УДК 004

Создание визуальных объектов с помощью SVG

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В этой статье говорится о масштабируемой векторной графике, обычно называемой SVG. SVG – это веб-стандарт для создания векторной графики в браузере.
Ключевые слова: SVG, HTML, Векторная графика

Creating visual objects using SVG

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article talks about scalable vector graphics, commonly referred to as SVG. SVG is the web standard for creating vector graphics in a browser.
Keywords: SVG, HTML, Vector graphics

SVG - это язык разметки XML, который был разработан для отображения очень богатых 2D-визуализаций. SVG может использовать графический процессор компьютера для ускорения рендеринга, а также оптимизирован для взаимодействия с пользователем и анимации.

Ранее этим вопросом интересовались А.И.Телегин, Д.Н.Тимофеев, Д.И.Читалов, С. Г.Пудовкина развивали тему «SVG-Разметка двухмерной графики: опыт использования SVG в создании двухмерной графики» [1] в которой рассматриваются варианты использования языка декларативного программирования масштабируемой векторной графики. В.Я.Потапов с темой «Особенности использования векторной графики SVG для разработки пользовательских интерфейсов в обучающих интегрированных экспертных системах» [2] а подробнее про решение проблем при построении интерфейсов на базе технологии SVG. Д.В.Носиков, Л.Н.Королькова опубликовали статью «Методы визуализации объектов данных в интернете на основе технологии SVG» [3] рассказали про вопросы визуализации объектов данных в интернете при помощи графических веб-технологий, таких как WebGL, VRML, CANVAS и SVG.

Вместо прямого манипулирования пикселями на экране SVG использует векторы для построения модели, а затем преобразует в пиксели.

Это значительно упрощает кодирование визуальных элементов по сравнению с другими веб-технологиями, такими как HTML5 и Canvas.

Поскольку изображение сохраненный как векторное представление, визуализация модели может быть масштабируемой. Это связано с тем, что все визуальные элементы можно легко масштабировать (как в большую, так и в меньшую сторону), не приводя к визуальным артефактам в результате масштабирования.

SVG удобен потому, что он может использоваться непосредственно внутри HTML в браузерах, поддерживающих SVG.

Система координат SVG

Координата система SVG имеет начало в верхнем левом углу элемента SVG, который равен (0,0); значение x увеличивается вправо, в то время как значение y возрастает к низу. Это часто встречается в компьютерных графических системах, но иногда может вводить в заблуждение. Пример представлен на рисунке 1.

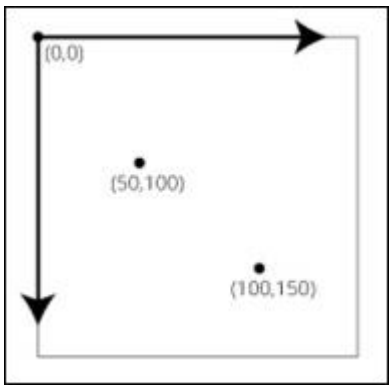

Рисунок 1

Атрибуты SVG

SVG, будучи в состоянии легко интегрироваться с HTML, не является HTML. В частности, свойства и стили могут работать по-разному. Примером этого является то, что большинство элементов HTML имеют элементы ширины и высоты, но не все элементы SVG используют эти свойства.

Второй важный момент в SVG заключается в том, что положение элемента задается через атрибуты. В связи с этим невозможно установить положение элементов SVG с использованием стиля.

Рисование кругов с помощью SVG

Мы работаем с SVG в HTML с использованием тега SVG и размещения элементы SVG в этом теге. Очень простой пример: создается три круга:

```
<svg width="720" height="120">
    <circle cx="40" cy="20" r="10"></circle>
    <circle cx="80" cy="40" r="15"></circle>
    <circle cx="120" cy="60" r="20"></circle>
</svg>
```

Это приводит к следующему изображению в браузере как на рисунке 2.

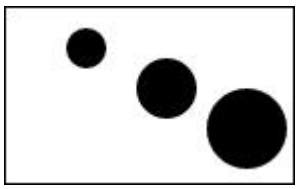

Рисунок 2

Элементу SVG всегда нужно указывать размеры, в этом примере он имеет 720 в ширину и 120 в высоту пикселей.

