**Пути (paths)**

* [Назад](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Basic_Shapes)
* [Далее](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Fills_and_Strokes)

Элемент [<path>](https://developer.mozilla.org/ru/docs/Web/SVG/Element/path) ("путь")– наиболее мощный элемент в библиотеке [основных форм](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Basic_Shapes) SVG. С его помощью можно создавать отрезки, кривые, дуги и многое другое.

С помощью элементов [<path>](https://developer.mozilla.org/ru/docs/Web/SVG/Element/path) создают сложные формы, объединяя многочисленные прямые и кривые линии. Сложные формы из одних только прямых линий можно создавать и через элемент [polyline](https://developer.mozilla.org/ru/docs/Web). Хотя результаты работы обоих элементов могут быть похожи, элемент polyline отображает кривые как много маленьких прямых линий, что не очень хорошо масштабируется до больших размеров. Хорошее понимание path важно при рисовании SVG. При создании сложных paths не рекомендуется использовать XML или текстовые редакторы – понимание, как они работают, позволит установить и исправить проблемы с отображением SVG.

Форма элемента path определяется одним атрибутом: [d](https://developer.mozilla.org/ru/docs/Web/SVG/Attribute/d) (смотри подробности в [основные формы](https://developer.mozilla.org/ru/docs/Web)). Атрибут d содержит серию команд и параметров, используемых этими командами. Мы опишем доступные команды и покажем примеры того, что они делают.

Каждая команда обозначается специальной буквой. Например, нам надо переместиться в точку с координатами (10,10). Команда "Передвинуть к" вызывается буквой M. Когда синтаксический анализатор наталкивается на эту команду, он знает, что необходимо переместиться к указанной точке. Итак, для перемещения к точке (10,10) используется команда "M 10 10", и далее синтаксический анализатор переходит к следующей команде.

Все команды существуют в двух вариантах: вызов команды с **заглавной буквы** обозначает абсолютные координаты на странице, а команда со **строчной буквой** -относительные (например, *перемещение от последней точки* *на* *10px вверх и 7px влево*).

Координаты в атрибуте d всегда пишутся без указания единиц измерения и находятся в системе пользовательских координат (обычно это пиксели). Позже мы рассмотрим, как элемент paths может быть трансформирован для других нужд.

[**Команды линии**](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Paths#%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B_%D0%BB%D0%B8%D0%BD%D0%B8%D0%B8)

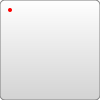
Существуют пять команд линии для узлов <path>. Первая команда - это "Переместиться к", или M, описанная выше. В качестве параметров она принимает координаты точки, к которой перемещается. Если курсор уже был где-либо на странице, между старым и новым местом линия не появится. Команда "Переместиться к" используется в начале элемента <path> для указания точки, откуда начнётся рисование, например:

M x y

или

m dx dy

Следующий пример рисует одну только точку (10,10). Заметьте, однако, что при обычном использовании <path> она вообще не была бы видна.



xmlCopy to Clipboard

<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">

<path d="M10 10"/>

<!-- Точки -->

<circle cx="10" cy="10" r="2" fill="red"/>

</svg>

Существуют три команды, которые рисуют линии. Самая общая - команда "Линия к", вызываемая буквой L. Эта команда принимает два параметра - координаты точки x и y - и рисует линию от текущего положения к этой точке.

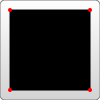
L x y (или l dx dy)

Для рисования горизонтальных и вертикальных линий есть две укороченные формы . H рисует горизонтальную линию, а V рисует вертикальную линию. Обе команды используют только один аргумент, так как они движутся только в одном направлении.

