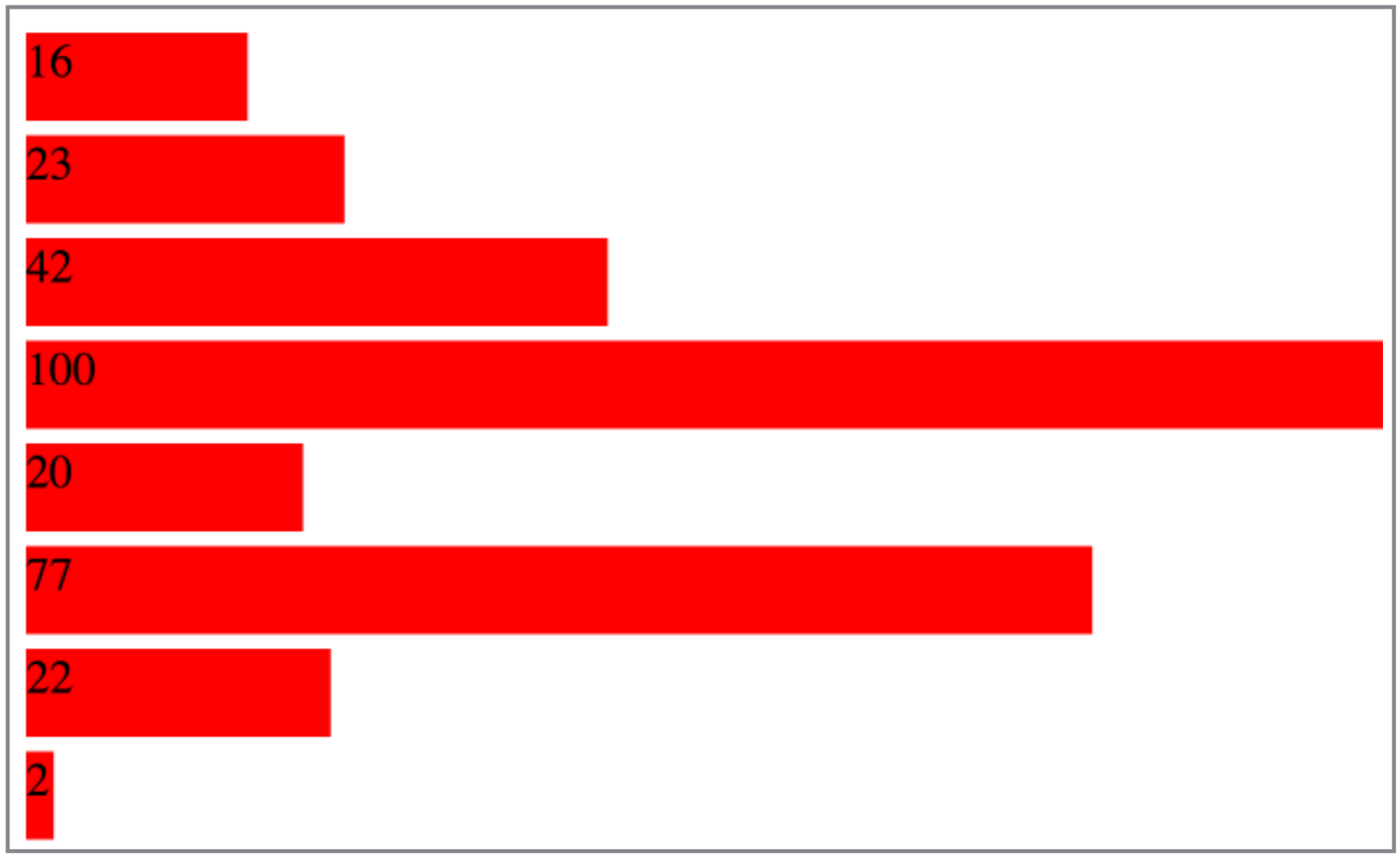
Линейные (linear)

```
01: var scale = d3.scale.linear();
02:
03: scale(10); // -> 10
04:
05: scale.domain([0, 100]);
06:
07: scale.range([1, 5]);
08:
09: scale(0); //-> 1
10: scale(100); //-> 5
11: scale(37); //-> 2.48
12: scale(64); //-> 3.56
13: scale(73); //-> 3.92
```

