



Бесплатная электронная книга

# УЧУСЬ SVG

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#svg

.....	1
<b>1: SVG</b> .....	<b>2</b>
.....	2
.....	2
Examples.....	3
SVG.....	3
SVG .....	4
SVG .....	4
<b>2: clipPath</b> .....	<b>6</b>
.....	6
.....	6
Examples.....	6
.....	6
<b>3: DEFS</b> .....	<b>7</b>
.....	7
.....	7
.....	7
Examples.....	7
.....	7
<b>4: Scripting</b> .....	<b>9</b>
.....	9
pathSegList   SVGPathSeg.....	9
getTransformToElement().....	9
Examples.....	9
.....	9
/ .....	11
.....	11
.....	11
SVG.....	12
<b>5:</b> .....	<b>15</b>
.....	15

Examples.....	15
h31.....	15
h32.....	15
<b>6: .....</b>	<b>16</b>
.....	16
.....	17
Examples.....	17
LinearGradient.....	17
RadialGradient.....	17
<b>7: .....</b>	<b>18</b>
.....	18
.....	18
Examples.....	18
.....	18
<b>8: .....</b>	<b>20</b>
.....	20
.....	20
Examples.....	20
.....	20
<b>9: .....</b>	<b>22</b>
.....	22
.....	22
Examples.....	22
, .....	22
-.....	23
-:.....	23
.....	24
<b>10: .....</b>	<b>25</b>
.....	25
.....	25
Examples.....	25

SVG, .....	25
, .....	25
<b>11: .....</b>	<b>28</b>
.....	28
.....	28
.....	28
Examples.....	29
.....	29
refX, refY orient.....	30
Units, markerWidth, markerHeight.....	31
, , , .....	33
<b>12: .....</b>	<b>35</b>
.....	35
.....	35
Examples.....	35
.....	35
.....	35
.....	36
.....	36
<b>13: .....</b>	<b>37</b>
.....	37
Examples.....	37
.....	37
<b>14: .....</b>	<b>38</b>
.....	38
Examples.....	38
.....	38
.....	39
.....	39
skewX, skewY.....	40
.....	40
.....	41

<b>15:</b>	<b>42</b>
.....	42
Examples.....	42
.....	42
.....	42
.....	42
.....	42
.....	43
<b>16:</b>	<b>44</b>
.....	44
.....	44
Examples.....	44
.....	44
.....	45
<b>17:</b>	<b>46</b>
.....	46
.....	46
.....	47
Examples.....	47
L path.....	47
, H.....	48
, I ( ).....	48
, V path.....	49
<b>18:</b>	<b>50</b>
.....	50
.....	50
.....	50
Examples.....	50
.....	50
.....	51
, .....	51

<b>19:</b>	<b>53</b>
.....	53
.....	53
Examples.....	53
.....	53
- .....	54
.....	54
X Y.....	54
<b>20:</b>	<b>56</b>
.....	56
.....	56
Examples.....	56
objectBoundingBox.....	56
patternUnits patternContentUnits.....	57
.....	58
<b>21:</b>	<b>60</b>
.....	60
Examples.....	60
.....	60
.....	60
<b>22:</b>	<b>61</b>
.....	61
.....	61
.....	62
Examples.....	64
: feGaussian Blur ( ).....	64
: feGaussianBlur ( x Y ).....	65
: feGaussianBlur 100%.....	66
: .....	66
: Bokeh Blur (3 , ).....	67
: Dropshadow.....	68
: .....	69

: ( , ).....	69
: .....	70
: ( ).....	71
: .....	71
: ().....	72
: .....	73
: .....	73
<b>23: .....</b>	<b>75</b>
Examples.....	75
- fill stroke.....	75
RGB .....	75
RGB - .....	75
currentColor.....	75
<b>24: SVG.....</b>	<b>77</b>
Examples.....	77
Viewbox.....	77
preserveAspectRatio.....	77
preserveAspectRatio - .....	78
<b>25: .....</b>	<b>80</b>
.....	80
Examples.....	80
.....	80
.....	81

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [svg](#)

It is an unofficial and free SVG ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official SVG.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)



# глава 1: Начало работы с SVG

## замечания

Масштабируемая векторная графика (SVG) является [стандартом W3C](#) для рисования векторных изображений.

Вот простой автономный файл SVG:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="25" fill="blue"/>
</svg>
```

SVG также может быть встроен в HTML, и в этом случае атрибут `xmlns` не требуется.

Другие графические элементы:

- `<line>`
- `<ellipse>`
- `<path>`
- `<polygon>` и `<polyline>`
- `<text>` включая дочерние элементы, такие как `<tspan>` и `<textPath>`

CSS используется для стилизации, хотя не все свойства CSS применяются к SVG, и сам SVG определяет некоторые конкретные свойства, такие как `fill` и `stroke`, которые не используются в других местах.

Формы могут быть заполнены градиентами или узорами, а дополнительные растровые эффекты могут быть достигнуты с использованием фильтров.

Обрезание доступно с использованием вышеуказанных графических элементов в качестве путей клипов.

Что касается версий стандарта W3C SVG:

- Текущей версией является [SVG 1.1 \(второе издание\)](#)
- В настоящее время W3C работает над проектом [SVG 2](#)

## Версии

Версия	Дата выхода
<a href="#">1,0</a>	2001-09-04
<a href="#">1.1 Первое издание</a>	2003-01-14

Версия	Дата выхода
1.2 Крошечный	2008-12-22
1.1 Второе издание	2011-08-16

## Examples

### Встроенный SVG

Inline SVG позволяет разметку SVG, написанную внутри HTML, генерировать графику в браузере.

При использовании SVG inline DOCTYPE строго не требуется. Вместо этого достаточно всего лишь тегов открытия и закрытия `<svg>` вместе с [атрибутами viewBox](#) или `width` и `height`:

```
<svg width="100%" height="100%">
  <!-- SVG elements go here -->
</svg>
```

Фрагмент `<svg>` действует как контейнер и структурный элемент. Этот фрагмент устанавливает собственную систему координат.

Ниже приведен пример рендеринга фрагмента SVG с некоторым контентом. Он создаст прямоугольник с «Hello World!». текст внутри него.

```
<svg width="50%" viewBox="0 0 10 10">
  <rect x="1" y="1" width="5" height="3" fill="teal" />
  <text x="2" y="2" font-family="Palatino, Georgia, serif" font-size="3%" font-weight="bold"
fill="white">Hello World!</text>
</svg>
```

Результат:



Hello World!

## SVG как

Вы можете отобразить содержимое SVG-файла в виде изображения в документе HTML с помощью `<img>`. Например:

```

```

Размеры изображения по умолчанию будут отображаться в соответствии с параметрами **ширины и высоты**, указанными в файле SVG, указанном в атрибуте `src`.

Стоит отметить различные ограничения, присущие этому подходу:

- Поддержка браузера, хотя и хорошая, не включает Internet Explorer 8 и более ранние версии, а также Android 2.3 и более ранние версии.
- Вы не можете стилизовать отдельные элементы, содержащиеся в SVG-файле, используя CSS, который является внешним по отношению к SVG-файлу. Все CSS должны находиться внутри самого файла изображения.
- JavaScript не запускается.
- Изображение должно быть заполнено в одном файле. Например, если файл SVG содержит растровые изображения, то эти внутренние изображения должны быть закодированы как URL-адреса данных.

## SVG в качестве фонового изображения

Вы можете отобразить SVG-файл в HTML-документе, указав его как фоновое изображение в CSS. Например:

```
.element {  
    background-size: 100px 100px;
```

```
background: url(my_svg_file.svg);  
height: 100px;  
width: 100px;  
}
```

Если размеры, указанные в вашем SVG-файле, больше размеров вашего HTML-элемента, может быть желательно указать свойство `background-size`, чтобы масштабировать SVG, чтобы соответствовать его элементу.

Как и при использовании [SVG в качестве <img>](#), стоит отметить некоторые ограничения этого подхода:

- Поддержка браузера не включает Internet Explorer 8 и более ранние версии, а также Android 2.3 и более ранние версии.
- Вы не можете стилизовать отдельные элементы, содержащиеся в SVG-файле, используя CSS, который является внешним по отношению к SVG-файлу. Все CSS должны находиться внутри самого файла изображения.

Прочитайте [Начало работы с SVG онлайн](#): <https://riptutorial.com/ru/svg/topic/963/начало-работы-с-svg>

# глава 2: clipPath

## параметры

параметр	Описание
clipPathUnits	система координат содержимого шаблона либо <i>objectBoundingBox</i> , либо <i>userSpaceOnUse</i>

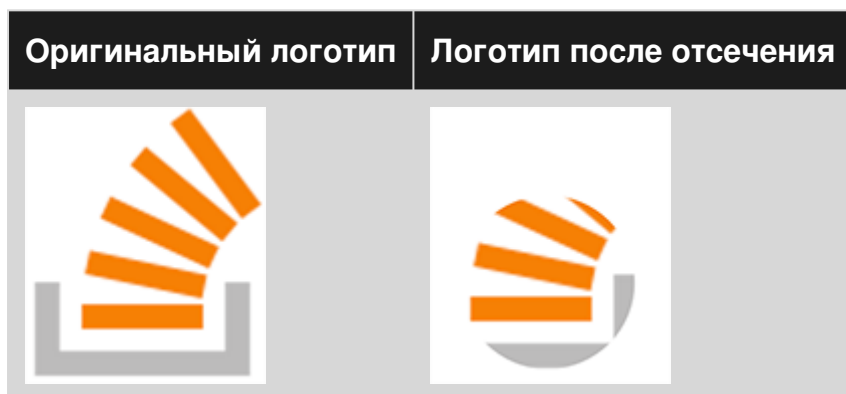
## замечания

Связанные данные W3C

## Examples

### Обрезание по круговой траектории

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <clipPath id="circleClip">
      <circle cx="50" cy="60" r="20" />
    </clipPath>
  </defs>
  <image width="100" height="100" style="clip-path:url(#circleClip)"
xlink:href="https://cdn.sstatic.net/Sites/stackoverflow/company/img/logos/so/so-icon.png" />
</svg>
```



Прочитайте clipPath онлайн: <https://riptutorial.com/ru/svg/topic/4840/clippath>

# глава 3: DEFS

## Синтаксис

- `<defs> ... определенные элементы ... </defs>`

## параметры

параметр	подробности
DEFS	Элемент <code>defs</code> не имеет параметров

## замечания

Элемент `<defs>` используется как элемент контейнера для элементов, которые предназначены для использования исключительно по ссылке и не отображаются непосредственно. Элементы, которые обычно будут отображаться (например, `<rect>`, `<circle>`), которые объявлены внутри блока `<defs>`, обрабатываются так, как если бы их стиль включал `display:none`.

Хотя это не является строго необходимым, спецификация SVG рекомендует помещать все определения градиента, фильтра, шаблона, маски, символа и маркера в блок `defs`.

## Examples

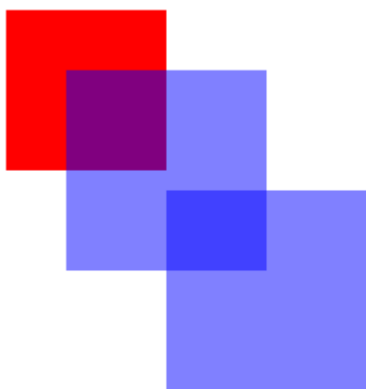
### основной пример

```
<svg width="400px" height="400px">
<defs>
  <rect id="defrect" fill="blue" fill-opacity=".5" x="50" y="50" width="100" height="100"/>
</defs>

<rect fill="red" x="20" y="20" width="80" height="80"/>
<use xlink:href="#defrect"/>
<use xlink:href="#defrect" x="50" y="60"/>

</svg>
```

### Результат



Прочитайте DEFS онлайн: <https://riptutorial.com/ru/svg/topic/5592/defs>

# глава 4: Scripting

## замечания

Scripting SVG с использованием собственных интерфейсов DOM в настоящее время (2016) в состоянии незначительного изменения. Текущий стандарт SVG (1.1) хорошо реализуется большинством основных веб-браузеров. Однако, поскольку стандарт SVG 2.0 находится в разработке, некоторые браузеры начали удалять функции SVG 1.1, которые будут устаревшими в версии 2.0. Вы можете увидеть полный список предлагаемых изменений от SVG 1.1 до SVG 2.0 в [Приложении L SVG 2.0](#).

## Замена `pathSegList` и другого использования `SVGPathSeg`

В SVG 1.1 элементы `<path>` определены как `pathSegList` свойство `pathSegList` которое дает доступ к собственному представлению всех [команд пути](#). Google Chrome v48 [удалил это свойство](#) в конце 2015 года в рамках подготовки к [предлагаемой замене в SVG 2.0](#). До тех пор, пока поддержка SVG 2.0 не будет добавлена, вы должны использовать polyfill, чтобы либо [вернуть функциональность 1.1](#), либо [реализовать предлагаемый API 2.0](#).

## Замена `getTransformToElement()`

Chrome v48 также [удалил](#) метод `SVGGraphicsElement.getTransformToElement()`. Для [реализации старого метода](#) существует простой полиполк.

## Examples

### Создание элемента

Самый простой способ понять, как создавать и изменять элементы SVG, - это работать с элементами, использующими [базовые](#) интерфейсы [DOM Level 2 Core](#), как и с HTML или XML.

Крайне важно, чтобы *элементы*, созданные из JavaScript, создавались в том же пространстве имен, которое было объявлено в элементе SVG - в этом примере: «<http://www.w3.org/2000/svg>». Однако почти все *атрибуты* элементов SVG не находятся в каком-либо пространстве имен. Вы **не** должны размещать их в пространстве имен SVG.