Позиционирование круга в элементе SVG выполняется путем указания центра x и y значений окружности. Это местоположение относится к верхнему левому углу элемента SVG, положительные x значения перемещаются вправо от начала координат, а положительные y значения перемещаются вниз. Размер круга определяется r атрибутом, который указывает радиус круга.

В примере не указывался цвет для этих кругов, поэтому цвет по умолчанию для кругов черный. Большинство элементов SVG определяют цвет, используя атрибут стиля CSS, а затем устанавливая fill атрибут стиля.

Например, следующий код дает разные цвета (красный, зеленый и синий) для трех кругов:

```
<svg width="720" height="120">
  <circle cx="40" cy="20" r="10" style="fill:red"></circle>
  <circle cx="80" cy="40" r="15" style="fill:green"></circle>
  <circle cx="120" cy="60" r="20" style="fill:blue"></circle>
</svg>
```

Это приводит к следующему результату как на рисунке 3.

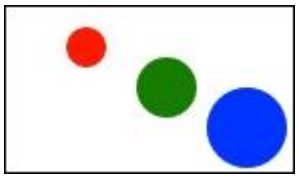

Рисунок 3

SVG имеет встроенные формы помимо круга.

Эллипс

Круг частный случай эллипса, который имеет одинаковые радиусы x и y. Эллипс указан в SVG с использованием <ellipse> тега. По-прежнему используется x и y атрибуты для размещения эллипса, но вместо использования r радиуса используется два атрибута rx и ry указывающие радиус в направлениях x и y:

```
<ellipse cx="50" cy="30" rx="40" ry="20" />
```

На рисунке 4 представлен результат отображения кода выше.

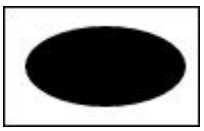

Рисунок 4

Прямоугольник

Прямоугольники указываются с помощью <rect> тега. Верхний левый угол задается с помощью x и y атрибутов. Высоту и ширину атрибутов определяют width и height соответственно.

```
<rect x="10" y="10" width="150" height="100"></rect>
```

На рисунке 5 изображен результат кода выше.

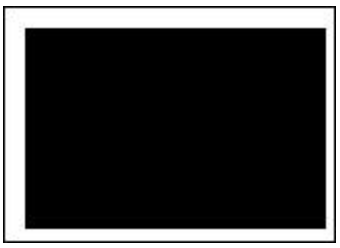

Рисунок 5

Линии

Можно рисовать линии с SVG с использованием <line> тега. Линия требует, по крайней мере, четыре атрибута, которые необходимо указать, и обычно использует пять. Первые два x1 и y1 укажите начальную позицию линии. Еще два атрибута x2 и y2 укажите конечную точку для линии. Последнее свойство, хотя и не обязательное, является stroke, которое определяет цвет линии.

```
<line x1="10" y1="10" x2="100" y2="100" stroke="black" />
```

На рисунке 6 изображен результат кода выше.

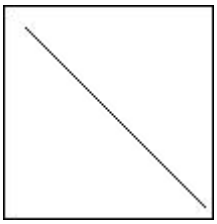

Рисунок 6

Пути

Пути одна из самых мощных чертежных конструкций в SVG. Они обеспечивают символическое понятие, которое можно использовать для создания сложных геометрий. Пути могут быть такими, как круги и

прямоугольники. Пути также предоставляют пользователю возможность создавать кривые с использованием контрольных точек.

Чертеж пути управляется путем указания одного атрибута, d которому передается строка, которая указывает команды, которые будут выполнены.

Основная концепция пути состоит в том, что вы можете нарисовать серию прямых или изогнутых линий, а затем заполнять пространство внутри, если форма закрыта. Например, следующая команда создает треугольник, заполненный черным цветом:

```
<path d="M 10 10 L 310 20 L 160 110 Z"/>
```

На рисунке 7 изображен результат кода выше.

