**Привет, SVG!**

SVG — это формат векторной графики. В отличие от растровой графики — PNG, GIF, JPEG — SVG может растягиваться и сжиматься без потери качества, то есть такие картинки будут одинаково чёткими и на обычных экранах, и на ретине.

Ещё одно из достоинств SVG — человекопонятный код: его можно не только прочитать, но и написать руками. Можно открыть файл и отредактировать его без использования графического редактора, можно самому написать простую картинку.

Также SVG-элементы можно оформить с помощью CSS и добавить им интерактивности с помощью JavaScript, а кроме того, SVG [достаточно хорошо поддерживается](http://caniuse.com/#search=svg) всеми современными браузерами, и его уже можно активно использовать.

Давайте познакомимся с ним поближе. Вот простой пример кода:

<svg>

<circle r="50" cx="50%" cy="50%" fill="yellowgreen"/>

</svg>

SVG-элемент вставляется с помощью тега svg, внутри которого уже находится остальное содержимое: фигуры, картинки или текст.

Содержимое в этом примере — это кружок (circle) зелёного цвета (fill="yellowgreen"). Вот так будет выглядеть действие этого кода в браузере:

SVG можно встраивать несколькими разными способами, мы рассмотрим их позже, а сейчас будем вставлять его непосредственно в код страницы.

# Рисуем прямоугольник

В SVG есть несколько способов нарисовать фигуру, для простых фигур есть свои теги.

Например, прямоугольник рисуется с помощью тега rect. Код простого прямоугольника выглядит так:

<rect width="150" height="100"/>

​Обратите внимание: все теги в SVG должны быть закрыты, то есть должно быть так: <rect .../> или так: <rect ...></rect>. Мы будем использовать первый способ.

Результат:

Атрибуты width и height управляют, соответственно, шириной и высотой фигуры. Значения можно задавать и в пикселях, и в процентах.

Для значений в пикселях после значения не нужно писать px, потому что пиксели — единица измерения, используемая в SVG по умолчанию. Проценты рассчитываются относительно размеров всего SVG-изображения: горизонтальные значения относительно ширины, вертикальные — относительно высоты.

В современных браузерах размерами и положением фигур нельзя управлять через CSS, но эта возможность появится в будущем.

# Координаты прямоугольника

Чтобы задать координаты прямоугольника, используются атрибуты x и y:

<rect width="50%" height="100" x="20" y="50"/>

Координаты определяют положение верхнего левого угла фигуры.

# Скругление углов

Скруглением углов прямоугольника управляют параметры rx и ry. Атрибут rx задаёт скругление по горизонтали, а ry — по вертикали. Если атрибут ry не задан, он будет равен rx.

Пример кода:

<rect width="50%" height="100" rx="50" ry="20"/>

В [задании части «Рамки и фоны. Погружение»](https://htmlacademy.ru/courses/88/run/12) описан похожий эффект скругления углов, но с помощью CSS-свойства border-radius.

# Многоугольники

В SVG можно рисовать не только четырёхугольники, но и многоугольники, это делается с помощью тега polygon. Пример кода:

<polygon points="70,5 90,41 136,48 103,80 111,126 70,105 29,126 36,80 5,48 48,41"/>

И результат:

В атрибуте points задаются координаты вершин фигуры. Каждая координата задаётся по x и y. Координаты в points нельзя задавать в процентах.

# Рисуем окружность

Окружность рисуется с помощью тега circle. Пример кода:

<circle r="50"/>

Атрибут r — радиус окружности.

В отличие от предыдущих фигур, положение окружности в пространстве определяется координатами центра фигуры: атрибут cx задаёт положение по горизонтальной оси, cy — по вертикальной.

По умолчанию координаты центра окружности равны 0,0, поэтому она находится в верхнем левом углу. Подвинем фигуру:

<circle r="50" cx="100" cy="50%"/>

Значения можно задавать как в пикселях, так и в процентах. Процентные значения рассчитываются относительно размеров SVG-элемента.

Радиус и координаты можно задавать только атрибутами, с помощью CSS это сделать нельзя.

# Рисуем эллипс

Эллипс рисуется почти так же, как круг, но у него два радиуса: по горизонтальной оси — rx, и по вертикальной — ry.

<ellipse rx="30" ry="40%"/>

Расположение эллипса, так же как и для circle, задаётся с помощью cx и cy.

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

# Рисуем линии

Линии рисуются с помощью тега line. Координаты начала линии задаются атрибутами x1 и y1, координаты конца — атрибутами x2 и y2. Координаты можно задавать в процентах.

Пример кода:

<line x1="220" y1="10" x2="20" y2="130"/>

Так как линия не образует фигуру с внутренним контуром, для отображения ей нужно задать не заливку, а обводку. Обводкой управляют два атрибута: stroke и stroke-width. Атрибут stroke задаёт цвет обводки, stroke-width — толщину линии.