Здесь мы демонстрируем SVG, размещенный внутри HTML, поскольку это обычный случай:

```
<!doctype HTML>
<html><title>Creating an Element</title>
<body>
```



```

<svg xmlns="http://www.w3.org/2000/svg"
    width="100%" height="100%"
    viewBox="0 0 400 300"></svg>

<script>
    var svgNS = "http://www.w3.org/2000/svg";

    // Create a circle element (not part of the DOM yet)
    var circle = document.createElementNS(svgNS, 'circle'); // Creates a <circle/>
    circle.setAttribute('fill', 'red'); // Note: NOT setAttributeNS()
    circle.setAttribute('cx', 150);      // setAttribute turns 150 into a string
    circle.setAttribute('cy', '80');     // using a string works, too
    circle.setAttribute('r', 35);        // give the circle a radius so we can see it

    // Now, add the circle to the SVG document so we can see it
    var svg = document.querySelector('svg'); // the root <svg> element
    svg.appendChild(circle);
</script>
</body></html>

```

Существует несколько атрибутов, которые необходимо создать в определенном пространстве имен. Они перечислены с двоеточиями в их именах в [Индексе атрибутов SVG](#). В частности, они: `xlink:actuate`, `xlink:arcrole`, `xlink:href`, `xlink:role`, `xlink:show`, `xlink:title`, `xlink:type`, `xml:base`, `xml:lang` и `xml:space`. Установите эти атрибуты с помощью `setAttributeNS()`:

```

var svgNS = "http://www.w3.org/2000/svg";
var xlinkNS = "http://www.w3.org/1999/xlink";
var img = document.createElementNS( svgNS, 'image' );
img.setAttributeNS( xlinkNS, 'href', 'my.png' );

```

Если вы часто создаете элементы, особенно со многими атрибутами, вспомогательная функция, подобная следующей, может спасти вас при наборе текста, избежать ошибок и сделать код более удобным для чтения:

```

<!doctype HTML>
<html><title>Creating an Element</title>
<body>
    <svg xmlns="http://www.w3.org/2000/svg"></svg>
    <script>
        var svg = document.querySelector('svg');
        var circle = createOn( svg, 'circle', {fill:'red', cx:150, cy:80, r:35} );

        // Create an SVG element on another node with a set of attributes.
        // Attributes with colons in the name (e.g. 'xlink:href') will automatically
        // find the appropriate namespace URI from the SVG element.
        // Optionally specify text to create as a child node, for example
        // createOn(someGroup, 'text', {x:100, 'text-anchor':'middle'}, "Hello World!");
        function createOn(parentEl, name, attrs, text) {
            var doc=parentEl.ownerDocument, svg=parentEl;
            while (svg && svg.tagName!='svg') svg=svg.parentNode;
            var el = doc.createElementNS(svg.namespaceURI, name);
            for (var a in attrs){
                if (!attrs.hasOwnProperty(a)) continue;
                var p = a.split(':');
                if (p[1]) el.setAttributeNS(svg.getAttribute('xmlns:'+p[0]), p[1], attrs[a]);
            }
            if (text) el.appendChild(doc.createTextNode(text));
        }
    </script>
</body>
</html>

```

```

        else      el.setAttribute(a,attrs[a]);
    }
    if (text) el.appendChild(doc.createTextNode(text));
    return parentEl.appendChild(el);
}
</script>
</body></html>

```

## Атрибуты чтения / записи

Вы можете использовать либо методы [DOM Level 2 Core](#) `getAttribute()` , `getAttributeNS()` , `setAttribute()` и `setAttributeNS()` для чтения и записи значений из элементов SVG, либо вы можете использовать настраиваемые свойства и методы, указанные в `setAttributeNS()` [SVG 1.1](#) (интерфейс Определение языка).

## Простые числовые атрибуты

Например, если у вас есть элемент окружности SVG:

```
<circle id="circ" cx="10" cy="20" r="15" />
```

вы можете использовать методы DOM для чтения и записи атрибутов:

```

var circ = document.querySelector('#circ');
var x = circ.getAttribute('cx') * 1; // Use *1 to convert from string to number value
circ.setAttribute('cy', 25);

```

... или вы можете использовать пользовательские свойства `cx` , `cy` и `r` определенные для [SVGCircleElement](#) . Обратите внимание, что это не прямые числа, а вместо этого, как и многие другие в SVG 1.1, они позволяют получить доступ к анимированным значениям. Эти свойства имеют тип [SVGAnimatedLength](#) . Без учета единиц анимации и длины вы можете использовать такие атрибуты, как:

```

var x = circ.cx.baseVal.value; // this is a number, not a string
circ.cy.baseVal.value = 25;

```

## Трансформации

[Группы SVG](#) могут использоваться для перемещения, поворота, масштабирования и, в противном случае, преобразования нескольких графических элементов в целом. (Подробнее о переводах SVG см. В [главе 7](#) ). Вот группа, которая делает смайликом лицо, которое вы можете настроить размер, поворот и размещение, регулируя `transform` :

```

<g id="smiley" transform="translate(120,120) scale(5) rotate(30)">
  <circle r="20" fill="yellow" stroke-width="2"/>
  <path fill="none" d="M-10,5 a 5 3 0 0 0 20,0" stroke-width="2"/>

```

```
<circle cx="-6" cy="-5" r="2" fill="#000"/>
<circle cx="6" cy="-5" r="2" fill="#000"/>
</g>
```

Использование сценария для настройки масштаба этого с помощью методов DOM требует манипулирования всем атрибутом `transform` как строки:

```
var face = document.querySelector('#smiley');

// Find the full string value of the attribute
var xform = face.getAttribute('transform');

// Use a Regular Expression to replace the existing scale with 'scale(3)'
xform = xform.replace( /scale\s*\([^)]+\)/, 'scale(3)' );

// Set the attribute to the new string.
face.setAttribute('transform',xform);
```

С помощью SVG DOM можно пройти определенные преобразования в списке, найти нужный и изменить значения:

```
var face = document.querySelector('#smiley');

// Get the SVGTransformList, ignoring animation
var xforms = face.transform.baseVal;

// Find the scale transform (pretending we don't know its index)
for (var i=0; i<xforms.numberOfItems; ++i){
  // Get this part as an SVGTransform
  var xform = xforms.getItem(i);
  if (xform.type == SVGTransform.SVG_TRANSFORM_SCALE){
    // Set the scale; both X and Y scales are required
    xform.setScale(3,3);
    break;
  }
}
```

- Для перемещения и управления количеством преобразований см. [SVGTransformList](#) .
- Для управления отдельными преобразованиями см. [SVGTransform](#) .

## Перетаскивание элементов SVG

Использование мыши для перетаскивания элемента SVG (или группы элементов) может быть выполнено с помощью:

1. Добавление обработчика `mousedown` для запуска перетаскивания: добавление перевода элемента, который нужно использовать при перетаскивании (при необходимости), отслеживание событий `mousemove` и добавление `mouseup` для завершения перетаскивания.
2. Во время `mousemove` , преобразуя положение мыши из экранных координат в локальные координаты для объекта, который вы перетаскиваете, и соответственно обновите

перевод.

### 3. Во время mouseup mousemove mouseup mousemove И mouseup .

```
// Makes an element in an SVG document draggable.
// Fires custom `dragstart`, `drag`, and `dragend` events on the
// element with the `detail` property of the event carrying XY
// coordinates for the location of the element.
function makeDraggable(el){
  if (!el) return console.error('makeDraggable() needs an element');
  var svg = el;
  while (svg && svg.tagName!='svg') svg=svg.parentNode;
  if (!svg) return console.error(el,'must be inside an SVG wrapper');
  var pt=svg.createSVGPoint(), doc=svg.ownerDocument;

  var root = doc.rootElement || doc.body || svg;
  var xlate, txStartX, txStartY, mouseStart;
  var xforms = el.transform.baseVal;

  el.addEventListener('mousedown',startMove,false);

  function startMove(evt){
    // We listen for mousemove/up on the root-most
    // element in case the mouse is not over el.
    root.addEventListener('mousemove',handleMove,false);
    root.addEventListener('mouseup', finishMove,false);

    // Ensure that the first transform is a translate()
    xlate = xforms.numberOfItems>0 && xforms.getItem(0);
    if (!xlate || xlate.type != SVGTransform.SVG_TRANSFORM_TRANSLATE){
      xlate = xforms.createSVGTransformFromMatrix( svg.createSVGMatrix() );
      xforms.insertBefore( xlate, 0 );
    }
    txStartX=xlate.matrix.e;
    txStartY=xlate.matrix.f;
    mouseStart = inElementSpace(evt);
    fireEvent('dragstart');
  }

  function handleMove(evt){
    var point = inElementSpace(evt);
    xlate.setTranslate(
      txStartX + point.x - mouseStart.x,
      txStartY + point.y - mouseStart.y
    );
    fireEvent('drag');
  }

  function finishMove(evt){
    root.removeEventListener('mousemove',handleMove,false);
    root.removeEventListener('mouseup', finishMove,false);
    fireEvent('dragend');
  }

  function fireEvent(eventName){
    var event = new Event(eventName);
    event.detail = { x:xlate.matrix.e, y:xlate.matrix.f };
    return el.dispatchEvent(event);
  }

  // Convert mouse position from screen space to coordinates of el
```

```
function inElementSpace(evt){  
    pt.x=evt.clientX; pt.y=evt.clientY;  
    return pt.matrixTransform(el.parentNode.getScreenCTM().inverse());  
}  
}
```

Прочитайте Scripting онлайн: <https://riptutorial.com/ru/svg/topic/5021/scripting>

# глава 5: Анимация

## замечания

Анимация SMIL через элемент `<animate>` в настоящее время (июль 2016) поддерживается в основных браузерах, за исключением браузеров Microsoft. Существует библиотека (fakeSMIL), которая может использоваться как полипол для совместимости с Microsoft.

Chrome 45 не одобрял SMIL в пользу анимации CSS и синтаксиса декларативной анимации предстоящей веб-анимации, которая, к сожалению, частично реализована только в текущих браузерах. Но разработчики Chrome недавно приостановили свое намерение (см. [Этот ответ StackOverflow](#) )

## Examples

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
    <animate
      attributeType="XML"
      attributeName="height"
      begin="0s"
      dur="10s"
      from="100"
      to="200"
      repeatCount="indefinite"
    />
  </rect>
</svg>
```

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="50" y="50" height="100" width="100" stroke="black" fill="yellow">
    <animateTransform
      attributeType="XML"
      attributeName="transform"
      type="rotate"
      begin="0s"
      dur="10s"
      from="0"
      to="360"
      repeatCount="indefinite"
    />
  </rect>
</svg>
```

Прочитайте Анимация онлайн: <https://riptutorial.com/ru/svg/topic/3260/анимация>

## глава 6: Градиенты

### параметры

общий	определение
GradientUnits	система координат атрибутов градиента. Любой объект BoundingBox или userSpaceOnUse
gradientTransform	преобразование, применимое к содержимому градиента
spreadMethod	определяет, что происходит вне границ градиента. Либо пэд, либо отражать, либо повторять
XLink: HREF	ссылка на другой градиент, который предоставляет атрибуты или контент
-----	-----
Линейный градиент	Определение
-----	-----
x1	определяет вектор градиента
x2	см. x1
y1	см. x1
y2	см. x1
-----	-----
Радиальный градиент	Определение
-----	-----
cx	координата x внешнего центра градиента
cy	координата y внешнего центра градиента
r	внешний радиус градиента. Расположение стоп-стоп
Fx	координата x центра внутреннего градиента. Место остановки 0%

общий	определение
FY	местоположение у центра внутреннего градиента. Место остановки 0%

## замечания

SVG чувствителен к регистру, поэтому не забудьте использовать столицу G посередине.

## Examples

### LinearGradient

```
<svg>
  <defs>
    <linearGradient id='g' y1="100%" x2="100%">
      <stop offset='0%' stop-color='yellow' />
      <stop offset='100%' stop-color='green' />
    </linearGradient>
  </defs>
  <rect width='100%' height='100%' fill='url(#g)' />
</svg>
```

### RadialGradient

```
<svg>
  <defs>
    <radialGradient id="g">
      <stop offset="10%" stop-color="green" />
      <stop offset="90%" stop-color="white" />
    </radialGradient>
  </defs>

  <rect width='100%' height='100%' fill='url(#g)' />
</svg>
```

Прочитайте Градиенты онлайн: <https://riptutorial.com/ru/svg/topic/3346/градиенты>



# глава 7: использование

## параметры

параметр	подробности
Икс	координата оси x верхнего левого угла
Y	координата оси Y верхнего левого угла
ширина	ширина элемента <code>&lt;use&gt;</code>
рост	высота элемента <code>&lt;use&gt;</code>
XLink: HREF	идентификатор ресурса (относится к идентификатору другого элемента). SVG 2 предлагает отказаться от этого и заменить его на простой атрибут href

## замечания

Подробности можно найти в [Рекомендации W3C для SVG](#), а также в новой [Рекомендации кандидата по SVG2](#)

## Examples

### Использование значка

Элемент `<use>` часто используется для повторного использования значков в сотрудничестве с элементом `<symbol>`. Это выглядит так:

```
<svg>
  <symbol viewBox="0 0 16 16" id="icon-star">
    <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904 5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
  </symbol>
</svg>
```

И элемент `<use>` :

```
<svg>
  <use xlink:href="#icon-star"/>
</svg>
```

Элемент `<use>` копирует `<symbol>` и отображает его. Вы также можете переопределить

стили на `<symbol>` на отдельных элементах `<use>` , например

```
<style>
  .red {
    fill: red;
  }
</style>

<svg>
  <use class="red" xlink:href="#icon-star"/>
</svg>
```

Прочитайте использование онлайн: <https://riptutorial.com/ru/svg/topic/6904/использование>

# глава 8: Круг

## параметры

параметры	подробности
cx	х-координата центра круга.
cy	у-координата центра круга.
r	Радиус круга.
Инсульт	Цвет границы круга.
заполнить	Цвет <i>внутри</i> границы круга.

## замечания

Подробную информацию о элементе SVG «circle» можно найти в [Рекомендации W3C для SVG](#).