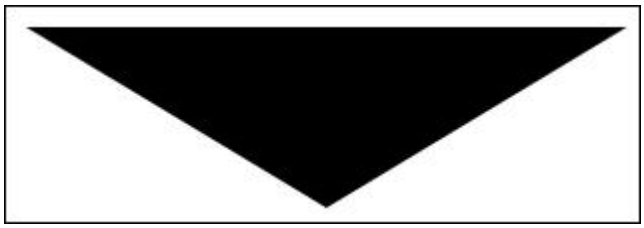

Рисунок 7

Текст

<text> SVG тег позволяет размещать текст в элементе SVG. Размещение текста в SVG отличается от того, как это делается в HTML. Элементы текста SVG рисуются векторной графикой вместо растровой. Следовательно, текст, отображаемый в SVG, более гибкий, чем растровый текст, отображаемый с помощью HTML. Текст позиционируется с использованием атрибутов x и y атрибутов, в которых указывается базовая линия текста, являющегося якорем позиционирования.

Также задает цвет шрифта, размер и цвет заливки. Фактический текст для отображения устанавливается с внутренним текстовым содержимым тега:

```
<text x="10" y="20"
      fill="Red" font-family="arial" font-size="16">
  Content of the SVG text
</text>
```

На рисунке 8 изображен результат кода выше.

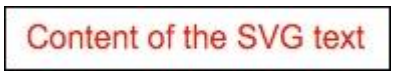

Рисунок 8

Элементы SVG могут быть одинаково оформлены так, как HTML элементы. С помощью CSS можно стилизовать элементы SVG. Тем не менее, многие из фактических стилей в HTML отличаются в SVG. Например, SVG использует fill для прямоугольника, тогда как HTML будет использовать фон для div тега, который представляет прямоугольник.

Следующие пример демонстрирует стиль SVG с CSS. В примере используются два стиля для установки заливок нескольких прямоугольников. Первый стиль сделает все прямоугольники красными по умолчанию. Второй определяет стиль, который делает все прямоугольники с идентификатором, willBeGreen заполненным зеленым цветом. В примере затем создаются три прямоугольника: первые два с использованием стилей CSS, а третий - с использованием CSS в стиле, attributeset заполнен blue.

Стили, определенные в образце, следующие:

```
<style>
    svg rect { fill: red; }
    svg rect#willBeGreen { fill: green; }
</style>
```

И прямоугольники создаются следующим образом:

```
<rect x="10" y="10" width="50" height="50" />
<rect x="70" y="10" width="50" height="50" id="willBeGreen" />
<rect x="130" y="10" width="50" height="50" style="fill:blue" />
```

Полученный результат будет таким, как показано на Рисунке 9.

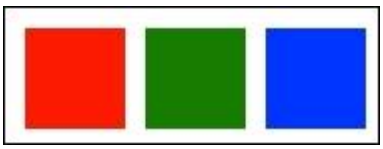

Рисунок 9

Формы SVG имеют атрибут, известный как stroke. Атрибут stroke определяет цвет линии, которая описывает форму SVG. Всякий раз, когда указывается stroke, обычно также указывается ширина штриха с использованием stroke-width атрибута. Это сообщает SVG о толщине (в пикселях) контура, который будет отображаться.

Демонстрирует stroke и stroke-width атрибуты, следующий пример.

```
<path d="M 10 10 L 210 10 L 110 120 z"
      style="fill:blue;stroke:red;stroke-width:5" />
```

Результат работы изображен на рисунке 10.

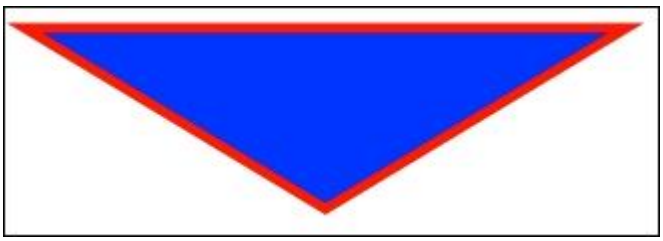

Рисунок 10

Можно устанавливать штрих на линии. Он также может иметь свой stroke-width набор. Увидеть это можно, изменив пример строки, чтобы установить толщину линии в 20 пикселей и цвет green:

```
<line x1="10" y1="10" x2="110" y2="110"
      stroke="green" stroke-width="20" />
```

Результат представлен на рисунке 11.