<line x1="220" y1="20" x2="20" y2="90" stroke="violet" stroke-width="5" />

Результат:

Можно задать только цвет линии, тогда толщина обводки по умолчанию будет равна одному пикселю.

# Рисуем ломаные линии

Ломаные линии рисуются с помощью тега polyline. Координаты точек на линии задаются в атрибуте points, как для polygon.

Пример кода:

<polyline points="10,135 100,10 55,135 10,10 105,135"/>

Результат:

Разница между polygon и polyline заключается в поведении обводки: у многоугольника обводка замыкается сама по себе (левая фигура), а у ломаной линии — остаётся незамкнутой (фигура справа):

# Заливки

В [части про фигуры](https://htmlacademy.ru/courses/130) вы уже немного познакомились с возможностями оформления векторных фигур, теперь мы рассмотрим этот вопрос подробнее.

SVG-фигуры имеют богатые возможности оформления: им, как и HTML-элементам, можно задавать заливку цветом, градиентом или картинкой, но помимо этого также можно управлять отдельно прозрачностью заливки и обводки, а в качестве заливки можно использовать, например, текст.

Также интересные возможности имеет обводка. Например, можно управлять видом пунктирной обводки и сделать обводку точками, пунктиром или морзянкой, а ещё обводку можно сделать не только цветом, но также градиентом или картинкой.

Начнём с заливки. Если она не задана, по умолчанию фигура заполняется чёрным цветом:

<circle r="60" cx="150" cy="50%"></circle>

Цвет заливки задаётся атрибутом fill:

<circle r="60" cx="150" cy="50%" fill="gold"></circle>

либо аналогичным свойством в CSS:

circle {

fill: gold;

}

Результат будет одинаковым:

# Прозрачность заливки

Управлять прозрачностью заливки можно с помощью свойства fill-opacity. Прозрачность также можно задавать как атрибутом, так и через CSS.

Значение задаётся числом от 0 до 1, например:

<rect width="150" height="100" fill="gold" fill-opacity="0.5"></rect>

rect {

fill: gold;

fill-opacity: 0.5;

}

Прозрачность работает для всех видов заливок, в том числе для градиентов и паттернов:

# Отсутствие заливки

Иногда бывает нужно полностью убрать заливку, например, если вам нужен только контур фигуры. Это можно сделать ключевым словом none, результатом будет полная прозрачность фигуры.

# Обводки

Обводки задаются с помощью нескольких атрибутов, причём цвет и толщина обводки задаются отдельно. Цвет задаётся атрибутом stroke:

<circle r="60" cx="150" cy="50%" fill="none" stroke="orange"></circle>

либо через CSS:

circle {

stroke: orange;

}

Результат будет одинаковым, у фигуры появится обводка толщиной один пиксель:

# Толщина обводки

Для однопиксельной обводки достаточно задать только цвет в stroke. Если же нужно управлять толщиной обводки, это делается с помощью свойства stroke-width, также атрибутом или через CSS:

<circle r="60" cx="150" cy="50%" fill="none" stroke="orange"

stroke-width="5"></circle>

или:

circle {

stroke: orange;

stroke-width: 5;

}

# Прозрачность обводки

Прозрачность задаётся свойством stroke-opacity со значениями от 0 до 1, например: stroke-opacity="0.5".

Либо через CSS:

rect {

stroke-opacity: 0.5;

}

# Концы линий

В SVG можно указывать, как ведёт себя обводка на концах линий, это делается с помощью атрибута stroke-linecap. В примерах ниже мы добавили вспомогательные светлые линии с обводкой и показали возможные значения stroke-linecap:

1. butt — значение по умолчанию. С этим значением обводка просто заканчивается на концах линии:

2. round, с этим значением обводка равномерно закругляется вокруг концов линии:

3. square, с этим значением вокруг концов линии добавляется дополнительная обводка с прямоугольными краями:

В CSS внешним видом обводки на концах линий можно управлять с помощью аналогичного свойства stroke-linecap.

# Вид сгибов

Атрибут stroke-linejoin позволяет управлять видом обводки на сгибах линий.

Аналогично [прошлому заданию](https://htmlacademy.ru/courses/187/run/9) мы добавили «воображаемую» светлую линию в примеры. Итак, возможные значения stroke-linejoin:

1. miter — значение по умолчанию. Обводка в месте сгиба линии никак не видоизменяется.

2. round, обводка в месте сгиба линии равномерно закругляется.

3. bevel, обводка в месте сгиба линии складывается как лента.

В CSS внешним видом обводки на сгибах линий можно управлять с помощью аналогичного свойства stroke-linejoin.