Было



Стало

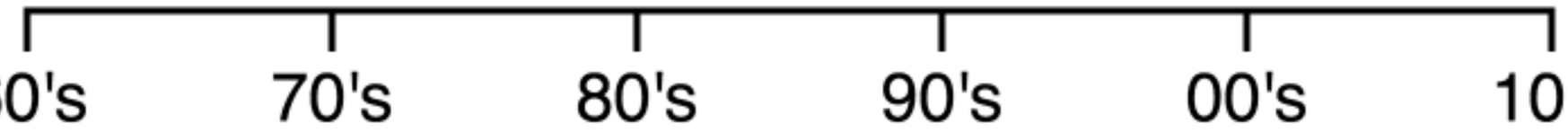


Порядковые (ordinal)

```
01: var scale = d3.scale.ordinal();
```

Порядковые (ordinal)

```
01: var scale = d3.scale.ordinal();
```



Порядковые (ordinal)

```
01: var scale = d3.scale.ordinal();
02:
03: var names = ['60', '70', '80',
04:                '90', '00' ,'10'].map(function(n){
05:   return n + "'s";
06: });
07:
08: scale.domain(names);
```

Порядковые (ordinal)

```
01: var scale = d3.scale.ordinal();
02:
03: var names = ['60', '70', '80',
04:                '90', '00' ,'10'].map(function(n){
05:   return n + "'s";
06: });
07:
08: scale.domain(names);
09:
10: scale.rangePoints([0, 100]);
```

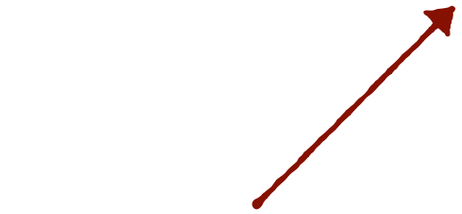
ОСИ

Ось (asix)

```
01: var y = d3.scale.linear()
02:              .range([0, 100])
03:              .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis();
```

Ось (asix)

```
01: var y = d3.scale.linear()
02:              .range([0, 100])
03:              .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis();
```



пустая ось

Ось (asix)

```
01: var y = d3.scale.linear()
02:             .range([0, 100])
03:             .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis()
06:             .scale(y)
07:             .orient('left')
08:             .ticks(10, .01);
```

Ось (asix)

```
01: var y = d3.scale.linear()
02:             .range([0, 100])
03:             .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis()
06:             .scale(y)  шкала оси
07:             .orient('left')
08:             .ticks(10, .01);
```

Ось (asix)

```
01: var y = d3.scale.linear()
02:             .range([0, 100])
03:             .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis()
06:             .scale(y)
07:             .orient('left')
08:             .ticks(10, .01);
```

шкала оси

ориентация на графике

Ось (asix)

```
01: var y = d3.scale.linear()
02:             .range([0, 100])
03:             .domain([1, 5]);
04:
05: var yAxis = d3.svg.axis()
06:             .scale(y)
07:             .orient('left')
08:             .ticks(10, .01);
```

шкала оси

ориентация на графике

размер шага

Ось (asix)

```
01: var yAxis = d3.svg.axis()
02:               .scale(y)
03:               .orient('left')
04:               .ticks(10, .01);
05:
06: var axisElement = d3.select('svg')
07:       .attr('height', 500)
08:       .attr('width', 50)
09:     .append('g')
10:       .attr('transform', 'translate(30, 50)')
11:       .call(yAxis)
```


1
1.5
2
2.5
3
3.5
4
4.5
5

Ось (asix)

```
01: var yAxis = d3.svg.axis()
02:               .scale(y)
03:               .orient('left')
04:               .ticks(10, .01);
05:
06: var axisElement = d3.select('svg')
07:       .attr('height', 500)
08:       .attr('width', 50)
09:     .append('g')
10:       .attr('transform', 'translate(30, 50)’)
11:       .attr('class', 'yAxis')
12:       .call(yAxis)
```