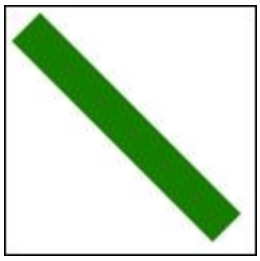

Рисунок 11

Обратите внимание, что эта строка выглядит как прямоугольник. Это связано с тем, что строки имеют атрибут, stroke-linecap который описывает форму конца строки, известную как ограничение строки.

По умолчанию для этого значение butt, которое дает нам острые углы на 90 градусов. другие значения, которые могут быть использованы, являются square или round. Следующие пример демонстрирует одну и ту же строку со всеми этими разными stroke-linecap значениями и изображены на рисунке 12.

```
<line x1="10" y1="20" x2="110" y2="100"
        stroke="red" stroke-width="20" stroke-linecap="butt" />
<line x1="60" y1="20" x2="160" y2="100"
      stroke="green" stroke-width="20" stroke-linecap="square" />
<line x1="110" y1="20" x2="210" y2="100"
      stroke="blue" stroke-width="20" stroke-linecap="round" />
<path d="M 10 20 L 110 100 M 60 20 L 160 100 M 110 20 L 210 100"
 stroke="white" />
```

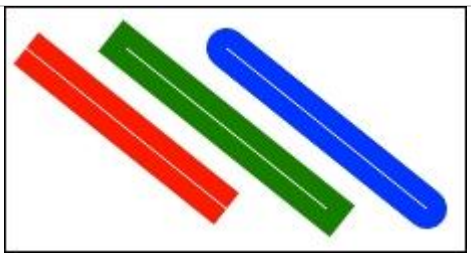

Рисунок 12

По умолчанию строки SVG являются сплошными, но они также могут быть созданы как тире, указанными с помощью stroke-dasharray атрибута. Этому атрибуту присваивается список целых значений, которые определяют повторяющийся рисунок ширины сегмента линии.

```
<line x1="10" y1="20" x2="110" y2="120"
        stroke="red" stroke-width="5"
        stroke-dasharray="5,5" />
<line x1="60" y1="20" x2="160" y2="120"
        stroke="green" stroke-width="5"
        stroke-dasharray="10,10" />
```

```
<line x1="110" y1="20" x2="210" y2="120"
        stroke="blue" stroke-width="5"
        stroke-dasharray="20,10,5,5,5" />
```

Результат можно наблюдать нарисунке13.

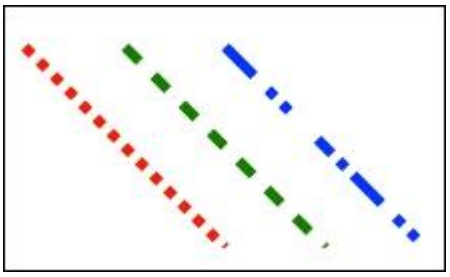

Рисунок 13

Применение SVG-преобразований

S в SVG означает масштабируемый, а V обозначает вектор. Это две важные части названия. Это позволяет нам применять различные преобразования форм SVG перед отображением.

Каждая форма SVG представлена одним или несколькими векторами, где вектор в SVG является кортежем (x, y) от начала координат в системе координат. В качестве примера прямоугольник будет представлен четырьмя двумерными векторами, по одному для каждого угла прямоугольника.

При создании графических визуализаций это моделирование данных с векторами имеет несколько преимуществ. Один из них состоит в том, что мы можем определить форму вокруг системы координат только для этой формы. Моделирование таким образом позволяет нам создавать копии фигуры, но размещать их в разных местах на большом изображении, поворачивать их, масштабировать и выполнять многие другие операции.

Во-вторых, эти преобразования применяются к модели до того, как будут отображаться на экране. Из-за этого SVG может гарантировать, что независимо от уровня масштабирования, применяемого к изображению, он не будет пиксельным.

Еще одна важная концепция трансформаций заключается в том, что они могут применяться в цепочке и в любой последовательности. Эта является чрезвычайно мощным понятием для создания составных моделей визуальных эффектов.