## Examples

### Нарисуйте черный круг без заполнения

- Значения `cx` и `cy` обозначают местоположение центра круга.
- Атрибут `r` определяет размер радиуса круга.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  
  <circle cx="40" cy="40" r="30" stroke="black" fill="none" />  
</svg>
```

Результат:



Прочитайте Круг онлайн: <https://riptutorial.com/ru/svg/topic/1559/круг>

## глава 9: Линия

### параметры

атрибут	Описание
x1	Горизонтальное положение начала линии.
y1	Вертикальное положение начала линии.
x2	Горизонтальное положение конца линии.
y2	Вертикальное положение конца линии.
Инсульт	Цвет линии.
Ход ширина	Ширина линии.
штрих-непрозрачность	Непрозрачность линии.
штрих-dasharray	Чертеж для линии
штрих-linecap	Как линия заканчивается

### замечания

Подробную информацию о элементе SVG «line» можно найти в [Рекомендации W3C для SVG](#).

### Examples

Нарисуйте крест, используя диагональные красные линии

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <line x1="10" y1="10" x2="100" y2="100" stroke="red" stroke-width="10" />
  <line x1="100" y1="10" x2="10" y2="100" stroke="red" stroke-width="10" />
</svg>
```

Результат:



## Пунктирная линия с инсультом-дашараем

```
<svg width="400px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <line x1="10" y1="10" x2="300" y2="10" stroke="red" stroke-width="10" stroke-
dasharray="20,2,5,2"/>
</svg>
```

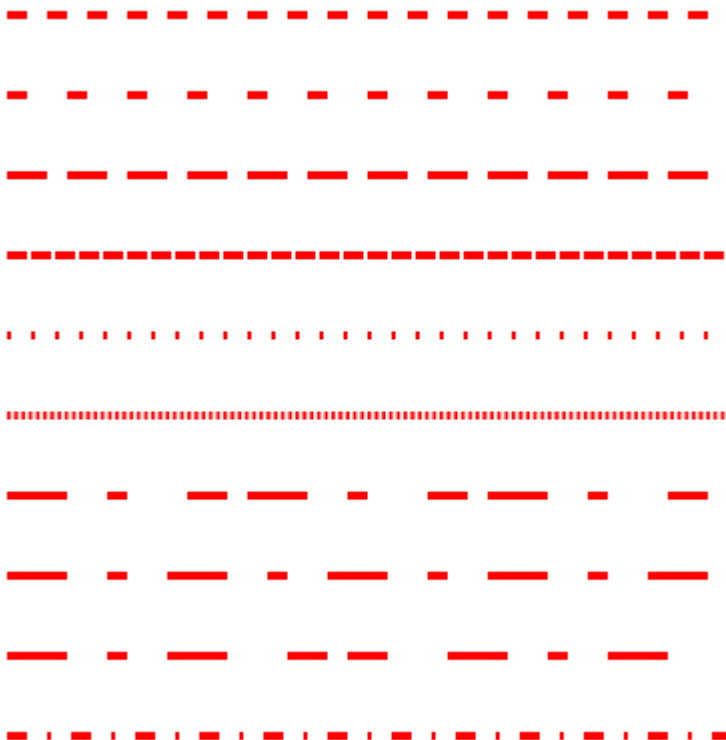
### Результат



## Различные примеры инсульта-дашара:

```
<svg width="200" height="200" viewBox="0 0 200 200" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <line stroke-dasharray="5, 5" x1="10" y1="10" x2="190" y2="10" />
  <line stroke-dasharray="5, 10" x1="10" y1="30" x2="190" y2="30" />
  <line stroke-dasharray="10, 5" x1="10" y1="50" x2="190" y2="50" />
  <line stroke-dasharray="5, 1" x1="10" y1="70" x2="190" y2="70" />
  <line stroke-dasharray="1, 5" x1="10" y1="90" x2="190" y2="90" />
  <line stroke-dasharray="0.9" x1="10" y1="110" x2="190" y2="110" />
  <line stroke-dasharray="15, 10, 5" x1="10" y1="130" x2="190" y2="130" />
  <line stroke-dasharray="15, 10, 5, 10" x1="10" y1="150" x2="190" y2="150" />
  <line stroke-dasharray="15, 10, 5, 10, 15" x1="10" y1="170" x2="190" y2="170" />
  <line stroke-dasharray="5, 5, 1, 5" x1="10" y1="190" x2="190" y2="190" />
  <style><![CDATA[
    line{
      stroke: red;
      stroke-width: 2;
    }
  ]]></style>
</svg>
```

### Результат:



## Варианты ограничения линии с использованием штриховки

```
<svg width="600px" height="400px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

  <line x1="10" y1="20" x2="300" y2="20" stroke="red" stroke-width="20" stroke-
linecap="butt"/>
  <text x="320" y="20">stroke-linecap="butt" (default)</text>
  <line x1="10" y1="70" x2="300" y2="70" stroke="red" stroke-width="20" stroke-
linecap="round"/>
  <text x="320" y="70">stroke-linecap="round"</text>
  <line x1="10" y1="120" x2="300" y2="120" stroke="red" stroke-width="20" stroke-
linecap="square"/>
  <text x="320" y="120">stroke-linecap="square"</text>
</svg>
```

### Результат



Прочитайте Линия онлайн: <https://riptutorial.com/ru/svg/topic/3034/линия>

# глава 10: Ломаная

## Синтаксис

- `<polyline points="10,5 25,15 20,10" />`

## параметры

параметр	подробности
точки	Атрибут точек определяет список точек. Каждая точка определяется координатой <i>ax</i> и <i>ay</i> в пользовательской системе координат.
Ход ширина	Ширина хода
штрих-непрозрачность	Непрозрачность удара
штрих-dasharray	(Необязательно) Определяет штрих-код для штриха
штрих-linecap	(Необязательно) Указывает, должен ли конец строки быть скрытым, круглым или квадратным («butt» (по умолчанию) / «round» / «square»)
штрих-linejoin	(Необязательно) Определяет, как сегменты линии должны быть соединены - митированные, округлые или скошенные («митра» (по умолчанию) / «круглый» / «скос»)
штрих-miterlimit	(Необязательно) Определяет максимальный размер митра. Согласованные соединения, превышающие этот предел, преобразуются в коническое соединение. По умолчанию = "4"

## Examples

### SVG, включая полилинию

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <polyline points="10,5 25,15 20,10" />
</svg>
```

### Полилинии с альтернативными линиями, линиями и митрами



```

<svg width="600px" height="600px" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">

  <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" />

  <text x="320" y="20">Default drawing stroke</text>

  <g transform="translate(0,150)">
    <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="butt" stroke-linejoin="miter" stroke-
miterlimit="2"/>

    <text x="320" y="20">stroke-linecap="butt" (default)</text>
    <text x="320" y="40">stroke-linejoin="miter" (default)</text>
    <text x="320" y="60">stroke-miterlimit="2"</text>
  </g>

  <g transform="translate(0,300)">
    <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="round" stroke-linejoin="round" />

    <text x="320" y="20">stroke-linecap="round" </text>
    <text x="320" y="40">stroke-linejoin="round" </text>

  </g>

  <g transform="translate(0,450)">
    <polyline points="10,10,50,40,80,30,120,90,130,10,180,50,250,100,300,10" fill="none"
stroke="red" stroke-width="10" stroke-linecap="square" stroke-linejoin="bevel"/>

    <text x="320" y="20">stroke-linecap="square"</text>
    <text x="320" y="40">stroke-linejoin="bevel"</text>
  </g>

</svg>

```

## Результат



Прочитайте Ломаная онлайн: <https://riptutorial.com/ru/svg/topic/3842/ломаная>

# глава 11: маркер

## Синтаксис

- `<marker viewBox = " xy width height " refX = " xoffset " refY = " yoffset " orient = " orientation " ... необязательные параметры >`
- `... элементы рисования маркера ...`
- `</marker >`
- `< elementname marker-start = "url (# markerid )" />` применяет маркер к началу элемента
- `< elementname marker-mid = "url (# markerid )" />` применяет маркер к середине сегмента элемента
- `< elementname marker-end = "url (# markerid )" />` применяет маркер к концу элемента
- Маркеры могут применяться к элементам `<line>` , `<polyline>` , `<polygon>` И `<path>`

## параметры

параметр	подробности
Viewbox	Указывает систему единиц для элементов, которые рисуют маркер
REFx	Расстояние оси x системы координат для рисования маркера должно быть смещено от точки рисования по умолчанию. По умолчанию 0.
refY	Расстояние оси y системы координат для рисования маркера должно быть смещено от точки рисования по умолчанию. По умолчанию 0.
восток	Значения являются <code>auto</code> или <code>angle in degrees</code> и указывают вращение, применяемое к маркеру. Он применяется после всех других корректировок координат ( <code>viewBox</code> , <code>preserveAspectRatio</code> и <code>refX</code> , <code>refY</code> ). Значение по умолчанию 0. Расчет угла для <code>auto</code> является сложным - подробности см. В спецификации SVG.
markerUnits	<code>strokeWidth</code> или <code>userSpaceOnUse</code> . По умолчанию используется значение <code>strokeWidth</code> .
markerWidth	Ширина маркера в маркерах. По умолчанию 3.
markerHeight	Высота маркера в маркерах. По умолчанию 3

## замечания

Сценарии: отображаемые маркерные элементы не отображаются в DOM, поэтому

невозможно настроить свойства или элементы для определенных отображаемых маркеров (хотя вполне возможно скриптировать определенный элемент маркера).

Свойство `overflow` элемента маркера автоматически устанавливается в `hidden`. Это то, что зажимает любой чертеж, который переполняет маркерную плитку. Это может быть явно установлено `visible` в CSS. По состоянию на июль 2016 года Chrome не поддерживает маркеры с `overflow: visible` - но обходным путем является установка фильтра на элемент маркера, который, как представляется, отключает отсечение переполнения.

Фильтры могут применяться к элементам маркера. Хотя явно не разрешено в спецификации, фильтры также работают, если они указаны в самом элементе маркера.

Более подробную информацию о элементе маркера см. В [разделе маркера в спецификации SVG 1.1](#).

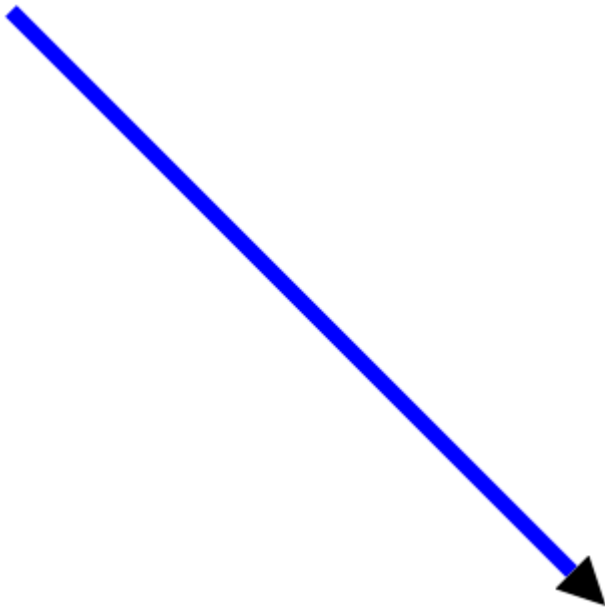
## Examples

### Основной маркер

Это пример конечного маркера, заданного с минимальным количеством параметров. Обратите внимание, что цвет штриха исходного элемента не наследуется маркером.

```
<svg width="800px" height="600px">
<defs>
  <marker id="examplemarker"
    viewBox="0 0 10 10"
    refX="0" refY="5"
    orient="auto">
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>
</defs>

  <line x1="20" y1="20" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#examplemarker)" />
</svg>
```



## Эффекты альтернативных значений refX, refY и orient

refX смещения для рисования маркера, заданные refX и refY , применяются до поворота, заданного параметром orient . Ниже вы можете увидеть эффекты различных комбинаций orient и refX , refY чтобы проиллюстрировать это.

```
<svg width="800px" height="600px">
<defs>
  <marker id="marker1"
    viewBox="0 0 10 10" refX="0" refY="5" orient="auto" >
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>

  <marker id="marker2"
    viewBox="0 0 10 10" refX="0" refY="0" orient="0" >
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>

  <marker id="marker3"
    viewBox="0 0 10 10" refX="20" refY="20" orient="0" >
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>

  <marker id="marker4"
    viewBox="0 0 10 10" refX="20" refY="20" orient="180" >
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>
</defs>

<line x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />

<text x="20" y="150"> refX,Y (0,5) orient (auto) </text>

<line x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />

<text x="220" y="150"> refX,Y (0,0) orient (0) </text>

<line x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
```

```

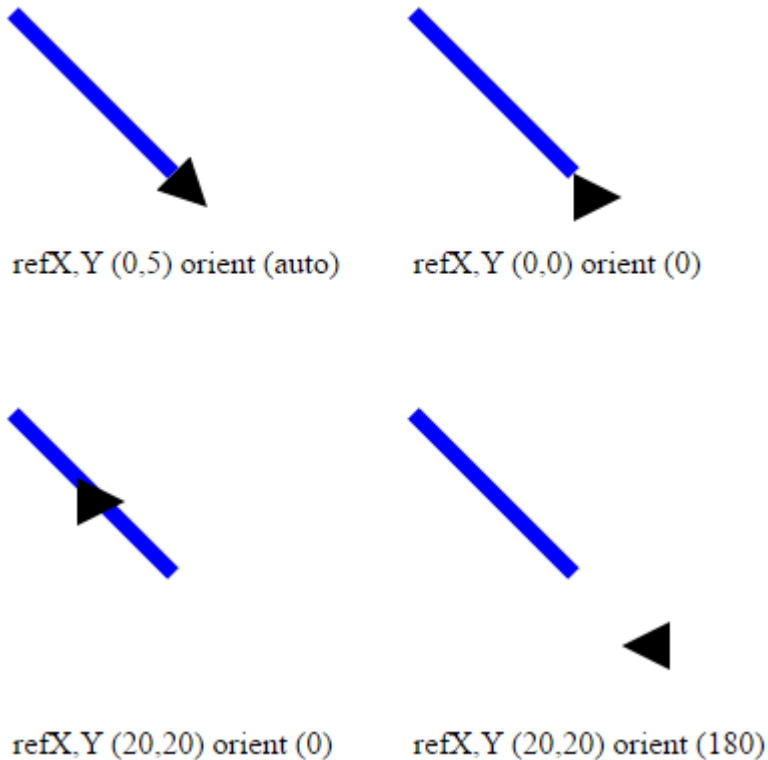
end="url(#marker3)" />

<text x="20" y="390"> refX,Y (20,20) orient (0) </text>

<line x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />

<text x="220" y="390"> refX,Y (20,20) orient (180) </text>
</svg>

```



## Эффекты альтернативных значений для маркеровUnits, markerWidth, markerHeight

По умолчанию для рисования маркеров используется ширина штриха вызывающего элемента, но вы можете явно указать, что маркеры будут рисоваться с использованием единичной системы для элемента, к которому применяется маркер, указав `markerUnits="userSpaceOnUse"`. Маркеры вставляются в поле 3x3 `markerUnits` (3 строковых интервала, если маркерные узлы не указаны). Но ширину и высоту окна можно явно указать с помощью `markerHeight` и `markerWidth`. Ниже приведены эффекты различных комбинаций `markerUnits`, `markerHeight` и `markerWidth`.