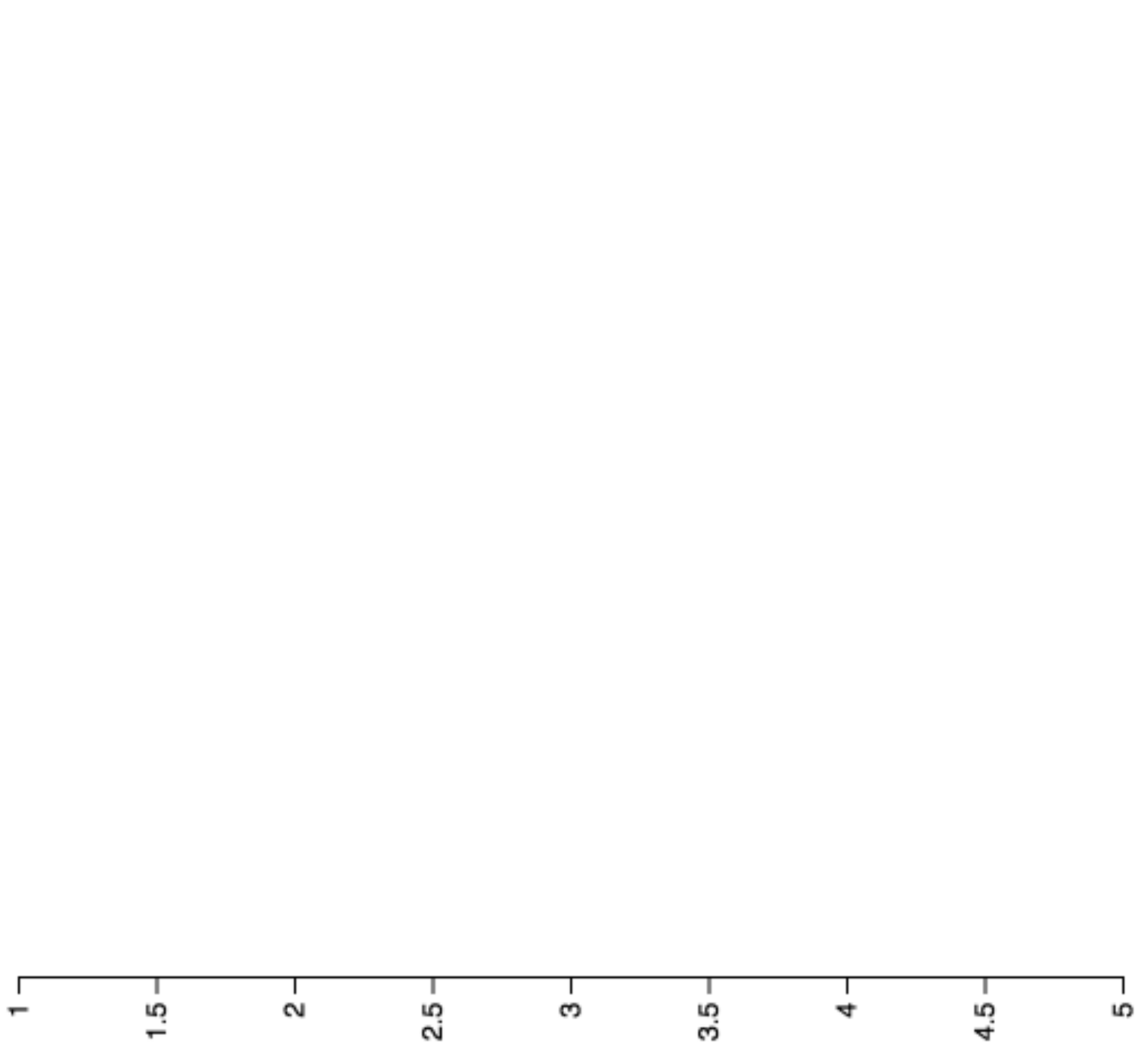
Ось (asix)

```
01: var yAxis = d3.svg.axis()
02:               .scale(y)
03:               .orient('left')
04:               .ticks(10, .01);
05:
06: var axisElement = d3.select('svg')
07:       .attr('height', 500)
08:       .attr('width', 50)
09:     .append('g')
10:       .attr('transform', 'translate(30, 50)’)
11:       .attr('class', 'yAxis')
12:       .call(yAxis)
```