Рассмотрим три основных типа преобразований, предусмотренных SVG: translate, rotate и scale. Преобразования могут быть применены к элементу SVG с использованием transform атрибута.

Чтобы продемонстрировать преобразования, рассмотрим несколько примеров, которые применяют каждое преобразование к прямоугольнику, чтобы увидеть, как они влияют на результат отображения прямоугольника.

Поворот

Первое преобразование - это вращение. Мы можем повернуть объект SVG по указанному количество градусов rotate(x), где x указано количество градусов для поворота элемента.

Следующий пример поворачивает прямоугольник на 45 градусов.

```
<line x1="0" y1="150" x2="0" y2="0" stroke="black" />
<line x1="0" y1="0" x2="150" y2="0" stroke="black" />
<rect x="0" y="0 " width="100" height="100" fill="red"
      transform="rotate(45)" />
```

Предыдущий фрагмент дает нам следующий результат как на рисунке 14.

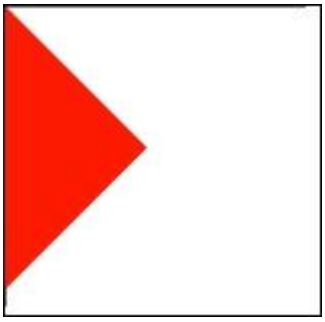

Рисунок 14

Перемещение

SVG элемент может быть перемещен используя translate() функцию. translate() принимает два значения: расстояние в x и y и расстояние для перемещения элемента внутри его родителя.

Следующий пример рисует прямоугольник и перемещает его в 30 пикселей вправо и 30 пикселей вниз:

```
<rect x="0" y="0 " width="100" height="100" fill="red"
      transform="translate(30,30)" />
```

Результат кода выше изображен на рисунке 15.

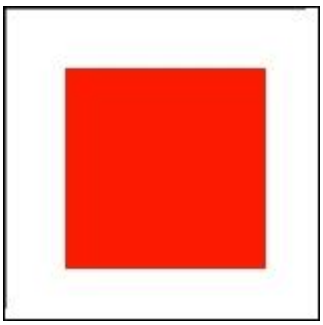

Рисунок 15

Масштаб

Масштабирование объекта это изменение своего визуального размера на определенный процент вдоль оси x и y. Масштабирование выполняется с

использованием функции scale(). Он может быть равномерно применен к каждой оси.

Следующий пример демонстрирует масштабирование. Рисуем два прямоугольника, один поверх другого. Прямоугольник внизу будет синим, а сверху, красный. Затем красный прямоугольник будет масштабироваться до 50% собственного размера:

```
<rect x="0" y="0" width="100" height="100" fill="blue"/>
<rect x="0" y="0" width="100" height="100" fill="red"
      transform="scale(0.5)" />
```

Результат кода выше изображен на рисунке 16.

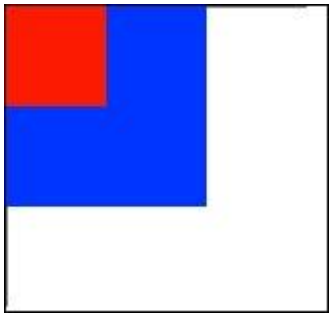

Рисунок 16

Группы

Элементы SVG могут быть сгруппированы вместе с помощью <g> тега. Любые преобразования, применяемые к группе, будут применяется к каждому из элементов в группе. Это удобно для применения общего преобразования только к определенной группе элементов.

Следующий пример демонстрирует как перенос группы элементов (синий прямоугольник с текстом) влияет на оба элемента. Обратите внимание, что зеленый прямоугольник не перемещается, потому что он не является частью преобразования:

```
<g transform="translate(100,30) rotate(45 50 50)">
  <rect x="0" y="0" width="100" height="100" style="fill:blue" />
  <text x="15" y="58" fill="White" font-family="arial"
        font-size="16">
        In the box
    </text>
</g>
```

Результат кода выше изображен на рисунке 17.