```

<svg width="800px" height="600px">
<defs>
  <marker id="marker1"
    viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
    markerWidth="1" markerHeight="1">
    <path d="M 0 0 L 10 5 L 0 10 z" />
    </marker>

  <marker id="marker2"

```

```

    viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="strokeWidth"
markerWidth="4" markerHeight="4">
    <path d="M 0 0 L 10 5 L 0 10 z" />
    </marker>

    <marker id="marker3"
viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="15" markerHeight="15">
    <path d="M 0 0 L 10 5 L 0 10 z" />
    </marker>

    <marker id="marker4"
viewBox="0 0 10 10" refX="0" refY="5" orient="auto" markerUnits="userSpaceOnUse"
markerWidth="30" markerHeight="30">
    <path d="M 0 0 L 10 5 L 0 10 z" />
    </marker>

</defs>

<line x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker1)" />
<text x="20" y="150"> markerUnits = strokeWidth </text>
<text x="20" y="170"> markerWidth|Height = 1 </text>

<line x1="220" y1="20" x2="300" y2="100" stroke-width="8" stroke="blue" marker-
end="url(#marker2)" />
<text x="250" y="150"> markerUnits = strokeWidth </text>
<text x="250" y="170"> markerWidth|Height = 4 </text>

<line x1="20" y1="220" x2="100" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker3)" />
<text x="20" y="390"> markerUnits = userSpaceOnUse </text>
<text x="20" y="410"> markerWidth|Height = 15 </text>

<line x1="220" y1="220" x2="300" y2="300" stroke-width="8" stroke="blue" marker-
end="url(#marker4)" />
<text x="250" y="390"> markerUnits = userSpaceOnUse </text>
<text x="250" y="410"> markerWidth|Height = 30 </text>
</svg>

```



`markerUnits = strokeWidth`  
`markerWidth|Height = 1`



`markerUnits = strokeWidth`  
`markerWidth|Height = 4`



`markerUnits = userSpaceOnUse`  
`markerWidth|Height = 15`



`markerUnits = userSpaceOnUse`  
`markerWidth|Height = 30`

## Стартовые, средние и конечные маркеры в линиях, полилинии, полигонах и элементах пути

Элементы могут указывать маркеры начала, середины и конца отдельно. Ниже приведены примеры начальных, средних и конечных маркеров для всех элементов, которые могут быть отмечены. Обратите внимание, что Chrome в настоящее время (июль 2016 г.) не рассчитывает автоматическую ориентацию для маркеров начала и конца для полигонов ([ошибка # 633012](#)), а также неправильно размещает средние маркеры для сегментов дуги ([ошибка # 583097](#))

```
<svg width="800px" height="600px">
<defs>
  <marker id="red-chevron"
    viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
    <path d="M 0 0 L 10 5 L 0 10" fill="none" stroke="red" />
  </marker>

  <marker id="black-arrow"
    viewBox="0 0 10 10" refX="0" refY="5" orient="auto">
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>

  <marker id="red-circle"
    viewBox="0 0 10 10" refX="5" refY="5" orient="auto" >
    <circle fill="red" cx="5" cy="5" r="5" />
  </marker>
</defs>
```



```

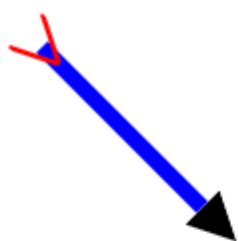
<line x1="20" y1="20" x2="100" y2="100" stroke-width="8" stroke="blue" marker-
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
<text x="20" y="150"> line: marker-mid not applied</text>

<polyline points="220,20 300,100 400,20" fill="none" stroke-width="8" stroke="blue" marker-
start="url(#red-chevron)" marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" />
<text x="250" y="150"> polyline </text>

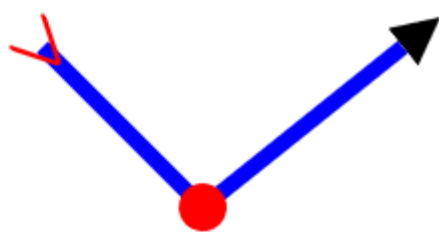
<polygon points="20,190 100,200 150,300 100,350 20,260" marker-start="url(#red-chevron)"
marker-end="url(#black-arrow)" marker-mid="url(#red-circle)" fill="none" stroke-width="5"
stroke="black" />
<text x="20" y="390"> polygon: end/start overlap </text>

<path d="M250,350 l 25,-25
a15,5 -16 0,1 10,-15 l 20,-5
a15,10 -16 0,1 10,-15 l 20,-5
a15,25 -16 0,1 10,-15 l 20,-5
a15,35 -16 0,1 10,-15 l 20,-5"
fill="none" stroke="green" stroke-width="2" marker-start="url(#red-chevron)" marker-
end="url(#black-arrow)" marker-mid="url(#red-circle)" />
<text x="250" y="390"> path with arc segments </text>
</svg>

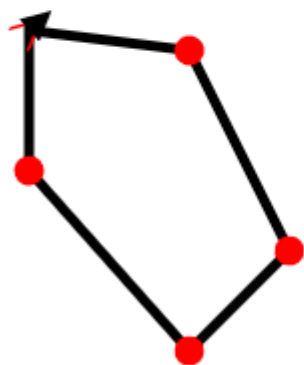
```



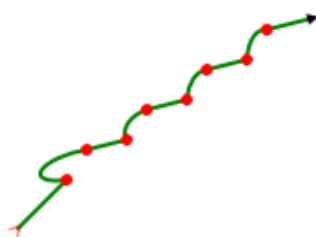
line: marker-mid not applied



polyline



polygon: end/start overlap



path with arc segments

Прочитайте маркер онлайн: <https://riptutorial.com/ru/svg/topic/4839/маркер>

# глава 12: маскировать

## Вступление

Элемент `mask` позволяет вам «закрепить» мягкими краями. Вы можете создавать маски из множества элементов, включая текст. Вся маска, белая, будет полностью непрозрачной. Все, что является черным, будет полностью прозрачным. Значения между белым и черным будут полупрозрачными.

## замечания

Помните, что маски - это дорогостоящая операция. Расчет должен выполняться для каждого пикселя в области маски. Поэтому держите область вашей маски как можно меньше.

## Examples

### основная маска

Зеленый прямоугольник с круглым отверстием посередине, показывающий фоновый фон.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <mask id="myMask">
    <rect x="0" y="0" width="100" height="100" fill="white"/>
    <circle cx="50" cy="50" r="45" fill="black"/>
  </mask>
  <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
  <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

### сложный пример с текстом и фигурами

Зеленый прямоугольник со сложной маской, наложенной на него, показывает фоновое изображение.

```
<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <mask id="myMask0">
    <circle cx="50" cy="50" r="30" fill="white"/>
  </mask>
  <mask id="myMask">
    <rect x="0" y="0" width="100" height="100" fill="white"/>
    <text x="5" y="60" font-size="40">Mask</text>
    <circle cx="50" cy="50" r="30" fill="black"/>
    <text x="5" y="60" font-size="40" mask="url(#myMask0)" fill="white">Mask</text>
  </mask>
  <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>
```

```

</mask>
<image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
<rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>

```

## полупрозрачность

зеленый прямоугольник (снова ...) с 4 отверстиями, созданными с использованием 4 оттенков серого, что приводит к 4 различным непрозрачности.

```

<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <mask id="myMask">
    <rect x="0" y="0" width="100" height="100" fill="white"/>
    <circle cx="25" cy="25" r="20" fill="black"/>
    <circle cx="75" cy="25" r="20" fill="#333"/>
    <circle cx="25" cy="75" r="20" fill="#666"/>
    <circle cx="75" cy="75" r="20" fill="#999"/>
  </mask>
  <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
  <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>

```

## маска с градиентом

Зеленый прямоугольник с отверстием посередине, с мягкими краями.

```

<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <radialGradient id="rg">
    <stop offset="0" stop-color="black"/>
    <stop offset="1" stop-color="white"/>
  </radialGradient>
  <mask id="myMask">
    <rect x="0" y="0" width="100" height="100" fill="white"/>
    <circle cx="50" cy="50" r="45" fill="url(#rg)"/>
  </mask>
  <image xlink:href="https://cdn.pixabay.com/photo/2013/04/06/05/06/ship-100978_960_720.jpg"
width="100" height="100"/>
  <rect x="0" y="0" width="100" height="100" fill="green" mask="url(#myMask)"/>
</svg>

```

Прочитайте маскировать онлайн: <https://riptutorial.com/ru/svg/topic/8143/маскировать>

---

# глава 13: переключатель

## замечания

`<switch>` - это атрибут условной обработки. Он не мешает элементам ссылаться на другие элементы. В нашем случае `<switch>` оценивает значение **systemLanguage** на своих прямых дочерних элементах, которые соответствуют языку пользователя. Как только будет найден, ребенок будет вынесен, а остальные дети будут обходить стороной.

Если **системный** язык не указан, будет отображаться **дочерний элемент**, что позволит нам указать **резервную копию**.

Связанные данные W3C

## Examples

### Альтернативный просмотр в зависимости от языка пользователя

```
<svg xmlns="http://www.w3.org/2000/svg">
  <switch>
    <text systemLanguage="en-UK" x="10" y="10">UK English</text>
    <text systemLanguage="fr" x="10" y="10">Français</text>
    <text systemLanguage="ru" x="10" y="10">Русский</text>
    <text x="10" y="20">English</text> <!-- fallback (if none of the languages match) -->
  </switch>
</svg>
```

Прочитайте переключатель онлайн: <https://riptutorial.com/ru/svg/topic/4702/переключатель>

# глава 14: преобразование

## замечания

Графические элементы могут быть изменены путем добавления атрибута *преобразования*.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="30" height="30" transform="translate(10, 10)" />
</svg>
```

Вместо того, чтобы левое верхнее начало отображалось в координатах (0, 0), оно будет показано вниз и вправо от точки (10, 10).

Целые группы элементов могут быть преобразованы вместе, а преобразования могут дополнять друг друга.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <g transform="rotate(45)">
    <rect x="0" y="0" width="30" height="30" />
    <circle cx="5" cy="5" r="5" transform="scale(3)" />
  </g>
</svg>
```

Во-первых, каждая точка круга будет масштабироваться с коэффициентом 3 относительно начала координат, доведя центр круга до середины прямоугольника в точке (15, 15) и его радиусе до 15. Затем прямоугольник и круг будет вращаться вместе на 45 градусов по часовой стрелке вокруг начала координат.

## Examples

### переведите

Переместите прямоугольник на 10 единиц вправо и на 20 единиц вниз:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="30" height="30" transform="translate(10, 20)" />
</svg>
```

Переместите его на 0 единиц по горизонтали и на 20 единиц вверх:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="20" width="30" height="30" transform="translate(0, -20)" />
</svg>
```

Переместите его на 10 единиц влево и 0 единиц по вертикали:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="10" y="0" width="30" height="30" transform="translate(-10)" />
</svg>
```

## масштаб

Масштабируйте прямоугольник горизонтально по коэффициенту 2 и по вертикали в 0,5:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="10" y="10" width="40" height="40" transform="scale(2, 0.5)" />
</svg>
```

### Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="20" y="5" width="80" height="20" />
</svg>
```

Зеркально прямоугольник горизонтально:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="20" height="40" transform="scale(-1, 1)" />
</svg>
```

Масштаб действует относительно источника, поэтому это эквивалентно

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="-20" y="0" width="20" height="40" />
</svg>
```

## вращаться

Поверните полигон по часовой стрелке на 90 градусов вокруг начала координат:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 15,20" transform="rotate(90)" />
</svg>
```

### Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 0,30 -20,15" />
</svg>
```

Центр вращения может быть задан явно:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 15,20" transform="rotate(90, 15, 15)" />
</svg>
```

## Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="30,0 30,30 10,15" />
</svg>
```

## skewX, skewY

перекосить многоугольник горизонтально на 45 градусов:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 30,30 0,30" transform="skewX(45)" />
</svg>
```

## Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 60,30 30,30" />
</svg>
```

значения x вычисляются как  $x + y * \tan(\text{angle})$ , значения y остаются неизменными

перекосить многоугольник вертикально на 30 градусов:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 30,30 0,30" transform="skewY(30)" />
</svg>
```

## Результат эквивалентен (допускает округление)

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,17.32 30,47.32 0,30" />
</svg>
```

значения x остаются неизменными, значения y вычисляются как  $y + x * \tan(\text{angle})$

## матрица

Примените матрицу преобразования к многоугольнику:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="0,0 30,0 30,30 0,30" transform="matrix(1,0.6,-1.2,1,40,10)" />
</svg>
```

Каждая точка (x, y) преобразуется путем применения матрицы (a, b, c, d, e, f) следующим образом:

$$\begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix} \cdot \begin{bmatrix} x_{\text{old}} \\ y_{\text{old}} \end{bmatrix} = \begin{bmatrix} x_{\text{old}} * a + y_{\text{old}} * c + e \\ x_{\text{old}} * b + y_{\text{old}} * d + f \end{bmatrix}$$

## Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <polygon points="40,10 70,28 34,58 4,40" />
</svg>
```

## Множественные преобразования

Преобразования могут быть объединены и применены **справа налево**

Поверните прямоугольник на 90 градусов, а затем переместите его на 20 единиц и направо на 20 единиц:

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="-10" y="-20" width="20" height="40"
    transform="translate(20 20) rotate(90)" />
</svg>
```

## Результат эквивалентен

```
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="10" width="40" height="20" />
</svg>
```

Прочитайте преобразование онлайн: <https://riptutorial.com/ru/svg/topic/3249/преобразование>



---

# глава 15: преобразование

## Синтаксис

- `transform = " [functions] * "`
- переводить (x [, y])
- вращаться (θ [x, y])
- Шкала (x [, y])
- `skewX (θ)`
- `skewY (θ)`
- Матрица (a, б, в, г, д, е)

## Examples

### Применение преобразований

Преобразования могут применяться к элементам путем добавления атрибута `transform` :

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

Или группам элементов, заключенных в теги `<g>` :

```
<g transform="translate(50,50)">
<circle cx="0" cy="0" r="50"/>
<circle cx="0" cy="0" r="25" fill="white"/>
</g>
```

Дополнительные преобразования могут быть применены к одному и тому же элементу, добавив их в разделенные пространства:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50) scale(.5)"/>
```

### Функции преобразования

---

## Переведите

`translate` перемещает графики по указанным векторам:

```
<circle cx="0" cy="0" r="50" transform="translate(50,50)"/>
```

Первое значение - это x-перевод, а второе - y. Если y опущено, по умолчанию будет 0.

