УДК 004

Создание гистограммы на javascript библиотеке D3.js

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассмотрен процесс работы с библиотекой D3.js на примере создания простой гистограммы.
Ключевые слова: JavaScript, D3.js

Creating a histogram on the javascript library D3.js

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
In this article, the process of working with the D3.js library will be discussed in detail with the example of creating a simple histogram.
Keywords: JavaScript, D3.js

В современном мире все строится на статистике и закономерностях, и не нужно быть профессором наук, чтобы самостоятельно прогнозировать и строить гипотезы основываясь на данных, даже не нужно иметь мощный компьютер, чтобы все это просчитать, достаточно иметь браузер, а что делать дальше подскажет эта статья.

Цель исследования – это создание простой гистограммы на языке программирования JavaScript и библиотеке D3.js.

Ранее этим вопросом интересовался А. С. Лыкошин развивал тему «Визуализируем данные на javascript с помощью d3.js» [1] в которой рассказывается про возможности библиотеки d3.js и где её можно применять. М.А.Зотов, А.В.Иванова, А.А.Золотин с темой «Визуализация алгебраических байесовских сетей с помощью javascript библиотеки d3.js» [2], а подробнее про подход к визуализации локальной и глобальной структуры алгебраических байесовских сетей. Представлен обоснованный выбор технологии, используемой для визуализации, а также разработаны алгоритмы позиционирования элементов сети в случае визуализации локальной структуры фрагмента знаний и глобальной структуры сети. А.В.Диков опубликовал статью «Математические алгоритмы на javascript» [3] рассказал, как в языке JavaScript среди прочих объектов выделяется объект Math, который обладает множеством математических свойств и

методов. Таким образом открывается возможность использования языка веб программирования для решения математических задач.

Сначала нужно создать шаблон, в который будут прописаны JavaScript команды. Стандартный шаблон выглядит следующим образом.

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
    <script>
    </script>
  </body>
</html>
```

Между двумя тегами script будет код на JavaScript. Первым делом нужно задать массив данных, в данном случае это будут массив целых чисел.

```
var data = [55, 44, 30, 23, 17, 14, 16, 25, 41, 61, 85,
            101, 95, 105, 114, 150, 180, 210, 125, 100, 71,
            75, 72, 67];
```

Потом нужно определить две переменные, которые определяют ширину каждого столбца и размер интервалов между столбцами.

```
var barWidth = 15, barPadding = 3;
```

Также нужно масштабировать высоту каждого столбца относительно максимального значения из массива данных. Это делается с помощью функции d3.max().

```
var maxValue = d3.max(data);
```

Теперь создаем основной элемент SVG, помещая его внутрь тела документа и назначая ширину и высоту.

```
var graphGroup = d3.select('body')
    .append('svg')
    .attr({ width: 1000, height: 250 })
    .append('g');
```

Нужно рассчитать положение x и y столбцов. Они представлены в виде пикселей, начиная с нижнего левого graphGroup. Нужны две функции для их вычисления. Первая вычисляет местоположение x левой части столбца:

```
function xloc(d, i) { return i * (barWidth + barPadding); }
```

Во время вызова ей будет передано текущее значение и его положение в data массиве.

Поскольку SVG использует начальные координаты с верхнего левого угла, нужно рассчитать расстояние графика от верхней точки до нижней:

```
function yloc(d) { return maxValue - d; }
```

Когда рисуется каждый столбец, можно использовать преобразование translate. Объявив функцию, которая с учетом текущего элемента данных и его позиции в массиве возвращает вычисленную строку для свойства преобразования на основе этих данных и функций:

```
function translator(d, i) {
    return "translate(" + xloc(d, i) + "," + yloc(d) + ")";
   }
```

Все, что нужно сделать, это генерировать визуальные изображения SVG из данных:

```
barGroup.selectAll("rect")
    .data(data)
    .enter()
    .append('rect')
    .attr({
        fill: 'steelblue',
        transform: translator,
        width: barWidth,
        height: function (d) { return d; }
    });
```

Добавление меток в столбцы

Теперь будем добавлять метку, удерживающую значение нулевой точки в верхней части каждого столбца.

Для этого нужно изменить SVG таким образом, чтобы:

1. Каждый столбец представлялся группой элементов SVG вместо одного.

2. Внутри каждой группы, представляющей столбец, добавляем SVG и текстовый элемент.
3. Затем группа отображается и, следовательно, отображает дочерние элементы.
4. Размер параметра rect установлен, как и раньше, в результате чего содержащая группа расширяется до того же размера.
5. Текст помещается относительно верхнего левого угла его содержащей группы.

Сгруппировав эти элементы, можем повторно использовать предыдущий код для отображения и использовать преимущество группы для поиска всех дочерних элементов. Более того, нам нужно знать только размер и положение этих дочерних элементов относительно их собственной группы. Код для этого идентичен предыдущему примеру посредством объявления функций позиционирования.

Первое изменение заключается в создании селектора, который представляет столбцы:

```
var barGroups = g.selectAll('g')
    .data(data)
    .enter()
    .append('g')
    .attr('transform', translator);
```

Вместо создания rect теперь создается элемент группы. Группа изначально пуста, и ей присваивается преобразование, которое перемещает его в соответствующее положение.

Используя селектор, на который ссылается barGroups, теперь код добавляет, rect в каждую группу, а также устанавливает соответствующие атрибуты.