можем использовать CSS

Стилизация осей

```
.yAxis path,
.yAxis line {
    fill: none;
    stroke: #000;
    shape-rendering: geometricPrecision;
}
```


1
1.5
2
2.5
3
3.5
4
4.5
5

Добавление легенды

```
01: axisElement.append('text')
02:       .attr('transform', 'rotate(-90)')
03:       .attr('y', 5)
04:       .attr(‘dy’, 6)
05:       .style('text-anchor', 'end')
06:       .text('Legend');
```

Добавление легенды

```
01: axisElement.append('text')
02:       .attr('transform', 'rotate(-90)')
03:       .attr('y', 5)
04:       .attr(‘dy’, 6)
05:       .style('text-anchor', 'end')
06:       .text('Legend');
```

Поворачиваем на 90°
(потому что ось вертикальная)

Добавление легенды

```
01: axisElement.append('text')
02:       .attr('transform', 'rotate(-90)')
03:       .attr('y', 5)
04:       .attr('dy', 6)
05:       .style('text-anchor', 'end')
06:       .text('Legend');
```

Задаем координаты

Добавление легенды

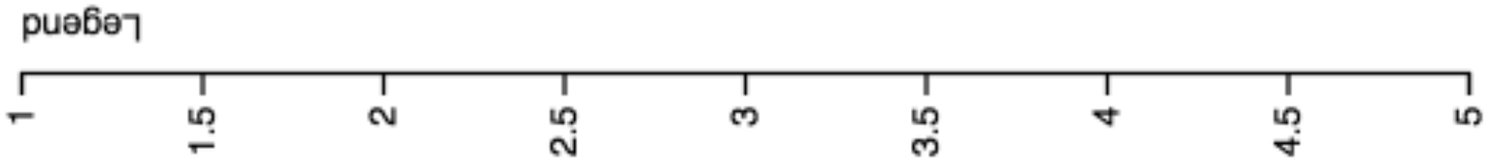
```
01: axisElement.append('text')
02:       .attr('transform', 'rotate(-90)')
03:       .attr('y', 5)
04:       .attr('dy', 6)
05:       .style('text-anchor', 'end')
06:       .text('Legend');
```

Позиционирует точку начала текста

Добавление легенды

```
01: axisElement.append('text')
02:       .attr('transform', 'rotate(-90)')
03:       .attr('y', 5)
04:       .attr('dy', 6)
05:       .style('text-anchor', 'end')
06:       .text('Legend');
```