# Масштаб

`scale` изменяет размеры элементов по заданным коэффициентам:

```
<circle cx="50" cy="50" r="25" transform="scale(.5,2)"/>
```

Как и `translate`, аргументами являются `x`, то `y`. Однако в `scale`, если `y` опущен, по умолчанию будет установлено значение `x`; другими словами, он масштабирует элемент без изменения соотношения сторон.

---

# Поворот

`rotate` поворачивает элементы по заданным углам.

```
<!-- <rect> used for this example because circles can't be rotated -->  
<rect width="100" height="5" transform="rotate(90,50,50)"/>
```

Первое значение - это угол в градусах. Преобразование применяется по часовой стрелке. Остальные два значения представляют собой точку, которая должна быть повернута вокруг, по умолчанию - в начале координат.

Прочитайте преобразование онлайн: <https://riptutorial.com/ru/svg/topic/7100/преобразование>

# глава 16: Прямоугольник

## параметры

атрибут	Описание
Икс	Горизонтальное положение прямоугольника с левого края.
У	Вертикальное положение прямоугольника с верхнего края.
ширина	Ширина прямоугольника.
рост	Высота прямоугольника.
гх	Горизонтальный радиус эллипса, используемый для округлых углов прямоугольника
чень	Вертикальный радиус эллипса, используемый для круглых углов прямоугольника
Инсульт	Цвет прямоугольника.
Ход ширина	Ширина границы прямоугольника.
заполнить	Цвет <i>внутри</i> границы прямоугольника.

## замечания

Подробную информацию о элементе SVG «rect» можно найти в [Рекомендации W3C для SVG](#).

## Examples

### Нарисуйте черный прямоугольник без заливки

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="10" y="10" width="50" height="100" stroke="black" stroke-width="5" fill="none" />
</svg>
```

Результат:



## Нарисуйте черный прямоугольник с желтым заполнителем и закругленными углами

- Атрибуты `width` и `height` обозначают размеры прямоугольника. Эти значения находятся в пикселях по умолчанию
- Значение `fill` задает цвет для прямоугольника. Если значение для `fill` не указано, по умолчанию используется черный

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="10" y="10" width="50" height="100" rx="10" ry="10" stroke="black" stroke-
width="5" fill="yellow" />
</svg>
```

Результат:



Прочитайте Прямоугольник онлайн: <https://riptutorial.com/ru/svg/topic/2993/прямоугольник>

# глава 17: пути

## Вступление

Пути - самый гибкий элемент SVG. Путь представляет собой ряд кубических или квадратичных кривых Безье, расположенных в соединенных сплайнах. Путь может быть открытым или закрытым в петлю, или он может быть сложным с несколькими подкомпонентами. Если путь не является простым, правило заполнения важно при определении того, какие области находятся внутри или вне пути.

Пути обычно генерируются автоматическими редакторами. Обычно для шрифтов используются квадратичные пути и кубические пути для иллюстраций.

## параметры

Атрибуты / параметры	Описание
d	Определяет последовательность команд рисования, которые создают форму. например d = "M 50,60 L50,60". Команды рисования в верхнем регистре обозначают абсолютные координаты. Строчные команды рисования обозначают относительные координаты.
(...)	<b>Команды рисования</b>
м / M	Переместить текущее положение чертежа в положение XY d = "M XY"
л / л	Нарисуйте линию на X, Y d = "L XY"
V / V	Нарисуйте вертикальную линию до Y d = "V Y"
ч / H	Нарисуйте горизонтальную линию до X d = "H X"
а / A	Нарисуйте дугу на X, Y с подразумеваемым радиусом Rx и Ry и поворот, заданный вращением по оси X. Большие дуги и флаги развертки выбирают, какая из 4 возможных дуг, которые удовлетворяют этим ограничениям, должны быть нарисованы. d = «A Rx Ry Ось-поворот оси (градусы) флаг флага (0/1) большой дуги (0/1) X, Y".
Q / Q	Нарисуйте квадратичную кривую Безье на X, Y, используя контрольную точку X1Y1 d = "X1Y1 X Y"

Атрибуты / параметры	Описание
T / T	Нарисуйте сокращенную квадратичную кривую безье (контрольная точка вычисляется как отражение контрольной точки предыдущей команды рисования q / Q через текущую позицию чертежа)
C / C	Нарисуйте кубическую кривую безье для X, Y, используя контрольные точки X1, Y1 и X2, Y2 d = "C X1Y1, X2Y2, XY"
S / S	Нарисуйте сокращенную кубическую беззерновую кривую (первая контрольная точка рассчитывается как отражение второй контрольной точки предыдущей команды рисования c / C через текущую позицию чертежа).
- z / Z	Закройте путь, вычерчивая строку для начала пути (или pathsegment, если другой z использовался ранее)
(...)	(конец списка)
длина пути	(Необязательно) Позволяет автору указать номинальную длину пути, которая будет использоваться для калибровки в других вычислениях, например, для текста вдоль пути
<b>Параметры обводки</b>	<i>общий среди всех элементов формы и рисунка</i>
Инсульт	Цвет пути
Ход ширина	Ширина пути

## замечания

Подробную информацию о элементе `path` SVG можно найти в [Рекомендации W3C для SVG](#).

## Examples

**Нарисуйте диагональную синюю линию с помощью команды L path**

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <path d="M 10,10 L 100,50" stroke="blue" stroke-width="5" />
</svg>
```

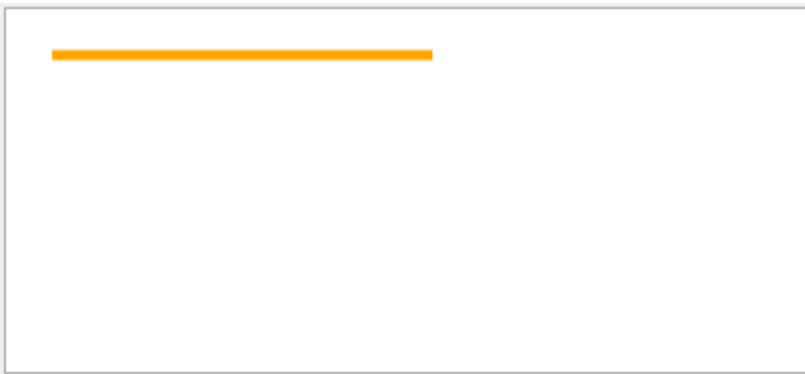
Результат:



**Нарисуйте горизонтальную оранжевую линию, используя команду рисования H**

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <path d="M 10,10 H 200" stroke="orange" stroke-width="5" />
</svg>
```

**Результат:**



**Нарисуйте красный крест, используя команды пути l (относительная линия)**

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <path d="M 10,10 l 90,90 M 100,10 l -90,90" stroke="red" stroke-width="10" />
</svg>
```

**Результат:**



## Нарисуйте вертикальную зеленую линию, используя команду V path

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">  
  <path d="M 10,10 V 200" stroke="green" stroke-width="5" />  
</svg>
```

Результат:



Прочитайте пути онлайн: <https://riptutorial.com/ru/svg/topic/2397/пути>



---

# глава 18: Создание шрифтов

## Вступление

Браузеры больше не поддерживаются шрифтами SVG. Тем не менее они очень удобны для программного создания шрифтов, таких как шрифты символов или шрифты штрих-кода. Существует множество инструментов, которые позволяют конвертировать SVG-шрифты в любой другой формат шрифта.

## замечания

Ниже приведен список инструментов, которые вы можете использовать с шрифтами SVG.

---

## Преобразователи

- <https://github.com/fontello/svg2ttf>

## Examples

### простой шрифт

Простой пример шрифта svg. Несколько вещей, чтобы отметить здесь:

- система координат глифов находится в противоречии с обычной системой координат в svg. **ось y указывает вверх**. Точка 0,0 находится в нижнем правом углу.
- Все дорожки в шрифте должны быть нарисованы против часовой стрелки.
- В большинстве инструментов поддерживается только атрибут d элемента глифа. Элементы ребенка не будут работать, хотя они технически разрешены.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <font id = "myFont"
    horiz-adv-x = "1000"
    vert-origin-x = "0"
    vert-origin-y = "0" >
    <font-face font-family = "myFont"
      font-weight = "normal"
      units-per-em = "1000">
      <font-face-src>
        <font-face-name name="myFont" />
      </font-face-src>
    </font-face>
    <glyph unicode="a" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
    <glyph unicode="b" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
  </font>
</svg>
```

Если у вас более широкие или более узкие глифы, просто измените `horiz-adv-x` на сам элемент глифа.

```
<glyph unicode="a" horiz-adv-x="512" d="M0 0 H1000 L500 1000z M200 200 L500 800 L800 200z" />
```

## подбор шрифтов

свойство `unicode` используется для последующего выбора глифов. Вы можете использовать простые буквы или кодовые страницы `unicode`, а также лигатуры (сочетание букв или кодов Unicode)

- `unicode="abc"`
- `unicode="&#97;&#98;"`
- `unicode="ab&#97;&#98;"`
- `unicode="a"`
- `unicode="&#98;"`

**глифы всегда выбираются с помощью первого совпадения, поэтому имеют все лигатуры перед любым символом.**

`unicode codepoints` могут быть записаны в decimal `&#123;` или в шестнадцатеричном `&#x1f`.

## восхождение, спуск и базовый уровень

свойство `units-per-em` является одним из самых важных свойств шрифта. Он используется, чтобы придавать любому значению любого другого свойства какое-либо значение.

спецификация шрифта CSS2 имеет хорошее [определение em square](#) :

Определенные значения, такие как метрики ширины, выражаются в единицах, которые относятся к абстрактному квадрату, высота которого представляет собой предполагаемое расстояние между строками типа одного размера

**базовая линия по умолчанию равна 0** в `em`-квадрате. для расчета высоты линии и выравнивания цели восхождения и спуска двух первостепенной важны.

Восхождение - это максимальное расстояние от базовой линии до самой высокой точки вашего самого большого глифа. Обычно это `1em`, поэтому значение, которое вы дали для единиц `po-em`.

Спуск - это максимальное расстояние от базовой линии до самой низкой точки в любом знаке вашего шрифта.

Вот шрифт с глифами, отображающий линию на самой низкой и самой высокой точке, а также на базовой линии.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1000 1000">
```

```
<font id = "myFont"
  horiz-adv-x    = "1000"
  vert-origin-x  = "0"
  vert-origin-y  = "0" >
  <font-face font-family = "myFont"
    font-weight = "normal"
    units-per-em = "1000"
    descent="500"
    ascent="1000">
    <font-face-src>
      <font-face-name name="myFont" />
    </font-face-src>
  </font-face>`
  <glyph unicode = "a" d = "M0 900h1000v100h-1000z" />
  <glyph unicode = "b" d = "M0 0h1000v100h-1000z" />
  <glyph unicode = "c" d = "M0 -500h1000v100h-1000z" />
</font>
</svg>
```

Подъем и спуск используются для определения высоты линии. Единицы по умолчанию и базовая линия используются для определения местоположения и размера по сравнению с другими используемыми шрифтами.

Прочитайте Создание шрифтов онлайн: <https://riptutorial.com/ru/svg/topic/8147/создание-шрифтов>

## глава 19: Текст

### параметры

<text>	подробности
Икс	Позиция x текста.
У	Позиция y текста.
дх	Относительный сдвиг в позиции x.
ду	Относительный сдвиг в позиции y.
вращаться	Определяет угловое смещение для текстовых символов.
textLength	Подходит к тексту в заданную длину.
lengthAdjust	Указывает, сжаты / растянуты ли кернинг или кернинг и глифы в соответствии с текстом в указанную текстовую длину. Значения: интервал или интервалAndGlyphs
-	<b>Параметры, общие для всех элементов текстового фрагмента (текст, tref, textPath, tspan)</b>
текст-якорь	Указывает горизонтальное выравнивание. Значения: начало, посередине, конец.
смещение базовой линии	Сдвигает текстовую базовую линию на основе любых значений, предоставляемых таблицей шрифтов для надстрочного или индексного позиционирования (sub, super) или положительным или отрицательным% или длиной. Значения: sub, super,% или length.

### замечания

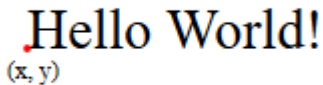
базовый сдвиг не поддерживается самыми современными версиями браузеров Firefox и Microsoft по состоянию на июль 2016 года.