# Пунктирные линии, часть 1

В SVG можно управлять видом пунктирных линий, это делается с помощью атрибута stroke-dasharray. В качестве значения задаётся длина отрезков и пробелов между ними. Можно задать одно число, в этом случае получится пунктирная линия, состоящая из отрезков и пробелов одинаковой длины:

1. stroke-dasharray="15":

2. Если задать два числа stroke-dasharray="50 10", первое будет управлять длиной отрезков, второе — длиной пробелов:

3. Последовательность можно продолжить stroke-dasharray="1 2 3 5 8 13 21", в этом случае получится пунктирная линия со сложным ритмом:

Аналогично работает CSS-свойство stroke-dasharray.

# Пунктирные линии, часть 2

Видом обводки круглой фигуры тоже можно управлять.

В отличие от прямоугольных форм сегменты пунктирной обводки круглой формы могут иметь неравномерную форму в зависимости от толщины обводки.

Давайте попробуем создать пунктирную обводку окружности и будем менять толщину, чтобы проследить, как меняется форма сегментов обводки.

# Сдвиг обводки

Пунктирной обводке с помощью атрибута stroke-dashoffset можно задать сдвиг. Возьмём такой пример:

stroke-dasharray="50"

и добавим stroke-dashoffset: stroke-dasharray="50" stroke-dashoffset="25"

Пунктиры обводки сдвинулись на заданный отрезок по направлению против часовой стрелки. Значение stroke-dashoffset может быть отрицательным, и тогда обводка будет смещаться по часовой стрелке.

Аналогично работает CSS-свойство stroke-dashoffset.

# Ширина и высота SVG

SVG ведёт себя иначе, чем привычные HTML-элементы: его содержимое отрисовывается на бесконечном холсте, и его размеры не зависят от содержимого. Видимая часть холста соответствует размерам SVG-элемента, эта область отрисовки называется **вьюпорт**.

При этом можно управлять как размерами SVG-элемента, так и поведением его содержимого: оно может отображаться целиком, обрезаться или сжиматься не сохраняя пропорции.

Если SVG просто вставить на страницу не указывая размеры, он отобразится размером 300×150 пикселей:

<svg>

…

</svg>

Поменять ширину и высоту можно с помощью width и height:

<svg width="350" height="200">

…

</svg>

Задавать размеры можно как атрибутами, так и в CSS:

svg {

width: 350px;

height: 200px;

}

Для размеров в CSS обязательно указывать единицы измерения. Для размеров в атрибутах, задаваемых в пикселях, единицы измерения не нужны.

# Атрибут viewBox

Вы наверняка заметили, что изменение размеров SVG-элемента не влияет на его содержимое — потому что содержимое отрисовывается на бесконечном холсте, и непонятно какого размера область нужно растягивать или сжимать.

Это поведение можно изменить, задав размер области, которая будет тянуться, с помощью свойства viewBox (его можно задать только атрибутом):

<svg viewBox="0 0 237 300" width="350" height="200">

…

</svg>

Первые два числа — координаты X и Y верхнего левого угла масштабируемой области, два других — её ширина и высота. Значения задаются в пикселях, единицы измерения указывать не нужно.

С вьюбоксом содержимое масштабируется, чтобы поместиться целиком в контейнер, и выравнивается по центру.

# viewBox и размеры

Как мы увидели в прошлом задании, SVG без размеров, но с viewBox, пытается занять всё доступное пространство. Это означает, что если на странице есть инлайновые иконки, размеры которым задаются в CSS, без CSS могут растянуться на весь экран.

Чтобы этого избежать, достаточно всем инлайновым иконкам в атрибутах явно задавать размеры по умолчанию, они потом легко переопределяются в CSS.

# Атрибут preserveAspectRatio

По умолчанию содержимое SVG с viewBox масштабируется сохраняя пропорции, и если соотношения сторон вьюпорта и вьюбокса не совпадают, вокруг содержимого появляются поля:

С помощью свойства preserveAspectRatio это поведение можно изменять: например, значение none указывает, что сохранять пропорции не нужно:

<svg viewBox="0 0 237 300" preserveAspectRatio="none">

…

</svg>

В этом случае область, размеры которой заданы вьюбоксом, растягивается на всё доступное пространство вьюпорта:

preserveAspectRatio задаётся только атрибутом.

# Резиновый фон с preserveAspectRatio

SVG, заданный в качестве фона, ведёт себя так же, как инлайновый SVG, поэтому, чтобы получить резиновый фон, используйте SVG с viewBox, но без размеров: в этом случае изображение подгонится под размер элемента, которому задан фон, и будет тянуться вместе с ним, сохраняя пропорции:

Это очень удобно для иконок: задайте размеры родительскому элементу, и иконка, заданная фоном, сама под него растянется.