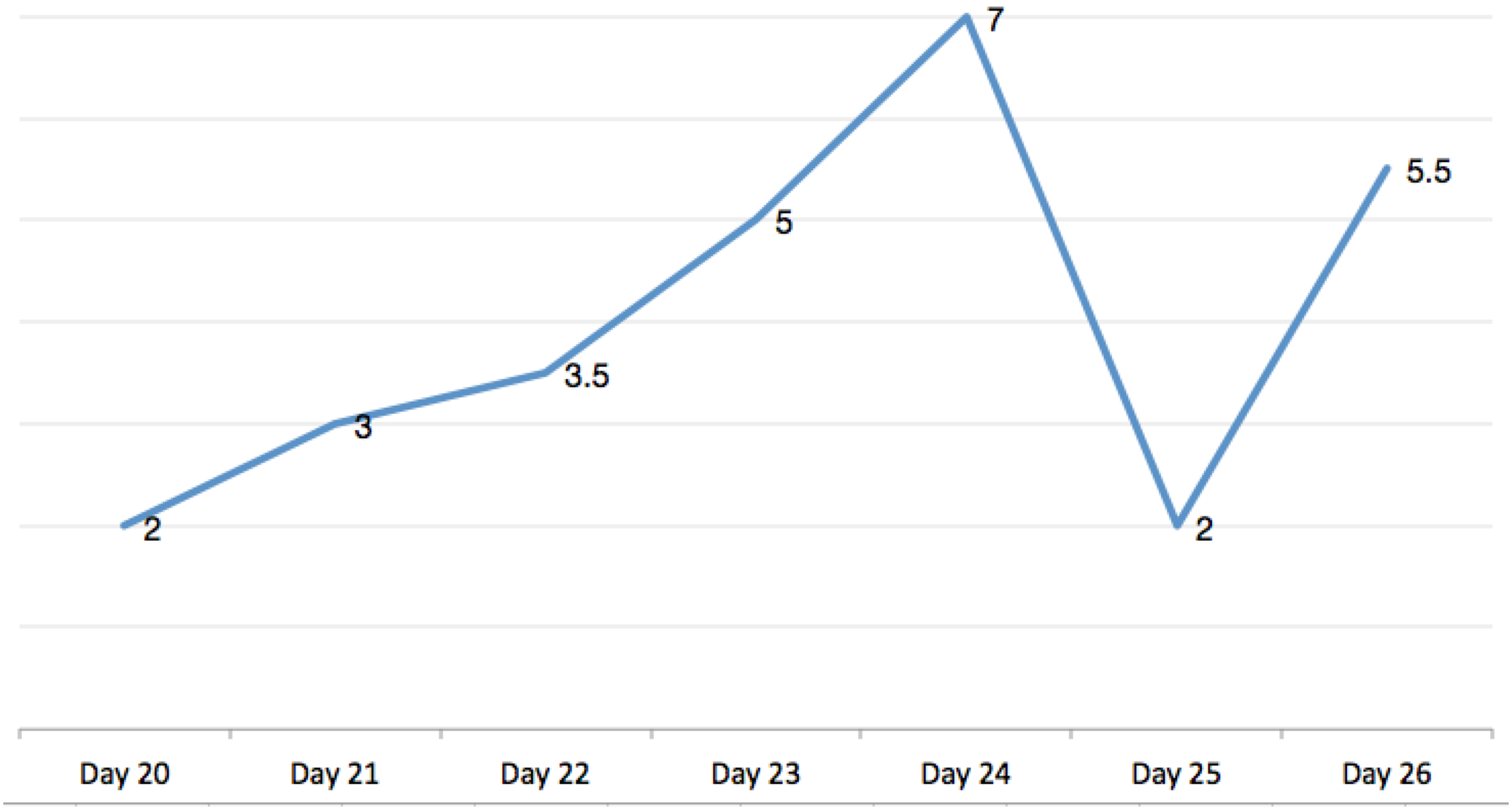
Задаем контент


Legend
1
1.5
2
2.5
3
3.5
4
4.5
5

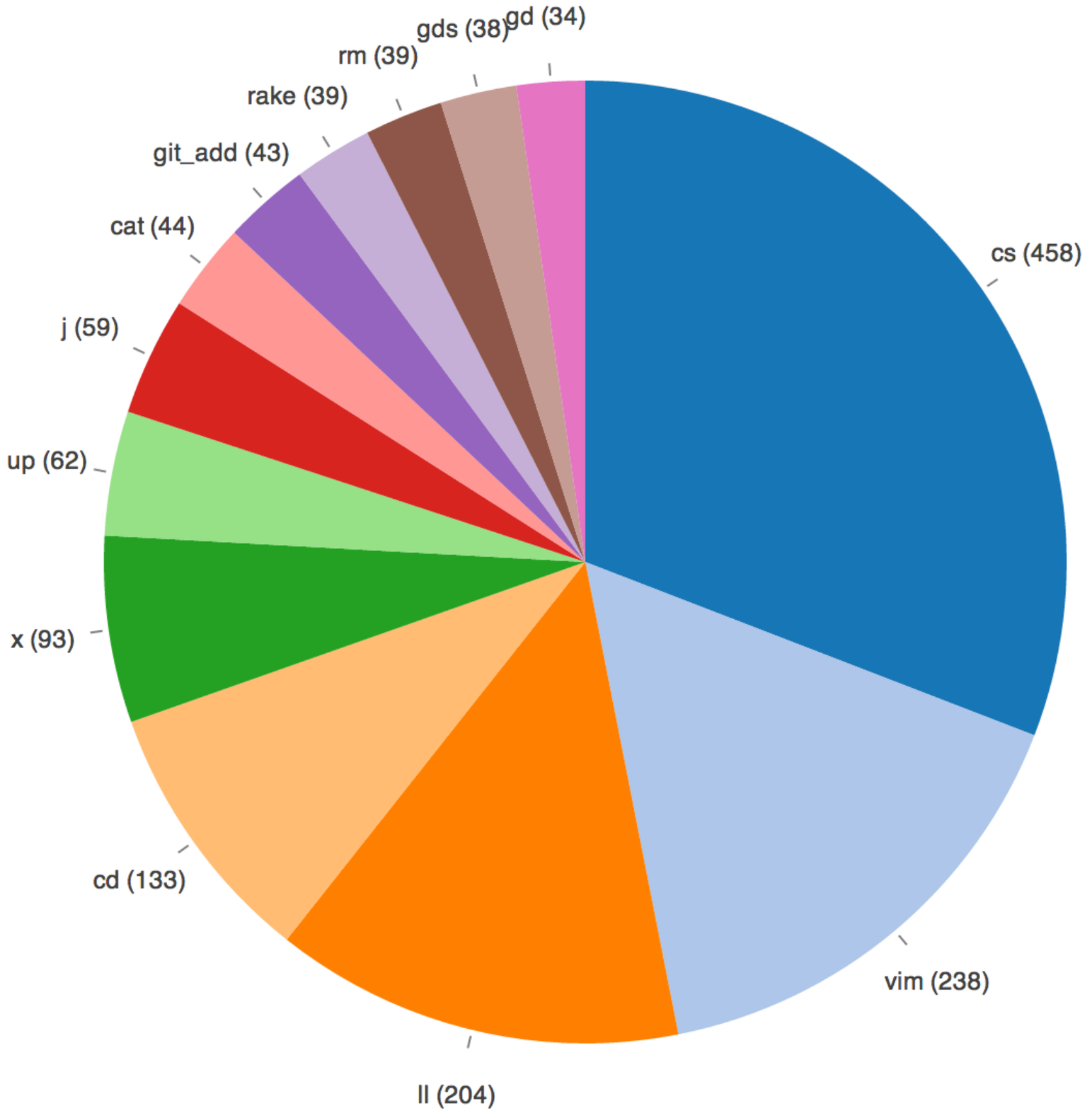
SVG Paths

SVG Paths

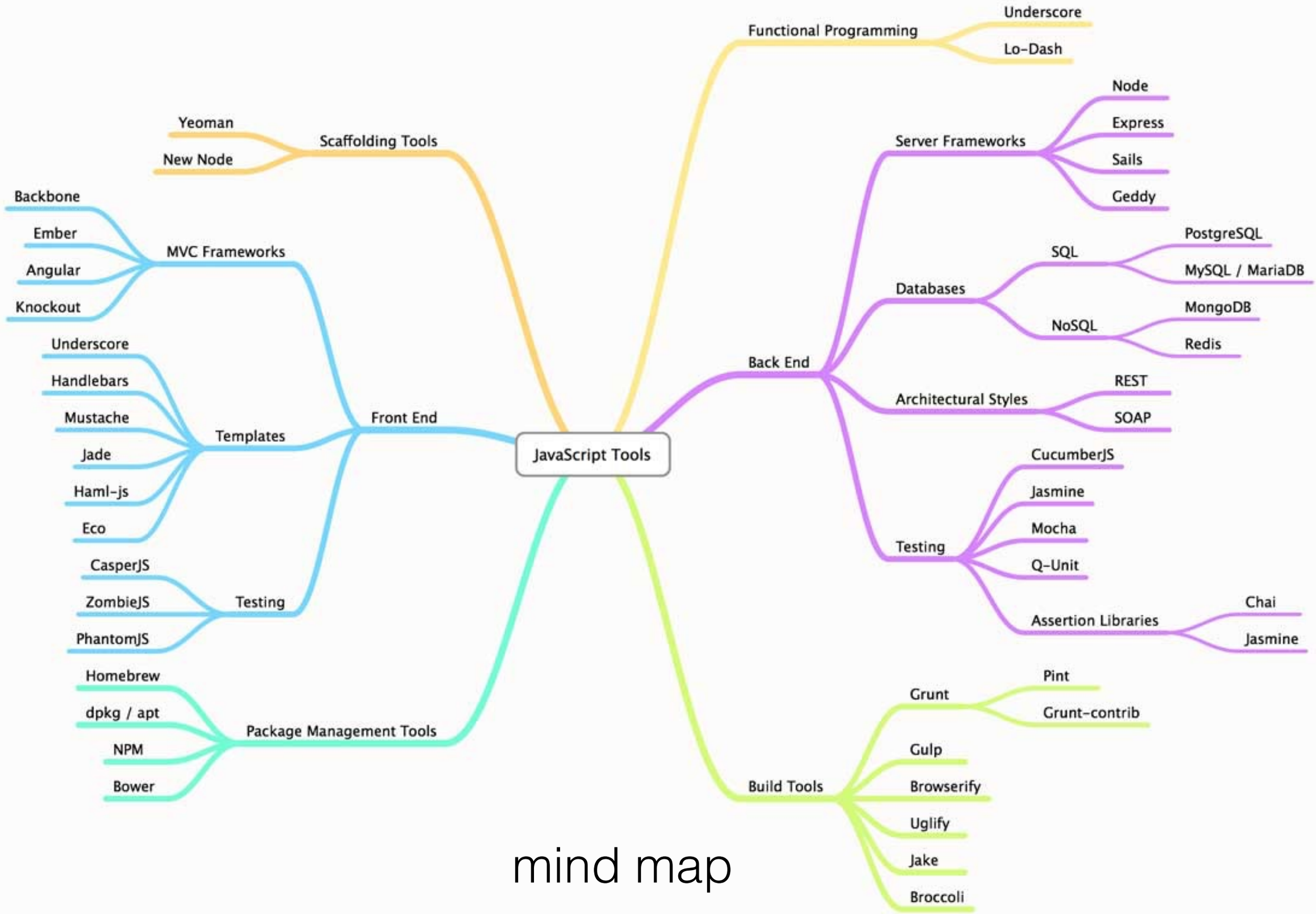
```
<svg width="259px" height="164px">
    <path
        d="M2.07421875,137.054688
C2.07421875,137.054688 77.3203125,-20.0976563
145.15625,5.08984375 C212.992188,30.2773438
-61.484375,146.007812 130.882812,161.074219
C323.25,176.140625 248.136236,12.0254488
213.519531,5.5703125"
        stroke="#000000"
        stroke-width="3"
        fill="none"/>
</svg>
```