## Examples

### Рисовать текст

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="40" y="60" font-size="28">Hello World!</text>
</svg>
```

Координаты x и y задают положение нижнего левого угла текста (если только текстовый якорь не был изменен).



## Супер- и индекс

Используя параметр базового сдвига, вы можете указать супер- или индекс. Но это не поддерживается всеми основными браузерами.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="20">x<tspan baseline-shift="super">2</tspan></text>
  <text x="10" y="60">f<tspan baseline-shift="sub">x</tspan></text>
</svg>
```

Для кросс-браузерного решения вы можете использовать dy, dx и относительный размер шрифта.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="40">x<tspan dy="-7" font-size=".7em">2</tspan></text>
  <text x="10" y="80">f<tspan dy="3" font-size=".7em">x</tspan></text>
</svg>
```

## Повернуть текст

Свойство rotate поворачивает каждый символ на указанный угол.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="20" rotate="30">Each character is rotated</text>
</svg>
```

Чтобы повернуть весь текстовый элемент, вы должны использовать свойство transform.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <text transform="translate(10, 60) rotate(30)">The whole text is rotated</text>
</svg>
```

## Индивидуальное позиционирование букв с массивами значений X и Y

```
<svg width="400px" height="200px">
```

```
<text x="1em, 2em, 3em, 4em, 5em" y="3em, 4em, 5em">  
  Individually Spaced Text  
</text>  
</svg>
```

Элемент Text поддерживает индивидуальное размещение букв, принимая массив значений для x и y.

Прочитайте Текст онлайн: <https://riptutorial.com/ru/svg/topic/3033/текст>

## глава 20: Узоры

### параметры

параметр	описание
patternUnits	система координат шаблона атрибутов либо objectBoundingBox (по умолчанию), либо userSpaceOnUse
patternContentUnits	система координат содержимого шаблона либо objectBoundingBox, либо userSpaceOnUse (по умолчанию)
patternTransform	преобразование, применяемое к содержимому шаблона
Икс	смещение x шаблона (по умолчанию - ноль)
У	смещение y шаблона (по умолчанию - ноль)
ширина	ширина рисунка (требуется)
рост	высота рисунка (обязательно)
XLink: HREF	ссылка на другой шаблон, который предоставляет некоторые атрибуты или контент
preserveAspectRatio	следует ли сохранить соотношение сторон рисунка

### замечания

По умолчанию шаблон будет черепицей, установив середину блока шаблонов в верхнем левом углу фигуры.

### Examples

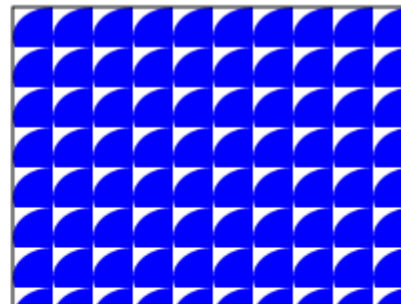
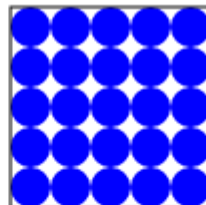
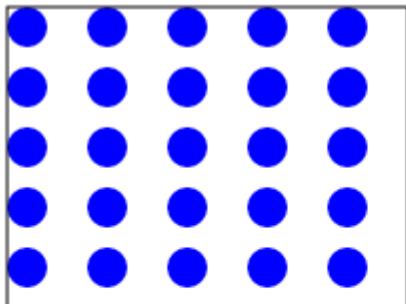
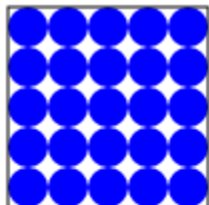
#### Пример шаблона с единицами objectBoundingBox

```
<svg width="400" height="400">
<defs>
  <pattern id="pattern1" width="0.2" height="0.2" patternUnits="objectBoundingBox">
    <circle cx="10" cy="10" r="10" fill="#0000ff" />
  </pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)" />
</svg>
```

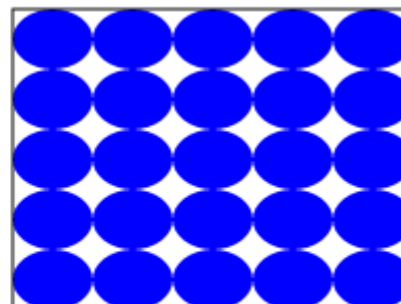
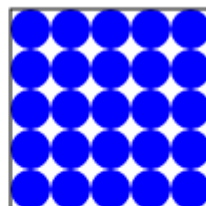
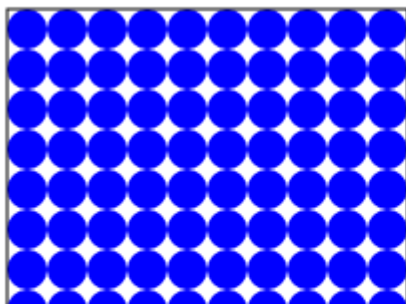
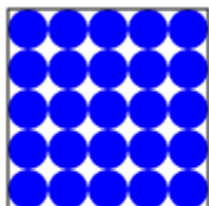
## Покрывание шаблонов с комбинациями patternUnits и patternContentUnits

Шаблоны SVG ведут себя существенно иначе, чем фоновые изображения CSS при заполнении эквивалентных фигур. Это может привести к большим неожиданностям для новых пользователей SVG. Ниже приведены примеры шаблона, определенного во всех возможных комбинациях patternUnits и patternContentUnits - показывая, как эти параметры влияют на поведение заполнения.



patternUnits="objectBoundingBox" (20% of shape)  
patternContentUnits="userSpaceOnUse" (20px circle)  
(Units used by default)

patternUnits="userSpaceOnUse" (10px square box)  
patternContentUnits="objectBoundingBox" (radius=



patternUnits="userSpaceOnUse" (10px square box)  
patternContentUnits="userSpaceOnUse" (20px circle)

patternUnits="objectBoundingBox" (20% of shape)  
patternContentUnits="objectBoundingBox" (radius=

```
<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2" patternUnits="objectBoundingBox"
patternContentUnits="userSpaceOnUse">
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>

  <pattern id="pattern2" x="10" y="10" width="20" height="20" patternUnits="userSpaceOnUse"
patternContentUnits="objectBoundingBox">
  <circle cx=".1" cy=".1" r="0.1" fill="blue" />
</pattern>

  <pattern id="pattern3" x="10" y="10" width="20" height="20" patternUnits="userSpaceOnUse"
patternContentUnits="userSpaceOnUse">
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>
```



```

</pattern>

<pattern id="pattern4" x="0" y="0" width="0.2" height="0.2"
patternUnits="objectBoundingBox" patternContentUnits="objectBoundingBox">
  <circle cx=".1" cy=".1" r="0.1" fill="blue" />
</pattern>
</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)" />
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)" />
  <text x="10" y="200">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="10" y="220">patternContentUnits="userSpaceOnUse" (20px circle) </text>
  <text x="10" y="240" stroke="blue" stroke-width="1">(Units used by default)</text>

<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)" />
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)" />
  <text x="10" y="500">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="10" y="520">patternContentUnits="userSpaceOnUse" (20px circle) </text>

<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)" />
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)" />
  <text x="410" y="200">patternUnits="userSpaceOnUse" (10px square box)</text>
  <text x="410" y="220">patternContentUnits="objectBoundingBox" (radius="10%") </text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)" />
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)" />
  <text x="410" y="500">patternUnits="objectBoundingBox" (20% of shape)</text>
  <text x="410" y="520">patternContentUnits="objectBoundingBox" (radius="10%") </text>

</svg>

```

## Примеры шаблонов шаблона

```

<svg width="800px" height="800px">
<defs>
<pattern id="pattern1" x="0" y="0" width="0.2" height="0.2" >
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>

<pattern id="pattern2" x="0" y="0" width="0.2" height="0.2" patternTransform="scale(1.5)">
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>

<pattern id="pattern3" x="0" y="0" width="0.2" height="0.2" patternTransform="skewX(45)">
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>

<pattern id="pattern4" x="0" y="0" width="0.2" height="0.2" patternTransform="matrix(1.5,-
.70,.10,1.1,-30,10)">
  <circle cx="10" cy="10" r="10" fill="blue" />
</pattern>

</defs>

<rect x="10" y="10" width="100" height="100" stroke="black" fill="url(#pattern1)" />
<rect x="150" y="10" width="200" height="150" stroke="black" fill="url(#pattern1)" />
  <text x="10" y="200">Original</text>

```

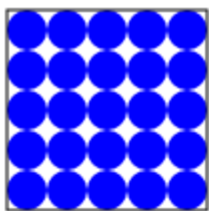
```

<rect x="410" y="10" width="100" height="100" stroke="black" fill="url(#pattern2)"/>
<rect x="550" y="10" width="200" height="150" stroke="black" fill="url(#pattern2)"/>
<text x="410" y="200">patternTransform="scale(1.5)"</text>

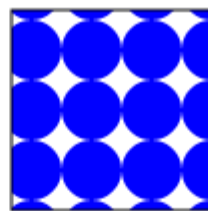
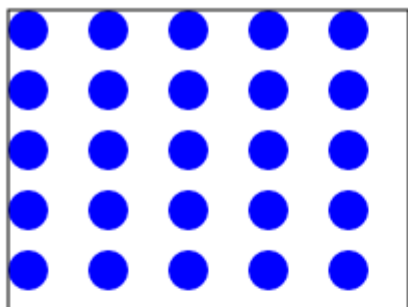
<rect x="10" y="310" width="100" height="100" stroke="black" fill="url(#pattern3)"/>
<rect x="150" y="310" width="200" height="150" stroke="black" fill="url(#pattern3)"/>
<text x="10" y="500">patternTransform="skewX(45)"</text>

<rect x="410" y="310" width="100" height="100" stroke="black" fill="url(#pattern4)"/>
<rect x="550" y="310" width="200" height="150" stroke="black" fill="url(#pattern4)"/>
<text x="410" y="500">patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"</text>
</svg>

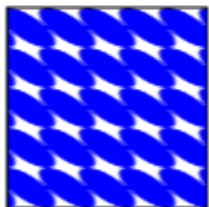
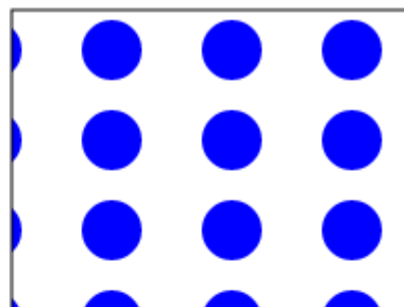
```



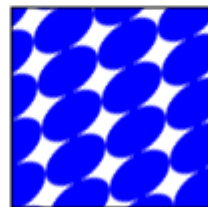
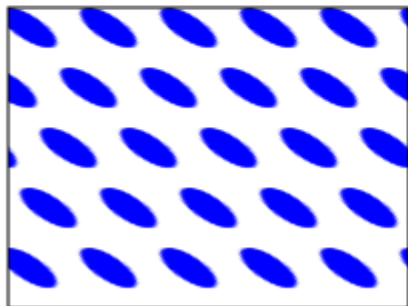
Original



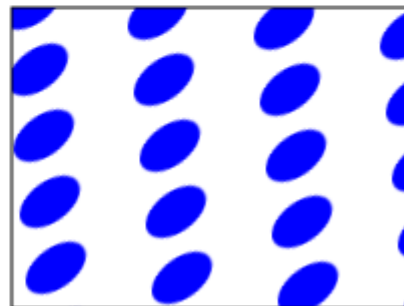
patternTransform="scale(1.5)"



patternTransform="skewX(45)"



patternUnits="matrix(1.5,-.70,.10,1.1,-30,10)"



Прочитайте Узоры онлайн: <https://riptutorial.com/ru/svg/topic/3251/узоры>

# глава 21: указатель событий

## Вступление

С свойством `pointer-events` вы можете контролировать, какая часть вашего рисунка будет реагировать на события указателя.

## Examples

### НИКТО

наиболее распространенным случаем является установка указателей-событий на `none` чтобы предотвратить определенные фигуры или весь ваш рисунок для захвата событий мыши и позволить фигурам под ними получать события.

Если вы наведете указатель мыши на область, где красный круг накладывается на синий круг, синий круг все равно будет принимать события мыши, так как указатели-события не равны `none`

```
<svg viewBox="0 0 150 100">
  <style>
    .target:hover{fill:green}
  </style>
  <circle class="target" cx="50" cy="50" r="50" fill="blue"/>
  <circle cx="100" cy="50" r="50" fill="red" pointer-events="none"/>
</svg>
```

### заполнить

Установка `pointer-events="fill"` позволяет получать события мыши на фигуре, даже если для ее заполнения установлено значение `none`