H x (или h dx)

V y (или v dy)

Начнём с рисования простой формы, например, прямоугольника (такого же, какой проще нарисовать с помощью элемента <rect>). Он состоит только из горизонтальных и вертикальных линий:



xmlCopy to Clipboard

<svg width="100" height="100" xmlns="http://www.w3.org/2000/svg">

<path d="M10 10 H 90 V 90 H 10 L 10 10"/>

<!-- Точки -->

<circle cx="10" cy="10" r="2" fill="red"/>

<circle cx="90" cy="90" r="2" fill="red"/>

<circle cx="90" cy="10" r="2" fill="red"/>

<circle cx="10" cy="90" r="2" fill="red"/>

</svg>

Нашу запись в примере выше можно немного сократить , используя команду "Закрыть путь", Z. Эта команда рисует прямую линию от текущего положения обратно к первой точке пути. Она часто встречается в конце узла пути, хотя и не всегда. Для неё регистр буквы не важен.

Z (или z)

Таким образом наш путь из примера можно сократить до:

xmlCopy to Clipboard

<path d="M10 10 H 90 V 90 H 10 Z" fill="transparent" stroke="black"/>

Точно такую же картину можно получить с помощью относительных форм этих команд. Как уже говорилось, относительные команды вызываются использованием букв нижнего регистра и перемещают курсор относительно его последнего положения, а не к точным координатам . В нашем примере, поскольку размеры нашего квадрата - 80 x 80, элемент <path> можно записать так:

xmlCopy to Clipboard

<path d="M10 10 h 80 v 80 h -80 Z" fill="transparent" stroke="black"/>

Путь начнётся от точки (10,10), пойдёт горизонтально на 80 точек вправо, затем 80 точек вниз, затем 80 точек влево, и затем обратно к старту.

Возможно, в этих примерах было бы проще использовать элементы <polygon> или <polyline>. Однако, пути используются при рисовании SVG настолько часто, что многим разработчикам может быть удобнее использовать их вместо других элементов. Нет никакой разницы в производительности при использовании того или другого.

[**Команды кривых линий**](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Paths#%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B_%D0%BA%D1%80%D0%B8%D0%B2%D1%8B%D1%85_%D0%BB%D0%B8%D0%BD%D0%B8%D0%B9)

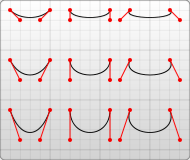
Существует три различных команды, которые вы можете использовать для создания плавных кривых линий. Две из этих кривых - кривые Безье, а третья - "дуга", или часть окружности. Вы, возможно, уже имели практический опыт с кривыми Безье, если работали с путями (paths) в программах lnkscape, Illustrator или Photoshop. Для полного описания математических понятий о кривых Безье, попробуйте пройти по ссылке на [Wikipedia](http://en.wikipedia.org/wiki/B%C3%A9zier_curve" \t "_blank). Информации о кривых Безье слишком много, чтобы попытаться охватить её здесь. Существует бесчисленное множество различных типов кривых Безье, но только две простые доступны для элементов путей: кубическая, C, и квадратная, Q.

[**Кривые Безье**](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Paths#%D0%BA%D1%80%D0%B8%D0%B2%D1%8B%D0%B5_%D0%B1%D0%B5%D0%B7%D1%8C%D0%B5)

Кубическая кривая, C, представляет собой немного более сложную кривую. Кубическая Безье принимает две контрольные точки для каждой точки. Таким образом, чтобы создать кубическую кривую Безье, вам необходимо указать три набора координат.

C x1 y1, x2 y2, x y (or c dx1 dy1, dx2 dy2, dx dy)

Последний набор координат (x,y) это точка, в которой заканчивается линия. Две другие - контрольные точки. (x1,y1) контрольная точка для начала вашей кривой, а (x2,y2) для конца вашей кривой. Если вы знакомы с вычислительной алгеброй, контрольные точки в описывают наклон вашей линии в каждой точке. Функция Безье создаёт плавную кривую, которая ведёт от наклона, который вы установили в начале вашей линии к наклону на другом конце.