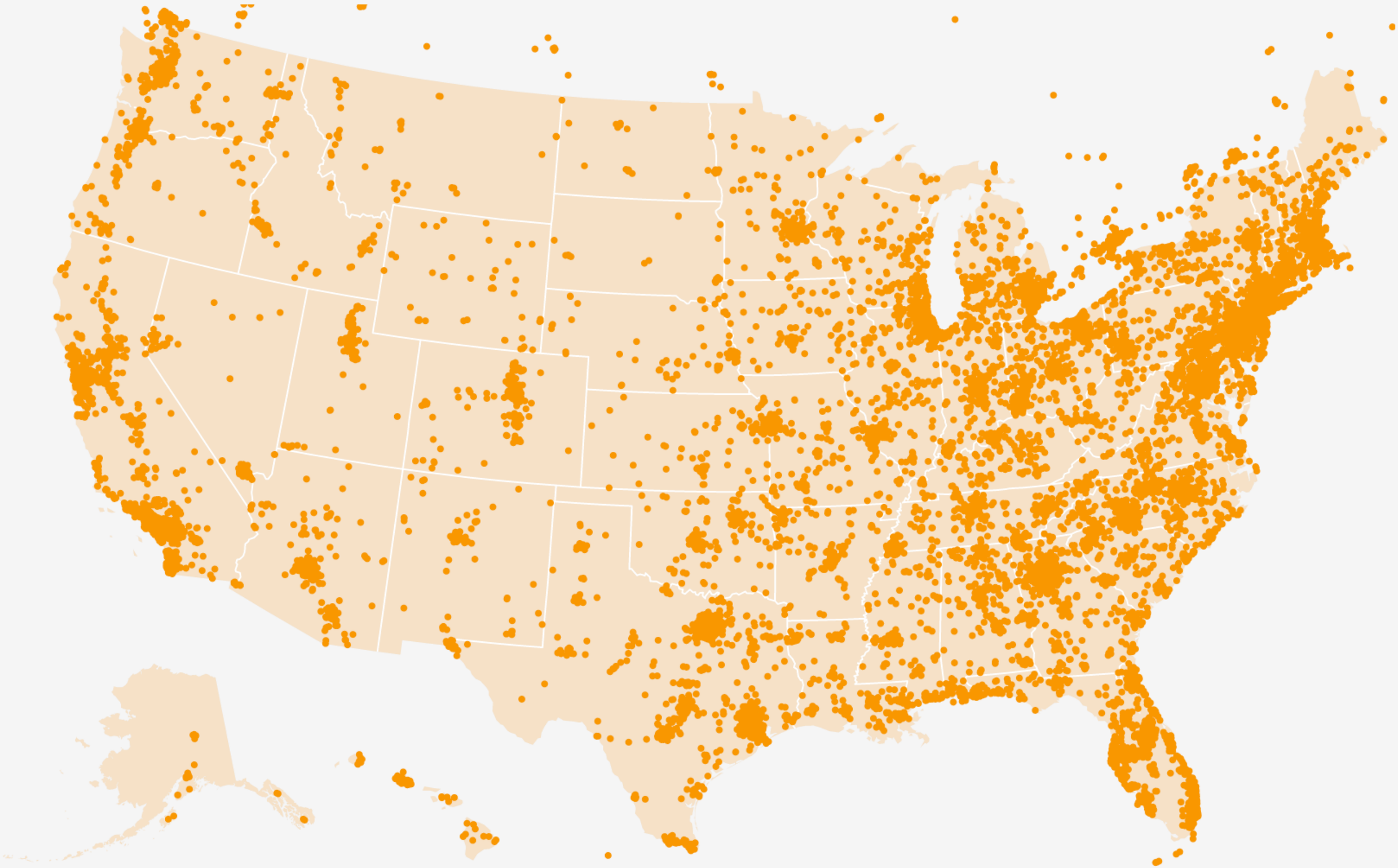
line chart


gd (34)
gds (38)
rm (39)
rake (39)
git_add (43)
cat (44)
j (59)
up (62)
x (93)
cd (133)
ll (204)
vim (238)
cs (458)

pie chart


JavaScript Tools
Front End
Scaffolding Tools
Yeoman
New Node
MVC Frameworks
Backbone
Ember
Angular
Knockout
Templates
Underscore
Handlebars
Mustache
Jade
Haml-js
Eco
Testing
CasperJS
ZombieJS
PhantomJS
Package Management Tools
Homebrew
dpkg / apt
NPM
Bower
Functional Programming
Underscore
Lo-Dash
Back End
Server Frameworks
Node
Express
Sails
Geddy
Databases
SQL
PostgreSQL
MySQL / MariaDB
NoSQL
MongoDB
Redis
Architectural Styles
REST
SOAP
Testing
CucumberJS
Jasmine
Mocha
Q-Unit
Assertion Libraries
Chai
Jasmine
Build Tools
Grunt
Pint
Grunt-contrib
Gulp
Browserify
Uglify
Jake
Broccoli

mind map



geo map

СОБЫТИЯ И ИНТЕРАКТИВНОСТЬ

Подписка

```
01: var button = d3.select('button.my-button');
02:
03: button.on('click', function (data, index) {
04:   console.log('Hey! You clicked at button!');
05: });
```

Подписка

```
01: var button = d3.select('button.my-button');
02:
03: button.on('click', function (data, index) {
04:   console.log('Hey! You clicked at button!');
05: });
```

возвращает тот же самый элемент

Подписка

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         console.log('Hey! You clicked at button!');
06:     })
07:     .on('mouseenter', function (data, index) {
08:         console.log('Hey! You hover at button!');
09:     });
10:
```

Контекст обработчиков

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         console.log(this);
06:     });
```

Контекст обработчиков

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         console.log(this);
06:     });
```

Контекст обработчиков

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         console.log(this);
06:     });
```

```
> button.on('click', function(data, index){
      console.log(this);
  })
<- [▶ Array[1]]
    <button></button>                                VM2017:3
>
```

Контекст обработчиков

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         d3.select(this)
06:             .attr('class', 'clicked')
07:     });
```

Контекст обработчиков

```
01: var button = d3.select('button.my-button');
02:
03: button
04:     .on('click', function (data, index) {
05:         d3.select(this)
06:             .attr('class', 'clicked')
07:     });
```

теперь можно творить чудеса!