

## Методы обработки и визуализации данных в веб-интерфейсе с помощью библиотеки Dimensional Charting

*И.А. Балеев, А.Н. Земцов, Д.А. Астахов*

*Волгоградский государственный технический университет*

**Аннотация:** В данной работе демонстрируется подход по визуализации данных при помощи манипуляций с json-данными. Опираясь на библиотеку Dimensional Charting, можно проводить рендеринг графической информации с учетом обработки больших данных, так как интерактивная визуализация данных в веб-интерфейсе находит большое применение при анализе пользовательского трафика в интернете. Dimensional Charting является связующим звеном двух встроенных библиотек. Библиотека Crossfilter ответственна за сортировку, фильтрацию больших данных, а d3.js необходим для отрисовки графического материала на html-страницах. Благодаря новым технологиям, можно обрабатывать большие данные, не используя сторонние языки программирования.

**Ключевые слова:** Обработка данных, визуализация данных, фильтрация, сортировка dimensional charting, javascript, json, crossfilter, data-driven documents

Вебсайты, корпоративные интранет и экстранет - порталы, системы электронного документооборота, CRM-веб системы, системы управления базами данных и другие веб-системы за последние 20 лет получили колоссальное развитие. В результате такого роста все больше информации хранится в веб-пространстве, и при увеличении объема такой информации, ценность ее растет [1].

За этот период JavaScript перерос из обычного встраиваемого языка сценариев в мультипарадигменный язык программирования. С появлением новых библиотек, обработка больших данных и отображение результата в виде диаграмм в веб-интерфейсе не составляет труда.

С помощью библиотеки Dimensional Charting (dc.js), можно визуализировать и проводить анализ данных в браузере и на мобильных устройствах. Dc.js включает в себя библиотеку Data-Driven Documents (d3.js), которая используется для визуализации диаграмм в формате SVG. Диаграммы, отображаемые с помощью dc.js, манипулируют данными и реагируют на

запросы, при этом обеспечивают мгновенную обратную связь при взаимодействии с пользователем [2].

Dc.js поддерживает обработку больших многомерных наборов данных, с помощью встроенной поддерживаемой библиотеки crossfilter. Благодаря данной библиотеке, мы можем фильтровать, сортировать, проводить группировку данных, при этом обеспечивая оперативную скорость взаимодействия [3, 4].

Соответственно, библиотека dc.js является связующим звеном crossfilter и d3.js, так как crossfilter способен только манипулировать данными, а d3.js – взаимодействовать с графиками [5, 6].

Для упрощения взаимодействия пользователя с html-страницей используется jQuery. Эта библиотека обеспечивает полное взаимодействие со структурой DOM (Document Object Model), позволяя взаимодействовать со структурой html-документа [7].

Для демонстрации работы возьмем датасет пользовательского трафика на веб-странице. На рисунке 1, можно наблюдать два блока, которые ответственны за количество посещений на веб-сайте. То есть, при начале веб-сеанса происходит запись в базу данных.

```
{
  "clientdataid": 149997,
  "availableresolution": "412x892",
  "browser": "Chrome",
  "browsermajorversion": 88,
  "colordepth": 24
},
{
  "clientdataid": 149996,
  "availableresolution": "360x800",
  "browser": "Chrome",
  "browsermajorversion": 80,
  "colordepth": 24
},
```

Рис. 1. – Пример набора входных данных.

Запись содержит несколько полей:

1. `clientdataid` – уникальный номер сеанса; присваивается, когда пользователь заходит на страницу;
2. `availableresolution` – максимальное разрешение, которое может иметь пользователь на своем устройстве;
3. `browser` – браузер, с которого был начат сеанс;
4. `browsermajorversion` – основная версия браузера;
5. `colordepth` – цветовая глубина устройства.

Набор данных будет содержаться в JSON файле, импортированном в функцию `crossfilter()`.

Данная функция предназначена для последующей группировки и фильтрации входных данных по необходимому ключу. В этом случае необходимым ключом будет являться `browser`. Для выдачи значений по ключу `browser`, используется метод `crossfilter.dimension(value)`.

С помощью указанного метода создается новое пространство доступа к значению. Он должен возвращать естественно упорядоченные значения, то есть значения, которые ведут себя правильно по отношению к операторам Javascript `<`, `<=`, `>` и `=`. Обычно это означает примитивы: логические значения, числа или строки.