xmlCopy to Clipboard

<svg width="190" height="160" xmlns="http://www.w3.org/2000/svg">

<path d="M10 10 C 20 20, 40 20, 50 10" stroke="black" fill="transparent"/>

<path d="M70 10 C 70 20, 120 20, 120 10" stroke="black" fill="transparent"/>

<path d="M130 10 C 120 20, 180 20, 170 10" stroke="black" fill="transparent"/>

<path d="M10 60 C 20 80, 40 80, 50 60" stroke="black" fill="transparent"/>

<path d="M70 60 C 70 80, 110 80, 110 60" stroke="black" fill="transparent"/>

<path d="M130 60 C 120 80, 180 80, 170 60" stroke="black" fill="transparent"/>

<path d="M10 110 C 20 140, 40 140, 50 110" stroke="black" fill="transparent"/>

<path d="M70 110 C 70 140, 110 140, 110 110" stroke="black" fill="transparent"/>

<path d="M130 110 C 120 140, 180 140, 170 110" stroke="black" fill="transparent"/>

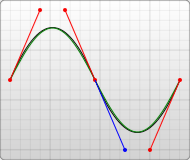
</svg>

Пример сверху показывает девять кубических кривых Безье. Чем ближе к левому краю изображения, тем более горизонтально разделёнными становятся контрольные точки. Ближе к правому, становятся более отделёнными от конечных точек. Стоит отметить, что кривая начинается в направлении первой контрольной точки, а затем изгибается, переходя в направление второй контрольной точки.

Вы можете связать вместе несколько кривых Безье, чтобы создавать более длинные плавные формы. В таком случае часто контрольные точки на разных сторонах будут отображением друг друга (чтобы поддерживать постоянный наклон). В этом случае можно использовать сокращённую запись для кривой Безье, используя команду S (или s).

S x2 y2, x y (or s dx2 dy2, dx dy)

Команда S задаёт тот же тип кривой, что и был, но если он следует за другой S или C командой, подразумевается, что первая контрольная точка - отражение той, что использовалась перед этим. Если команда S не следует за другой командой S или C, то подразумевается, что текущая позиция курсора используется как первая контрольная точка. Пример синтаксиса показан ниже. В фигуре слева контрольные точки показаны красным, а подразумеваемая контрольная точка - синим.



xmlCopy to Clipboard

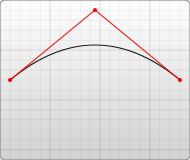
<svg width="190" height="160" xmlns="http://www.w3.org/2000/svg">

<path d="M10 80 C 40 10, 65 10, 95 80 S 150 150, 180 80" stroke="black" fill="transparent"/>

</svg>

Другой тип кривой Безье - квадратичная кривая, задаётся командой Q. Квадратичная кривая проще, чем кубическая: для неё требуется только одна контрольная точка, которая определяет наклон кривой как в начальной, так и в конечной точке. Она принимает два аргумента: контрольную точку и конец кривой.

Q x1 y1, x y (or q dx1 dy1, dx dy)



xmlCopy to Clipboard

<svg width="190" height="160" xmlns="http://www.w3.org/2000/svg">

<path d="M10 80 Q 95 10 180 80" stroke="black" fill="transparent"/>

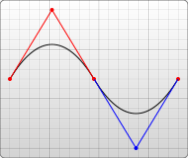
</svg>

Как и в случае кубической кривой Безье, существует сокращение для соединения нескольких квадратичных кривых Безье -T.

T x y (or t dx dy)

Как и ранее, сокращение смотрит на предыдущую контрольную точку, которую вы использовали, и выводит из неё новую. Это означает, что после первой контрольной точки вы можете делать довольно сложные фигуры, указав только конечные точки.

**Примечание:** Обратите внимание, что это работает только в том случае, если предыдущей командой была команда Q или T. Если это не так, то контрольная точка считается той же, что и предыдущая, и вы нарисуете только линии.



xmlCopy to Clipboard

<svg width="190" height="160" xmlns="http://www.w3.org/2000/svg">

<path d="M10 80 Q 52.5 10, 95 80 T 180 80" stroke="black" fill="transparent"/>

</svg>

Обе кривые дают похожие результаты, но кубическая кривая позволяет больше свободы в том, как должна выглядеть кривая. Решение, какую кривую использовать - ситуационное и зависит от количества симметрии в линиях.