```
<svg viewBox="0 0 100 100">
  <style>
    circle:hover{fill:green}
  </style>
  <circle class="target" cx="50" cy="50" r="50" fill="none"/>
</svg>
```

Прочитайте указатель событий онлайн: <https://riptutorial.com/ru/svg/topic/8166/указатель-событий>

# глава 22: фильтры

## Синтаксис

- Объявление фильтра: `<filter id="filter-id" > ... список дочерних примитивов ... </filter>`
- Применить фильтр через атрибут SVG: `<elementname filter="url(#filter-id)" ... />`
- Применить фильтр через свойство CSS: `(-prefix- ) filter: url ("# filter-id");`

## параметры

Элемент атрибутов	подробности
Область фильтра	Фильтрующий элемент может дополнительно определять положение, размеры, разрешение и единицы для вывода эффекта фильтра. Положение и размеры фильтра могут быть заданы с использованием следующих параметров: x, y, width, height. Значения по умолчанию <i>не</i> являются <i>интуитивными</i> и составляют: x: -10% y: -10% width: 120% height: 120%
Разрешение фильтра	<code>filterRes</code> является необязательным атрибутом в SVG 1.1, который может использоваться для указания разрешения, при котором фильтр обрабатывается. Этот атрибут имел неравномерную поддержку и теперь устарел в текущих браузерах.
Блоки фильтров	По умолчанию единицы и система координат для области эффектов фильтра (x, y, width, height) фильтрующего элемента установлены относительно ограничивающего блока элемента, ссылающегося на фильтр. В терминах SVG это называется «objectBoundingBox». Когда мы пишем x = «50%», это означает «установить начальную позицию x области фильтра в левой части ограничивающего блока ссылочного элемента + 50% от ширины элемента». Но вы также можете указать единицы и координаты явно, установив свойство <code>filterUnits</code> . Этими двумя альтернативами являются «objectBoundingBox» (по умолчанию мы только что описали) или «userSpaceOnUse». <code>userSpaceOnUse</code> устанавливает единицы фильтра и систему координат в холст элемента ссылки или в выражениях SVG - «userSpaceOnUse».
Примитивные единицы	В дополнение к единичной системе для самого фильтра вы также можете указать систему единиц, что примитивы дочерних фильтров фильтра будут использовать через атрибут <code>primitiveUnits</code> . Еще раз,

Элемент атрибутов	подробности
	выбор между <code>userSpaceOnUse</code> и <code>objectBoundingBox</code> . Они влияют на координаты 0,0 и значения единиц для примитивов фильтра так же, как и для фильтров.
Цветовое пространство	Цветовое пространство по умолчанию для SVG-фильтров - <code>linearRGB</code> . Дополнительный атрибут <code>color-interpolation-filters</code> может быть установлен в <code>sRGB</code> чтобы изменить цветовое пространство на более традиционное пространство <code>sRGB</code> .

## замечания

Большинство атрибутов фильтра могут быть доступны с помощью элемента `<animate>` `animate <animate>`, хотя вы должны использовать библиотеку «fakeSMIL» в IE для достижения тех же результатов. Анимация SMIL (элемент `<animate>`) будет устаревать в пользу новой спецификации Web Animations, которая имеет очень ограниченную поддержку по состоянию на середину 2016 года.

Элементы дочернего элемента `Filter` - примитивы фильтров - имеют два необязательных атрибута, которые определяют цветовое пространство, в котором выполняются вычисления цветовой интерполяции: цветовая интерполяция и цвето-интерполяционные фильтры. По умолчанию для первого используется `sRGB`, а по умолчанию для последнего - `linearRGB`. Манипуляции, которые инвертируют цветовое пространство (через `feColorMatrix` или `feComponentTransfer`), могут привести к неинтуитивным результатам - для тех, которые используются в цветовом пространстве CSS `sRGB`. Например, инверсия цвета изображения в оттенках серого в линейном RGB приведет к выраженному смещению в сторону более белых тонов. Это можно исправить, установив значение примитива в `sRGB`. Эти атрибуты могут быть установлены в отдельных примитивах фильтра или унаследованы от самого элемента фильтра.

Если никакой другой вход не указан, но требуется один, первый примитив фильтра внутри фильтра примет растрированную (растровую) версию ссылочного элемента в качестве своего ввода. Последующие примитивы фильтров, ожидающие ввода, будут принимать результат непосредственно предшествующего примитива фильтра в качестве входных данных.

В сложных фильтрах может возникнуть трудность отслеживать входы и выходы (и отлаживать), если они остаются скрытыми; и это хорошая практика для явного объявления входов и выходов для каждого примитива.

---

**Элементы фильтров SVG могут быть отдельно разделены на входы,**

**преобразования, световые эффекты и комбинации.**

### **Входы:**

feFlood: генерирует цветное поле

feTurbulence: генерирует широкий спектр шумовых эффектов

feImage: генерирует изображение из внешней ссылки на изображение, URI данных или ссылку на объект (ссылки на объекты не поддерживаются в Firefox по состоянию на середину декабря12)

### **Трансформации:**

feColorMatrix: преобразует входные значения пикселя RGBA в выходные значения

feComponentTransfer: регулирует цветовую схему отдельного цветного канала

feConvolveMatrix: заменяет каждый пиксель новым пикселем, вычисленным из значений пикселей в области относительно текущего пикселя)

feGaussianBlur: заменяет текущий пиксель средневзвешенным значением пикселей в области вокруг пикселя

feDisplacementMap: перемещает каждый пиксель из текущей позиции на основе значений R, G или B из другой входной графики.

feMorphology: заменяет каждый пиксель новым пикселем, вычисленным из максимального или минимального значения всех пикселей в прямоугольной области вокруг этого пикселя.

feOffset: перемещает вход с текущего положения

### **Эффекты освещения:**

feSpecularLighting: обеспечивает «блестящий» эффект 2D или псевдо-3D освещения

feDiffuseLighting: обеспечивает эффект «матового» 2D или псевдо-3D освещения

feDistantLight: обеспечивает отдаленный источник света для зеркального или диффузного освещения

feSpotLight: обеспечивает источник света конической секции для зеркального или диффузного освещения

fePointLight: обеспечивает точечный источник света для зеркального или диффузного освещения

### **Комбинации:**

**feMerge:** создает простой составной состав из нескольких входов (включая предыдущие входы фильтра)

**feBlend:** смешивает несколько входов с использованием правил смешивания

**feComposite:** объединяет несколько входов с использованием правил набора комбинаций с учетом альфа-значений.

**feTile:** ввод элементов для создания повторяющегося шаблона

---

## Другие примечания

Хотя SVG - это технология векторной графики, важно подчеркнуть, что фильтры SVG выполняют операции на *уровне пикселей* на всех входах (включая формы SVG) и создают растрингованные (растровые) выходы на определенном уровне разрешения. Применение преобразования масштаба 10x (например) на простой кривой SVG, которая была отфильтрована при нормальном разрешении экрана, будет создавать пиксельные края, поскольку сглаживание исходного изображения было преобразовано в пиксели с помощью фильтра и увеличено. (Неясно, соответствует ли это спецификации или просто ограничению текущих реализаций)

Помните, что SVG является XML, когда вы пишете фильтры, поэтому все теги должны быть закрыты, и многие свойства и атрибуты должны быть указаны явно или фильтр не будет выполняться.

Фильтрующий элемент никогда не отображается напрямую. Он ссылается только на свойство фильтра на элемент, к которому применяется фильтр. Обратите внимание, что свойство `display` не применяется к элементу фильтра, и элементы не отображаются непосредственно, даже если для свойства `display` установлено значение, отличное от «none». И наоборот, фильтрующие элементы доступны для ссылок даже тогда, когда свойство `display` на элементе `filter` или любом из его предков установлено на «none».

На фильтры SVG можно ссылаться через CSS-фильтр, хотя по состоянию на середину 2016 года через этот механизм поддерживается только подмножество примитивов, и этот механизм не поддерживается в браузерах Microsoft.

## Examples

### Фильтры размытия: **feGaussian Blur** (базовый)

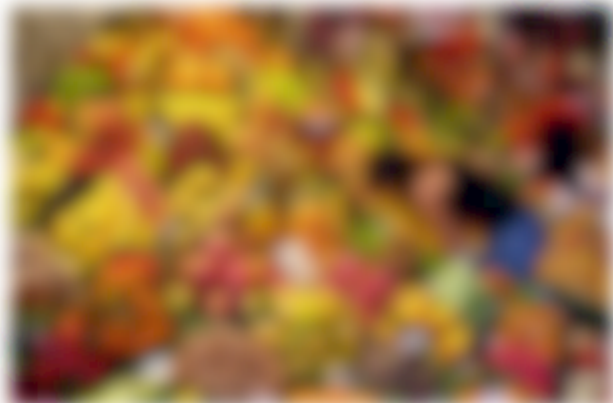
```
<svg width="900px" height="400px" viewBox="0 0 900 400">
  <defs>
    <filter id="basicGaussian">
      <feGaussianBlur stdDeviation="5"/>
    </filter>
  </defs>
```



```

<image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
  <image filter="url(#basicGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```



( Исходное изображение Daderot в Викискладе)

## Фильтры размытия: feGaussianBlur (размытие по оси x и по оси Y отдельно)

```

<svg width="900px" height="400px" viewBox="0 0 900 400">
  <defs>
    <filter id="xAxisGaussian">
      <feGaussianBlur stdDeviation="5 0"/>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#xAxisGaussian)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```



( Исходное изображение Daderot в Викискладе)



## Фильтры размытия: feGaussianBlur с жесткими краями и непрозрачностью 100%

```
<svg width="900px" height="400px" viewBox="900 400">
  <defs>
    <filter id="GaussianHardEdge" x="0%" y="0%" width="100%" height="100%">
      <feGaussianBlur stdDeviation="5"/>
      <feComponentTransfer>
        <feFuncA type="table" tableValues="1 1"/>
      </feComponentTransfer>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```



( Исходное изображение [Daderot](#) в Викискладе)

## Фильтры размытия: размытие ящика

```
<svg width="900px" height="400px" viewBox="900 400">
  <defs>
    <filter id="GaussianHardEdge" >
      <feConvolveMatrix order="3" kernelMatrix=" 1 1 1
                                                1 1 1
                                                1 1 1"/>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#GaussianHardEdge)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```



( Исходное изображение Daderot в Викискладе)

## Фильтры размытия: Bokeh Blur (3 слоя, обрезанные)

```
<svg width="900px" height="400px" viewBox="0 0 900 400">
  <defs>
    <filter id="BokehBlur" color-interpolation-filters="sRGB">
      <feGaussianBlur stdDeviation="2" result="blurSource"/>
      <feColorMatrix type="luminanceToAlpha"/>
      <feComponentTransfer result="brightness-mask" >
        <feFuncA type="discrete" tableValues="0 0 0 1 1"/>
      </feComponentTransfer>

      <!--bokeh Layer 1 -->
      <feTurbulence type="fractalNoise" seed="1" baseFrequency=".67" numOctaves="3"/>
      <feColorMatrix type="luminanceToAlpha"/>
      <feComponentTransfer>
        <feFuncA type="discrete" tableValues="0 0 0 1"/>
      </feComponentTransfer>
      <feComposite operator="in" in="brightness-mask"/>
      <feComposite operator="in" in="blurSource"/>

      <feMorphology operator="dilate" radius="5"/>
      <feGaussianBlur stdDeviation="8"/>
      <feColorMatrix type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0
        0 0 0 9 0" />
      <feComponentTransfer result="bokeh1">
        <feFuncA type="linear" slope=".5" />
      </feComponentTransfer>

      <!--bokeh Layer 2 -->
      <feTurbulence type="fractalNoise" seed="49" baseFrequency=".67" numOctaves="3"/>
      <feColorMatrix type="luminanceToAlpha"/>
      <feComponentTransfer>
        <feFuncA type="discrete" tableValues="0 0 0 1"/>
      </feComponentTransfer>
      <feComposite operator="in" in="brightness-mask"/>
      <feComposite operator="in" in="blurSource"/>

      <feMorphology operator="dilate" radius="10"/>
      <feGaussianBlur stdDeviation="12"/>
      <feColorMatrix type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0
        0 0 0 15 0" />
      <feComponentTransfer result="bokeh2">
```

```

        <feFuncA type="linear" slope=".3" />
    </feComponentTransfer>

    <!--bokeh Layer 3 -->

    <feTurbulence type="fractalNoise" seed="44" baseFrequency=".67" numOctaves="3"/>
    <feColorMatrix type="luminanceToAlpha"/>
        <feComponentTransfer>
            <feFuncA type="discrete" tableValues="0 0 0 1"/>
        </feComponentTransfer>
        <feComposite operator="in" in="brightness-mask"/>
        <feComposite operator="in" in="blurSource"/>

        <feMorphology operator="dilate" radius="10"/>
        <feGaussianBlur stdDeviation="18"/>
        <feColorMatrix type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0
                                           0 0 0 15 0" />

        <feComponentTransfer result="bokeh3">
            <feFuncA type="linear" slope=".2" />
        </feComponentTransfer>

    <!--Merge -->
    <feBlend mode="multiply" in="bokeh3" in2="bokeh2"/>
    <feBlend mode="lighten" in2="bokeh1"/>

    <feMorphology operator="erode" radius="0" result="bokeh"/>
    <feGaussianBlur stdDeviation="9" in="SourceGraphic"/>
    <feComposite operator="over" in="bokeh"/>
    <feComposite operator="in" in2="SourceGraphic"/>

</filter>
</defs>

<image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#BokehBlur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```



( Исходное изображение [Daderot](#) в Викискладе)

## Теневые фильтры: базовый Dropshadow

```
<svg width="800px" height="600px">
```

```

<defs>
  <filter id="drop-shadow">
    <feGaussianBlur in="SourceAlpha" stdDeviation="4"/>
    <feOffset dx="5" dy="5" result="offsetblur"/>
    <feFlood flood-color="red"/>
    <feComposite in2="offsetblur" operator="in"/>
    <feMerge>
      <feMergeNode/>
      <feMergeNode in="SourceGraphic"/>
    </feMerge>
  </filter>
</defs>

  <text filter="url(#drop-shadow)" x="30" y="100" font-size="80">SVG Filters</text>

</svg>

```

## Теневые фильтры: Внутреннее свечение

```

<svg width="800px" height="600px">
<defs>
  <filter id="inner-glow">
    <feFlood flood-color="red"/>
    <feComposite in2="SourceAlpha" operator="out"/>
    <feGaussianBlur stdDeviation="2" result="blur"/>
    <feComposite operator="atop" in2="SourceGraphic"/>
  </filter>
</defs>

  <text filter="url(#inner-glow)" x="30" y="100" font-size="80" font-family="Sans-Serif" font-weight="bold">SVG Filters</text>

</svg>

```

## Теневые фильтры: сложная дробина (контурная, шумная, формованная)

```

<svg width="800px" height="600px">
<defs>
<filter id="complex-shadow" color-interpolation-filters="sRGB" x="-50%" y="-50%" height="200%" width="200%">

  <!-- Take source alpha, offset it by angle/distance and blur it by size -->
  <feOffset id="offset" in="SourceAlpha" dx="11" dy="6" result="SA-offset"/>
  <feGaussianBlur id="blur" in="SA-offset" stdDeviation="4" result="SA-o-blur"/>

  <!-- Apply a contour by using a color curve transform on the alpha and clipping the result to the input -->

  <feComponentTransfer in="SA-o-blur" result="SA-o-b-contIN">
    <feFuncA id="contour" type="table" tableValues="0 1 .3 .1 0.05 .1 .3 1 "/>
  </feComponentTransfer>

  <feComposite operator="in" in="SA-o-blur" in2="SA-o-b-contIN" result="SA-o-b-cont"/>

  <!-- Adjust the spread by multiplying alpha by a constant factor --> <feComponentTransfer in="SA-o-b-cont" result="SA-o-b-c-sprd">
    <feFuncA id="spread-ctrl" type="linear" slope="2.8"/>
  </feComponentTransfer>

</filter>
</defs>

  <text filter="url(#complex-shadow)" x="30" y="100" font-size="80" font-family="Sans-Serif" font-weight="bold">SVG Filters</text>

</svg>

```

```

</feComponentTransfer>

<!-- Adjust color and opacity by adding fixed offsets and an opacity multiplier -->
<feColorMatrix id="recolor" in="SA-o-b-c-sprd" type="matrix" values="0 0 0 0 0.945 0 0 0 0
0.137 0 0 0 0 0.137 0 0 0 0.49 0" result="SA-o-b-c-s-recolor"/>

<!-- Generate a grainy noise input with baseFrequency between approx .5 to 2.0. And add the
noise with k1 and k2 multipliers that sum to 1 -->
<feTurbulence result="fNoise" type="fractalNoise" numOctaves="6" baseFrequency="1.98"/>
<feColorMatrix in="fNoise" type="matrix" values="1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 7 -3"
result="clipNoise"/>
<feComposite id="noisemix" operator="arithmetic" in="SA-o-b-c-s-recolor" in2="clipNoise"
k1="0.67" k2="0.33" result="SA-o-b-c-s-r-mix"/>

<!-- Merge the shadow with the original -->
<feMerge>
  <feMergeNode in="SA-o-b-c-s-r-mix"/>
  <feMergeNode in="SourceGraphic"/>
</feMerge>
</filter>
</defs>

<text filter="url(#complex-shadow)" x="30" y="100" font-size="80" font-family="Sans-Serif"
font-weight="bold">SVG Filters</text>

</svg>