Стоит отметить, что несравнимые значения, такие, как `NaN` и `undefined`, не поддерживаются. Также следует соблюдать осторожность при смешивании типов, например, строк и чисел. Если строки и числа смешаны, строки будут принудительно преобразованы в числа, в противном случае появятся неподдерживаемые значения `NaN` [8].

Так как отрисовка диаграммы будет происходить на html-странице, то сначала необходимо обратиться к `id` необходимого тега [9].

```
<div id="pie-chart"></div>
```

Зная id тега div, можем передать его в качестве аргумента в функцию pieChart(). PieChart – специальная функция, служащая для отрисовки круговой диаграммы [10].

```
const speed_pie = dc.pieChart("#sample-pie");
```

В sample\_data хранится набор метаданных, которые понадобятся в дальнейшем отображении на диаграммах.

Наложим оболочку на входные данные для дальнейшей фильтрации:

```
const cfData = crossfilter(sample_data);
```

Поместив json-объект в crossfilter, реализуем функцию обратного вызова, изымая ключ browser.

```
browserDimension = cfData.dimension((data) => {  
    return data.browser;  
})
```

Далее, создадим группу значений для browser, применив функцию group(). В данную группу будут входить такие подгруппы браузеров, как: chrome, mobile safari, mozilla firefox, opera.

```
const browserGroup = browserDimension.group();
```

Теперь произведем рендеринг круговой диаграммы, используя функцию ниже:

```
speed_pie  
    .width (800)  
    .height (300)  
    .dimension (browserDimension)  
    .group (browserGroup)  
    .on ('renderlet', function (chart) {  
    });
```

- Width () – ширина круговой диаграммы установлена на 800 пикселей.

---

- `height ()` – Высота круговой диаграммы установлена на 300 пикселей.
- `dimension ()` – метод, куда необходимо подставить изъятый ранее ключ `browser`.
- `group()` – метод, куда подставляются выбранные подгруппы браузеров по ключу `browser`.
- `on()` – событийная функция, где передаваемым параметром является `'renderlet'`.

Для рендеринга диаграмм используется следующий метод:

```
dc.renderAll();
```

На рисунке 2 можно наблюдать отрисованную круговую диаграмму на статической html-странице.

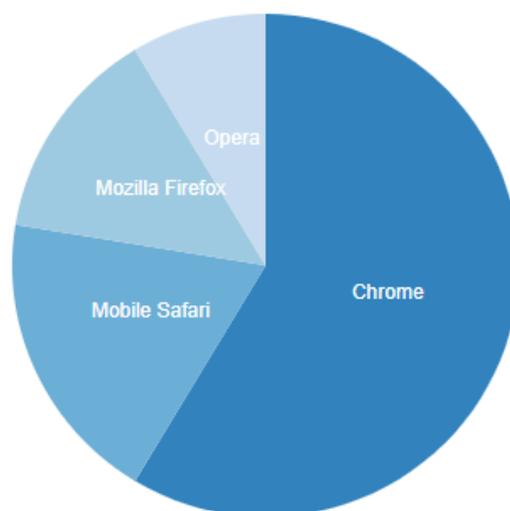


Рис. 2. – Круговая диаграмма с подгруппами браузеров.

Далее, следует подсчитать количество браузеров для каждой подгруппы. Подсчет к созданному выше набору данных – `browserDimension`, осуществляется с помощью функции `all ()`:

```
browserDimension.all ();
```

На рисунке 3 можно наблюдать json массив, где указана подгруппа браузеров с ключом `key` и количество упоминаний браузеров с ключом `value`.

```
· (4) [{...}, {...}, {...}, {...}] ⓘ  
  ▶ 0: {key: "Chrome", value: 34}  
  ▶ 1: {key: "Mobile Safari", value: 11}  
  ▶ 2: {key: "Mozilla Firefox", value: 8}  
  ▶ 3: {key: "Opera", value: 5}  
  length: 4  
  ▶ __proto__: Array(0)
```

Рис. 3. – Набор выходных данных по ключу browser.