[**Дуги**](https://developer.mozilla.org/ru/docs/Web/SVG/Tutorial/Paths#%D0%B4%D1%83%D0%B3%D0%B8)

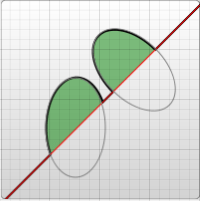
Другой тип кривых линий, которые можно создать с помощью SVG - дуга (команда A). Дуги - секции кругов или эллипсов. При заданных x- и y-радиусах есть два эллипса, которые могут соединяться любыми двумя точками (пока они находятся внутри радиуса круга). Вдоль любого из этих кругов есть два пути, которые могут использовать для соединения точек, так что в любой ситуации возможно 4 дуги.

A rx ry x-axis-rotation large-arc-flag sweep-flag x y

a rx ry x-axis-rotation large-arc-flag sweep-flag dx dy

В начале элемент дуги принимает два аргумента радиусов. Если нужно, обратитесь к разделу [Эллипсы](https://developer.mozilla.org/ru/docs/Web/SVG/Element/ellipse), чтобы увидеть, как они работают. Последние два аргумента предназначены для обозначения координат окончания дуги. Вместе эти четыре значения определяют основную структуры дуги.

Третий параметр описывает поворот дуги. См. пример ниже



xmlCopy to Clipboard

<svg width="320" height="320" xmlns="http://www.w3.org/2000/svg">

<path d="M10 315

L 110 215

A 30 50 0 0 1 162.55 162.45

L 172.55 152.45

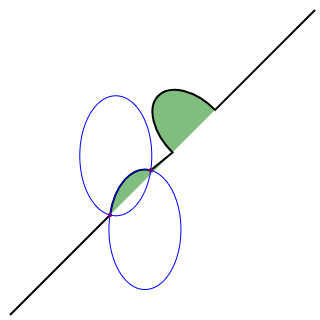
A 30 50 -45 0 1 215.1 109.9

L 315 10" stroke="black" fill="green" stroke-width="2" fill-opacity="0.5"/>

</svg>

Пример показывает элемент *path*, который проходит по странице диагонально. В центре этого элемента вырезаны две эллиптические дуги (радиус x = 30, радиус y = 50). В первой дуге параметр *x-asix-rotation = 0*, а это означает, что эллипс, по которому проходит дуга (показан серым) расположен вертикально. Во второй дуге параметр *x-asix-rotation = -45.* Это поворачивает эллипс так, что направление его малой оси совпадает с направлением пути, как это видно на рисунке выше.

Четыре разных пути, упомянутых выше определяются с помощью двух аргументов-флагов. Как упоминалось ранее, есть ещё два возможных эллипса для обхода пути и два разных возможных пути на обоих эллипсах, что даёт четыре возможных пути. Первый аргумент - *large-arc-flag.* Он определяет, должна ли дуга быть больше или меньше 180 градусов. В конечном счёте этот флаг определяет, в каком направлении дуга будет обходить данный круг. Второй аргумент - *sweep-flag.* Он определяет, должна дуга двигаться по отрицательным углам или по положительным, т.е. по сути определяет по какому из двух кругов она будет идти. Пример ниже показывает все четыре возможные комбинации.



xmlCopy to Clipboard

<svg xmlns="http://www.w3.org/2000/svg" width="320" height="320">

<path d="M 10 315

L 110 215

A 36 60 0 0 1 150.71 170.29

L 172.55 152.45

A 30 50 -45 0 1 215.1 109.9

L 315 10" stroke="black" fill="green" stroke-width="2" fill-opacity="0.5"/>

<circle cx="150.71" cy="170.29" r="2" fill="red"/>

<circle cx="110" cy="215" r="2" fill="red"/>

<ellipse cx="144.931" cy="229.512" rx="36" ry="60" fill="transparent" stroke="blue"/>

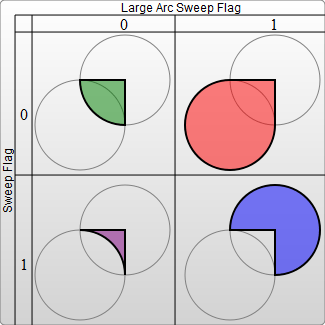
<ellipse cx="115.779" cy="155.778" rx="36" ry="60" fill="transparent" stroke="blue"/>

</svg>

Заметьте, что каждый голубой эллипс сформирован двумя дугами, в зависимости от того движетесь ли вы по часовой или против часовой стрелке. Каждый эллипс имеет короткую и длинную дуги. Оба эллипса просто зеркальные отражения друг друга. Они отражены вдоль линии, сформированной start->end точками.