Если же нужно, чтобы пропорции не сохранялись, добавьте preserveAspectRatio="none". Это пригодится для резиновых фонов:

**Выравнивание в preserveAspectRatio**

Содержимое SVG можно не только растягивать, но и сдвигать вправо-влево или вверх-вниз. Для этого нужно указать положение содержимого относительно осей X и Y, например xMinYMid:

<svg viewBox="0 0 237 300" preserveAspectRatio="xMinYMid">

…

</svg>

Возможные значения для каждой оси:

xMin

xMid

xMax

YMin

YMid

YMax

Положение задаётся двумя параметрами: первым всегда указывается положение по X, вторым по Y. Положение по оси Y всегда пишется с большой буквы. Оба параметра обязательны.

Значение по умолчанию — xMidYMid (содержимое выравнивается по середине большей стороны).

# preserveAspectRatio и viewBox

Нужно помнить, что preserveAspectRatio не работает без viewBox. viewBox определяет масштабируемую область, preserveAspectRatio — как эта область выравнивается и как заполняет собой вьюпорт.

Также preserveAspectRatio не работает, если содержимое отрисовывается без полей (то есть соотношения сторон вьюпорта и вьюбокса совпадают), тогда в нём просто нет необходимости.

**Заполнение пространства**

Второй параметр в свойстве preserveAspectRatio задаёт поведение содержимого относительно вьюпорта, определяет как именно содержимое заполняет пространство:

<svg viewBox="0 0 237 300" preserveAspectRatio="xMinYMin meet">

…

</svg>

Возможные значения:

meet

slice

meet — содержимое умещается целиком, оставляя пустые поля (как при background-size: contain). Значение по умолчанию.

slice — содержимое заполняет собой всё пространство, при этом часть содержимого может быть обрезана (похоже на background-size: cover). Пропорции сохраняются в обоих случаях.

Заполнение — необязательный параметр, его можно не задавать.

# Единицы измерения

Для базового использования SVG достаточно представлять как работают внешние размеры, но для создания более сложных конструкций нужно понимать как работают внутренние.

В SVG можно использовать разные единицы измерения, например: px, em, ex, pt, pc, cm, mm, in и проценты.

Также есть единицы системы координат — *user space units*, которые по умолчанию соответствуют пикселям, поэтому для размеров и координат в пикселях единицы измерения можно не указывать.

**Системы координат**

В SVG существует две системы координат:

1. Система координат вьюпорта — **viewport space**
2. Система координат содержимого — **user space**

Изначально системы и их единицы измерения соответствуют друг другу:

<svg width="350" height="200">

…

</svg>

Сейчас видно только систему координат содержимого (она показана красным), потому что системы совпадают и одна скрыта под другой.

Если добавить вьюбокс или трансформацию, содержимое и его система координат начинают смещаться и масштабироваться:

<svg width="350" height="200" viewBox="0 0 237 300">

…

</svg>

Отсчёт координат содержимого начинается из левого верхнего угла (в точке 0,0). Без вьюбокса это левый верхний угол вьюпорта (бирюзовая точка), с вьюбоксом — левый верхний край вьюбокса (красная точка).

То есть теперь расположение содержимого будет отсчитываться относительно новой системы координат, а не от вьюпорта, из-за чего фигура оказывается не слева, а ближе к центру, а системы координат больше не совпадают.

# Системы координат и трансформации

Трансформации тоже создают свою систему координат. Чтобы применить трансформацию ко всему содержимому, обернём его в группу (элемент <g>):

<svg width="350" height="200">

<g>

…

</g>

</svg>

И добавим трансформацию:

<svg width="350" height="200">

<g transform="translate(50, 50)">

…

</g>

</svg>

Всё содержимое сместилось на 50 пикселей по вертикали и по горизонтали вместе с системой координат, и если теперь добавить ещё одну трансформацию, она уже рассчитывается от новой системы координат:

<svg width="350" height="200">

<g transform="translate(50, 50) rotate(15)">

…

</g>

</svg>

В SVG центр вращения по умолчанию находится в точке 0,0. До первой трансформации это был левый верхний угол вьюпорта, после трансформации — левый верхний угол трансформируемого содержимого. Вторая трансформация снова изменит систему координат группы.

Вставим иконку инлайново.

Не забываем про атрибут aria-hidden="true" и focusable="false", чтобы скрыть иконку от скринридеров.

 <svg aria-hidden="true" focusable="false" width="16" height="16" xmlns="http://www.w3.org/2000/svg">

                <path fill-rule="evenodd" clip-rule="evenodd" d="M11 6.5a4.5 4.5 0 11-9 0 4.5 4.5 0 019 0zm-.665 5.249a6.5 6.5 0 111.414-1.414l3.958 3.958-1.414 1.414-3.958-3.958z" />