```
barGroups.append('rect')
    .attr({
        fill: 'steelblue',
        width: barWidth,
        height: function(d) { return d; }
    });
```

Следующий шаг - добавить text элемент, чтобы показать значение столбца. Мы собираемся разместить этот текст таким образом, чтобы он находился прямо в верхней части столбца.

Для этого нам нужно преобразование translate. Оно используется для каждого столбца, поэтому мы можем определить переменную:

```
var textTranslator = "translate(" + barWidth / 2 + ",0)";
```

Затем добавляем текстовый элемент в каждую группу, устанавливаем текст, соответствующие атрибуты для текста и стиль шрифта.

```
barGroups.append('text')
    .text(function(d) { return d; })
    .attr({
        fill: 'white',
        'alignment-baseline': 'before-edge',
        'text-anchor': 'middle',
        transform: textTranslator
    })
    .style('font', '10px sans-serif');
```

Результат можно наблюдать на рис.1.

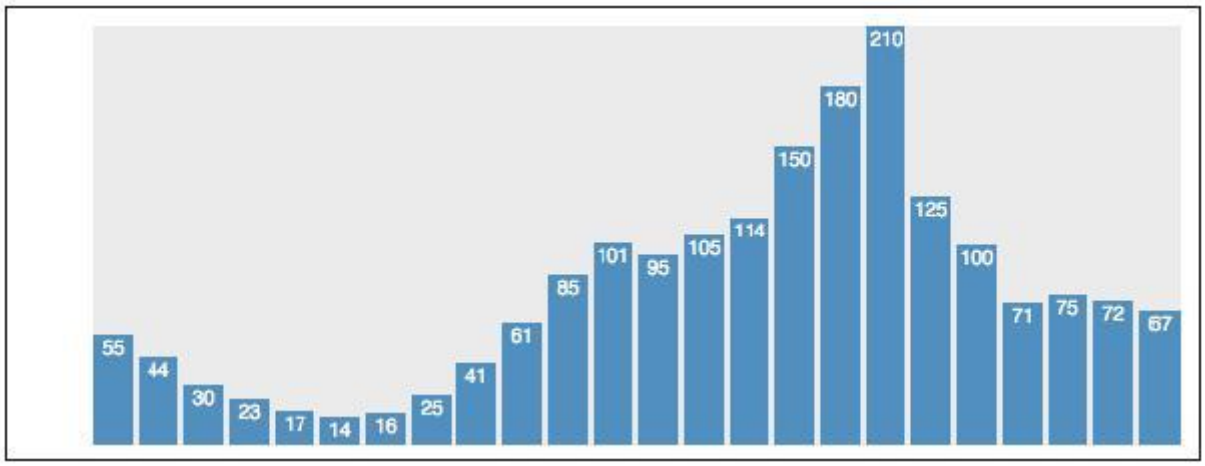

Рис. 1

Вывод

В этой статье показали, как создать простую гистограмму на библиотеке D3.js. Узнали, как размещать и сортировать каждый элемент графика в соответствии с его данными и как позиционировать группы данных, которые содержат несколько визуальных представлений, представляющих отдельный столбец, в частности, как добавить метку, которая представляет значение базового элемента данных.

Библиографический список

1. Лыкошин А. С. Визуализируем данные на javascript с помощью d3.js // Хакер. 2015. С. 94-101. URL: https://elibrary.ru/item.asp?id=27223050 (Дата обращения: 30.12.2017)
2. Зотов М. А., Иванова А. В., Золотин А. А. Визуализация алгебраических байесовских сетей с помощью javascript библиотеки d3.js // Интеллектуальные системы и технологии: современное состояние и перспективы 2017. С. 86-94. URL: https://elibrary.ru/item.asp?id=30320169 (Дата обращения: 03.12.2017)

3. Диков А.В. Математические алгоритмы на javascript // Известия Пензенского государственного педагогического университета им. В.Г. Белинского. 2009. С. 84-88. URL: https://elibrary.ru/item.asp?id=13051117 (Дата обращения: 30.12.2017)