Если start->end точки расположены далеко и не попадают в пределы градусов эллипсов по x и y, то в этом случае радиусы эллипсов будут увеличены до величины, нужной чтобы достичь точек start->end. Интерактивный codepen внизу этой страницы наглядно это демонстрирует. Для определения достаточны ли велики радиусы ваших эллипсов чтобы требовать увеличения, вам нужно решить систему уравнений подобную [этой на wolfram alpha](https://www.wolframalpha.com/input/?i=solve+((110+-+x)%5E2%2F36%5E2)+%2B+((215+-+y)%5E2%2F60%5E2)+%3D+1,+((150.71+-+x)%5E2%2F36%5E2)+%2B+((170.29+-+y)%5E2%2F60%5E2)+%3D+1" \t "_blank). Это вычисление для non-rotated эллипса с start->end (110, 215)->(150.71, 170.29). Решением, (x, y), является центр эллипса(ов). Следующее вычисление для non-rotated эллипса с start->end (110, 215)->(162.55, 162.45). Решение будет [мнимым](https://www.wolframalpha.com/input/?i=solve+((110+-+x)%5E2%2F30%5E2)+%2B+((215+-+y)%5E2%2F50%5E2)+%3D+1,+((162.55+-+x)%5E2%2F30%5E2)+%2B+((162.45+-+y)%5E2%2F50%5E2)+%3D+1" \t "_blank) если радиусы ваших эллипсов слишком малы. Решение содержит небольшой мнимый компонент потому, что эллипсы были лишь слегка расширены.

Четыре разных пути, упомянутых выше определяются с помощью двух аргументов-флагов. Как упоминалось ранее, есть ещё два возможных эллипса для обхода пути и два разных возможных пути на обоих эллипсах, что даёт четыре возможных пути. Первый аргумент - *large-arc-flag.* Он определяет, должна ли дуга быть больше или меньше 180 градусов. В конечном счёте этот флаг определяет, в каком направлении дуга будет обходить данный круг. Второй аргумент - *sweep-flag.* Он определяет, должна дуга двигаться по отрицательным углам или по положительным, т.е. по сути определяет по какому из двух кругов она будет идти. Пример ниже показывает все четыре возможные комбинации.



xmlCopy to Clipboard

<svg width="325" height="325" xmlns="http://www.w3.org/2000/svg">

<path d="M80 80

A 45 45, 0, 0, 0, 125 125

L 125 80 Z" fill="green"/>

<path d="M230 80

A 45 45, 0, 1, 0, 275 125

L 275 80 Z" fill="red"/>

<path d="M80 230

A 45 45, 0, 0, 1, 125 275

L 125 230 Z" fill="purple"/>

<path d="M230 230

A 45 45, 0, 1, 1, 275 275

L 275 230 Z" fill="blue"/>

</svg>

Последние два аргумента, если вы ещё не догадались, обозначают координаты x и y, где заканчивается дуга. Дуги - лёгкий способ создавать части кругов или эллипсов в ваших рисунках. Например, круговая диаграмма требует отдельную дугу для каждого куска диаграммы.

Если вы переходите в SVG из Canvas а, дуги могут быть самой трудной вещью для изучения, но они также очень мощные. Т.к. начальная и конечные точки для любого пути, обходящего круг, одно и то же место, существует бесконечное количество кругов, которые могут быть выбраны и действительный путь не определён. Возможно приблизить их, сделав начальную и конечную точку пути слегка разными и соединив их с другими сегментами пути. В этой точке, часто проще использовать настоящий круг или эллипс. Это интерактивное демо может помочь понять основные принципы SVG-дуг: <https://codepen.io/lingtalfi/pen/yaLWJG> (протестировано только в Chrome и Firefox, может не работать в вашем браузере)