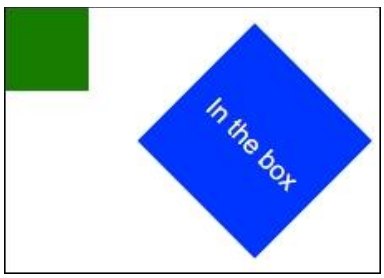

Рисунок 17

Также обратите внимание, что размещение текста поверх прямоугольника относительно верхнего левого угла группы, а не элемента SVG. Это важно для обеспечения правильного поворота текста относительно синего прямоугольника.

Прозрачность

SVG поддерживает прозрачность элементов. Эта может быть выполнено либо установкой opacity атрибута, либо использованием rgba (red-green-blue-alpha) значения при указании цвета.

Следующий пример отображает три круга разных цветов, все из которых прозрачны на 50 процентов. Первые два используют атрибут непрозрачности, а третий использует прозрачную спецификацию цвета для заливки.

```
<circle cx="150" cy="150" r="100"
        style="fill:red" opacity="0.5" />
<circle cx="250" cy="150" r="100"
        style="fill:green" opacity="0.5" />
<circle cx="200" cy="250" r="100"
        style="fill:rgba(0, 0, 255, 0.5)" />
```

Результат кода выше изображен на рисунке 18.

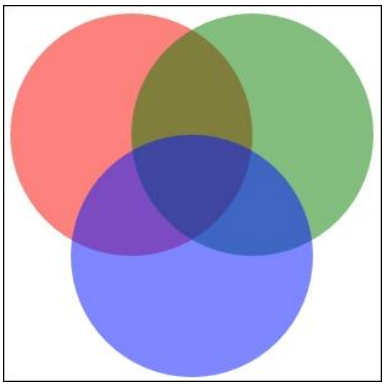

Рисунок 18

Слои

В предыдущем примере элементы накладывались друг на друга, хотя этого не было заметно. Следующий пример продемонстрирует как работают слои в SVG.

```
<circle cx="150" cy="150" r="100" style="fill:red" />
<circle cx="250" cy="150" r="100" style="fill:green" />
<circle cx="200" cy="250" r="100" style="fill:blue" />
```

Результат кода выше изображен на картинке 19.

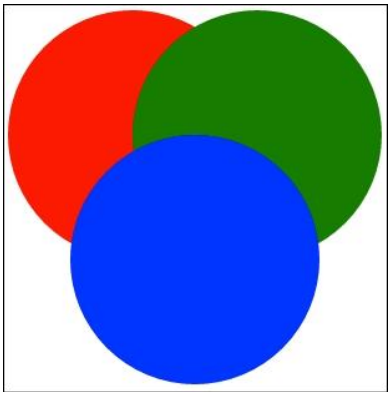

Рисунок 19

Синий круг рисуется перед зеленым кругом, который рисуется перед красным кругом. Этот порядок определяется последовательностью, указанной в разметке SVG, причем каждый последующий элемент визуализируется поверх предыдущих элементов.

Вывод

Мы узнали, как использовать SVG для создания различных фигур, как выложить элементы SVG с использованием координат SVG и как слои влияют на рендеринг. Мы также научились выполнять трансформацию на элементах SVG.

Библиографический список

1. Телегин А.И., Тимофеев Д.Н., Читалов Д.И., Пудовкина С.Г. SVG-разметка двухмерной графики: опыт использования SVG в создании двухмерной график. Челябинск: Южно-Уральский государственный университет (национальный исследовательский университет). 2015. С. 73. Ссылка https://elibrary.ru/item.asp?id=24744381 (Дата обращения: 19.11.2017)
2. Потапов В.Я. Особенности использования векторной графики SVG для разработки пользовательских интерфейсов в обучающих интегрированных экспертных системах // Научная сессия НИЯУ МИФИ-2015. 2015. С. 148. Ссылка https://elibrary.ru/item.asp?id=24780547 (Дата обращения: 19.11.2017)
3. Носиков Д. В., Королькова Л. Н. Методы визуализации объектов данных в интернете на основе технологии SVG // Инновационные направления развития в образовании, экономике, технике и технологиях. 2016. С. 126-131. Ссылка https://elibrary.ru/item.asp?id=26036844 (Дата обращения: 19.11.2017)