```

## Фильтры цветного манипулирования: основные оттенки серого

```

<svg width="800px" height="600px">
  <defs>
    <filter id="greyscale">
      <feColorMatrix type="matrix"
        values="0.2126 0.7152 0.0722 0 0
              0.2126 0.7152 0.0722 0 0
              0.2126 0.7152 0.0722 0 0
              0 0 0 1 0"/>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
  <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```





( Исходное изображение Daderot в Викискладе)

## Фильтры цветной манипуляции: оттенки серого (только зеленый канал)

```
<svg width="800px" height="600px">
  <defs>
    <filter id="greyscale">
      <feColorMatrix type="matrix"
        values="0 1 0 0 0
              0 1 0 0 0
              0 1 0 0 0
              0 0 0 1 0"/>
    </filter>
  </defs>

  <image
    xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
    x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#greyscale)"
    xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
    x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```



( Исходное изображение Daderot в Викискладе)

## Фильтры цветного манипулирования: монотонные

```
<svg width="800px" height="600px">
  <defs>
    <filter id="greyscale">
      <feColorMatrix type="matrix"
        values=".2 .2 .2 0 0
              .2 .2 .2 0 0
              .2 .2 .2 0 0
              0 0 0 1 0"/>
    </filter>
  </defs>

  <image
    xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
    x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#greyscale)"
    xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
    x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>
```

```

        .6 .6 .6 0 0
        .2 .2 .2 0 0
        0 0 0 1 0"/>

</filter>
</defs>

<image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#greyscale)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```



( Исходное изображение Daderot в Викискладе)

## Фильтры размытия: размытие фокуса (гауссовское)

```

<svg width="800px" height="600px">
  <defs>
    <filter id="focus-blur" >
      <feDiffuseLighting result = "diffOut" diffuseConstant = "1" lighting-color="white">
        <feSpotLight id="spotlight" x = "500" y = "100" z = "150" pointsAtX = "500" pointsAtY =
"100" pointsAtZ = "0" specularExponent = "12" limitingConeAngle="70"/>
      </feDiffuseLighting>

      <feColorMatrix in="diffOut" result="alphaMap" type="luminanceToAlpha"/>
      <feComponentTransfer in="alphaMap" result="invertlight">
        <feFuncA type="table" tableValues="1 0 0"/>
      </feComponentTransfer>

      <feGaussianBlur in="invertlight" result="featherspot" stdDeviation="5"/>
      <feComposite operator="xor" result="infocus" in2="SourceGraphic" in="featherspot"/>
      <feGaussianBlur in="SourceGraphic" result="outfocus" stdDeviation="2"/>
      <feComposite operator="over" in="infocus" in2="outfocus"/>
      <feComposite operator="in" in2="SourceGraphic"/>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="20px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#focus-blur)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/a/af/Fruit_Stall_in_Barcelona_Market.jpg"
x="340px" y="20px" width="300px" height="200px" preserveAspectRatio="xMinYMin meet"/>
</svg>

```



( Исходное изображение Daderot в Викискладе)

## Фильтры цветного манипулирования: плакат

```
<svg width="800px" height="600px" >
  <defs>
    <filter id="posterize" color-interpolation-filters="sRGB">
      <feComponentTransfer>
        <feFuncR type="discrete" tableValues="0 0.25 0.75 1.0"/>
        <feFuncG type="discrete" tableValues="0 0.25 0.75 1.0"/>
        <feFuncB type="discrete" tableValues="0 0.25 0.75 1.0"/>
      </feComponentTransfer>
    </filter>
  </defs>

  <image
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg" x="20px"
y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet" />
    <image filter="url(#posterize)"
xlink:href="https://upload.wikimedia.org/wikipedia/commons/4/42/Andy_Warhol_1975.jpg"
x="340px" y="20px" width="300px" height="600px" preserveAspectRatio="xMinYMin meet"/>
  </svg>
```

## Фильтры размытия: выделить размытие

Этот фильтр выбирает только области высокой яркости графического изображения источника, размывает содержимое и компонует размытое содержимое поверх оригинала.

```
<svg width="800px" height="600px">
  <defs>
    <filter id="highlightblur" color-interpolation-filters="sRGB">
      <feColorMatrix type="luminanceToAlpha" in="SourceGraphic" result="lumMap"/>
      <feComponentTransfer in="lumMap" result="highlightMask">
        <feFuncA type="discrete" tableValues="0 0 0 0 0 0 0 1"/>
      </feComponentTransfer>
      <feComposite operator="in" in="SourceGraphic" in2="highlightMask"
result="highlights"/>
      <feGaussianBlur in="highlights" stdDeviation="3" result="highBlur"/>
      <feComposite operator="over" in="highBlur" in2="SourceGraphic" result="final"/>
    </filter>
  </defs>

  <image filter="url(#highlightblur)" x="0" y="-40" width="780" height="600"
preserveAspectRatio="true"
```



```
xlink:href="http://i554.photobucket.com/albums/jj424/allbowerpower/Christmas%202009/ChristmasTablesett  
/>  
</svg>
```

Прочитайте фильтры онлайн: <https://riptutorial.com/ru/svg/topic/3262/фильтры>

## глава 23: Цвета

### Examples

Именованные цвета - используйте predefined имена для атрибутов fill и stroke

Список признанных имен ключевых слов цвета можно найти в [Рекомендации W3C для SVG](#)

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="red" stroke="green" />
  <rect x="200" y="200" width="50" height="50" fill="yellow" stroke="blue" />
</svg>
```

### Цвет RGB с использованием шестнадцатеричной нотации

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#ff0000" stroke="#00ff00" />
  <rect x="200" y="200" width="50" height="50" fill="#ffff00" stroke="#00ffff" />
</svg>
```

То же, что и выше, используя [сокращенную шестнадцатеричную форму](#) :

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="#f00" stroke="#0f0" />
  <rect x="200" y="200" width="50" height="50" fill="#ff0" stroke="#0ff" />
</svg>
```

### RGB цвета с функциональной нотацией - целые значения или проценты

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="rgb(255, 0, 0)" stroke="rgb(0, 255, 0)" />
  <rect x="200" y="200" width="50" height="50" fill="rgb(100%, 100%, 0%)" stroke="rgb(0%,
100%, 100%)" />
</svg>
```

в функциональной нотации также поддерживаются значения RGBA.

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <circle r="30" cx="100" cy="100" fill="rgba(255, 0, 0, 0.5)" stroke="rgba(0, 255, 0, 0.5)" />
  <rect x="200" y="200" width="50" height="50" fill="rgba(100%, 100%, 0%, 0.5)"
stroke="rgba(0, 100%, 100%, 0.5)" />
</svg>
```

### Ключевое словоcurrentColor

`currentColor` наиболее полезен для встроенных SVG. С этим вы можете наследовать цвет CSS родителей и использовать его везде, где используются цвета в SVG.

В этом примере первый круг использует цвет текста как цвет заливки, а второй круг использует его как цвет штриха.

```
<html>
  <head>
    div{color:green}
  </head>
  <body>
    <div>
      some Text
      <svg width="2em" height="1em" viewBox="0 0 200 100">
        <circle cx="50" cy="50" r="45" fill="currentColor"/>
        <circle cx="150" cy="50" r="45" fill="none" stroke-width=5
stroke="currentColor"/>
      </svg>
    </div>
  </body>
</html>
```

Прочитайте Цвета онлайн: <https://riptutorial.com/ru/svg/topic/2463/цвета>

---

## глава 24: Элемент SVG

### Examples

#### Viewbox

Атрибут `viewBox` определяет систему координат для элемента `<svg>`. Это позволяет легко изменять размер и относительную пропорцию изображения SVG без необходимости корректировать положение и размеры каждого отдельного рисованного элемента.

```
<!-- stretches a small icon to 60px square -->
<svg viewBox="0 0 16 16" height="60px" width="60px">
  <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904
5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

Этот код выглядит следующим образом:



Без `viewBox` это выглядит так:



#### preserveAspectRatio

`preserveAspectRatio` - это атрибут, который указывает, следует ли масштабировать изображение равномерно. Этот атрибут работает только в том случае, если элемент `<svg>` также имеет `viewBox`.

Значение по умолчанию - `xMidYMid`, которое поддерживает соотношение сторон и центрирует путь внутри контейнера SVG:

```
<!-- when not included `preserveAspectRatio` defaults to `xMidYMid` -->
<svg viewBox="0 0 16 16" height="60" width="120">
  <path d="M16 6.216l-6.095-.02L7.98.38 6.095 6.196 0 6.215h.02l4.912 3.57-1.904
5.834h.02l4.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```



Если для параметра `preserveAspectRatio` установлено значение `none` , значок растягивается в соответствии с полем:

```
<svg viewBox="0 0 16 16" height="60" width="120" preserveAspectRatio="none">
  <path d="M16 6.2161-6.095-.02L7.98.38 6.095 6.196 0 6.215h.0214.912 3.57-1.904
5.834h.0214.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```



Есть много других значений для `preserveAspectRatio` , но эти два являются, безусловно, наиболее распространенными.

## `preserveAspectRatio` - атрибуты соответствия и среза

Атрибут `preserveAspectRatio` имеет необязательный параметр: `meet` | `slice` . Поведение по умолчанию `meet` тянувшийся содержание как в x и y измерения , пока он не заполнит ширину **или** высоту Viewbox. Альтернатива - `slice` сохраняет соотношение сторон содержания , но масштабируется графику , пока она не заполнит **и** ширину и высоту окна видимости (обрезкой содержимое , которое перетекает в Viewbox).

Это пример использования `slice`

```
<svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin slice">
<path d="M16 6.2161-6.095-.02L7.98.38 6.095 6.196 0 6.215h.0214.912 3.57-1.904
5.834h.0214.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
```

который отображается как:



и тот же пример , использующий `meet`

```
<svg viewBox="0 0 16 16" height="60px" width="120px" preserveAspectRatio="xMinYMin meet">
  <path d="M16 6.2161-6.095-.02L7.98.38 6.095 6.196 0 6.215h.0214.912 3.57-1.904
5.834h.0214.972-3.59 4.932 3.59-1.904-5.815L16 6.215" />
</svg>
```

который отображается как:



Прочитайте Элемент SVG онлайн: <https://riptutorial.com/ru/svg/topic/6923/элемент-svg>

## глава 25: Эллипс

### параметры

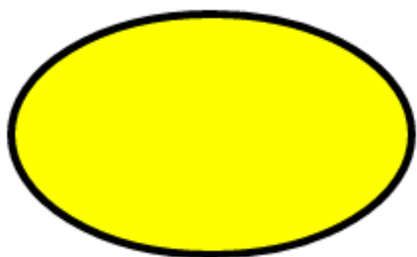
параметр	подробности
cx	Х-координата центра эллипса
cy	Координата Y центра эллипса
rx	Горизонтальный радиус
ry	Вертикальный радиус

### Examples

#### Простой желтый эллипс

```
<svg height="80" width="160">  
  <ellipse cx="80" cy="40" rx="50" ry="30"  
    style="fill:yellow; stroke:black; stroke-width:2" />  
</svg>
```

Вынесено:



Прочитайте Эллипс онлайн: <https://riptutorial.com/ru/svg/topic/3993/эллипс>

## кредиты

S. No	Главы	Contributors
1	Начало работы с SVG	<a href="#">almcd</a> , <a href="#">Community</a> , <a href="#">Robert Longson</a> , <a href="#">Timothy Miller</a> , <a href="#">uruloke</a> , <a href="#">w5m</a> , <a href="#">web-tiki</a>
2	clipPath	<a href="#">Danny_ds</a> , <a href="#">lodz</a>
3	DEFS	<a href="#">Michael Mullany</a>
4	Scripting	<a href="#">Michael Mullany</a> , <a href="#">Phrogz</a>
5	Анимация	<a href="#">Joachim Schirrmacher</a> , <a href="#">Michael Mullany</a>
6	Градиенты	<a href="#">Robert Longson</a>
7	использование	<a href="#">Robert Longson</a> , <a href="#">Timothy Miller</a>
8	Круг	<a href="#">almcd</a> , <a href="#">Kake_Fisk</a> , <a href="#">w5m</a>
9	Линия	<a href="#">Deni Spasovski</a> , <a href="#">Michael Mullany</a> , <a href="#">w5m</a>
10	Ломаная	<a href="#">adius</a> , <a href="#">Michael Mullany</a>
11	маркер	<a href="#">Michael Mullany</a>
12	маскировать	<a href="#">Holger Will</a>
13	переключатель	<a href="#">lodz</a>
14	преобразование	<a href="#">ccprog</a> , <a href="#">Michael Mullany</a> , <a href="#">Stephen Leppik</a>
15	Прямоугольник	<a href="#">almcd</a> , <a href="#">w5m</a>
16	пути	<a href="#">Malcolm McLean</a> , <a href="#">Michael Mullany</a> , <a href="#">w5m</a>
17	Создание шрифтов	<a href="#">Holger Will</a>
18	Текст	<a href="#">Kake_Fisk</a> , <a href="#">Michael Mullany</a>
19	Узоры	<a href="#">Michael Mullany</a> , <a href="#">Robert Longson</a>
20	указатель событий	<a href="#">Holger Will</a>
21	фильтры	<a href="#">Anko</a> , <a href="#">Michael Mullany</a> , <a href="#">RamenChef</a>



22	Цвета	<a href="#">Danny_ds</a> , <a href="#">Holger Will</a> , <a href="#">Joachim Schirmacher</a> , <a href="#">Michael Mullany</a> , <a href="#">w5m</a>
23	Элемент SVG	<a href="#">Michael Mullany</a> , <a href="#">Timothy Miller</a>
24	Эллипс	<a href="#">adius</a>