Опираясь на полученные данные, можно сделать следующие выводы: браузер “Chrome” занимает лидирующую позицию среди других браузеров, и его величина составляет 59% от общего количества. Самым малоиспользуемым браузером является “Opera”, его величина составляет 9%. Задействовав библиотеку Dimensional Charting, можно проводить веб-аналитику, не используя сторонние языки программирования. Достаточно иметь подключенный скрипт библиотеки в html-документе. Большое разнообразие передаваемых данных имеет json-формат, это облегчает обработку разного рода информации, полученной во время браузерной сессии.

## Литература

1. Прошчаева А.А., Синелобова С.В. Обзор программного обеспечения для построения распределенных вебсистем // Инженерный вестник Дона, 2017, № 4. URL: [ivdon.ru/uploads/article/pdf/IVD\\_130\\_proshchaeva\\_sinelobova.pdf\\_75c41b8172.pdf](http://ivdon.ru/uploads/article/pdf/IVD_130_proshchaeva_sinelobova.pdf_75c41b8172.pdf).
2. Dimensional Charting Javascript Library // [dc-js.github.io](http://dc-js.github.io). URL: [dc-js.github.io/dc.js/](http://dc-js.github.io/dc.js/).
3. Crossfilter // [crossfilter.github.io](http://crossfilter.github.io). URL: [crossfilter.github.io/crossfilter/](http://crossfilter.github.io/crossfilter/).
4. Fast Multidimensional Filtering for Coordinated Views // [square.github.io](http://square.github.io). URL: [square.github.io/crossfilter/](http://square.github.io/crossfilter/).

5. Crossfilter.js, dc.js и D3.js для визуализации Данных // habr.com  
habr.com/ru/post/189838/.
6. Data-Driven Documents // d3js.org/. URL: d3js.org.
7. Чекулаев А.А., Глод О.Д. Разработка веб-приложения с использованием Spring Framework и jQuery // Инженерный вестник Дона, 2018, № 4. URL: ivdon.ru/uploads/article/pdf/IVD\_305\_Glod.pdf\_9613ed584d.pdf.
8. API Reference, Crossfilter // github.com. URL: github.com/square/crossfilter/wiki/API-Reference.
9. dc.js getting started and how-to guide // dc-js.github.io URL: dc-js.github.io/dc.js/docs/stock.html dc.js getting started and how-to guide.
10. DC.js - Pie Chart // tutorialspoint.com. URL: tutorialspoint.com/dcjs/dcjs\_pie\_chart.htm.

### References

1. Proshhaeva A.A., Sinelobova S.V. Inzhenernyj vestnik Dona, 2017, № 4. URL: ivdon.ru/uploads/article/pdf/IVD\_130\_proshchaeva\_sinelobova.pdf\_75c41b8172.pdf.
  2. Dimensional Charting Javascript Library. dc-js.github.io. URL: dc-js.github.io/dc.js/.
  3. Crossfilter. Crossfilter.github.io. URL: crossfilter.github.io/crossfilter/.
  4. Fast Multidimensional Filtering for Coordinated Views. Square.github.io. URL: square.github.io/crossfilter/.
  5. Crossfilter.js, dc.js и D3.js dlja vizualizacii Danyyh. habr.com habr.com/ru/post/189838/.
  6. Data-Driven Documents. d3js.org/. URL: d3js.org.
-



7. Chekulaev A.A., Glod O.D. Inzhenernyj vestnik Dona, 2018, № 4.  
URL: [ivdon.ru/uploads/article/pdf/IVD\\_305\\_Glod.pdf\\_9613ed584d.pdf](http://ivdon.ru/uploads/article/pdf/IVD_305_Glod.pdf_9613ed584d.pdf).
8. API Reference, Crossfilter. Github.com. URL:  
[github.com/square/crossfilter/wiki/API-Reference](https://github.com/square/crossfilter/wiki/API-Reference).
9. Dc.js getting started and how-to guide. Dc-js.github.io URL: [dc-js.github.io](https://dc-js.github.io/dc.js/docs/stock.html) dc.js getting started and how-to guide.
10. DC.js Pie Chart. Tutorialspoint.com. URL:  
[tutorialspoint.com/dcjs/dcjs\\_pie\\_chart.htm](https://www.tutorialspoint.com/dcjs/dcjs_pie_chart.htm).