Лабораторная работа №6. Знакомство с SVG. Основные паттерны SVG-разметки

Конспект

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 всеми современными браузерами, и его уже можно активно использовать.

Необходимые инструменты

Для выполнения работы вам необходимо использовать:

1) любой текстовый редактор, сохраняя код разметки в файле с расширением FIO_4.html. При этом минимальная структура тегов в файле должна быть следующей:

```
<html>
<head>
<!--  если необходимо, то прикрепляем файл со стилями -->
<link rel="stylesheet" href="style.css">
</head>
<body>

<svg>

<!-- Добавьте сюда содержание svg-объекта-->

</svg>
</body>
</html>
```

2) любую облачную среду разработки, например

   2.1) codepen.io

   2.2) тренажер w3schools.com

Работа с svg-объектами

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

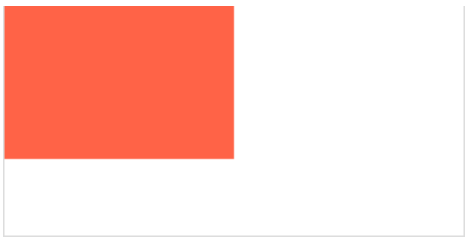
В SVG есть несколько способов нарисовать фигуру, для простых фигур есть свои теги.

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, lолжно 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"/>
```

Задание 1.

1. Нарисуйте круг красного цвета и прямоугольник шириной **50%** и высотой **100** оранжевого цвета (**fill="orange"**).
2. Задайте прямоугольнику атрибут **x** равным 25%, а атрибут **y**— 25 пикселям.
3. Задайте значение **rx** равным **20**. Задайте значение **ry** равным **50**, обратите внимание как изменилась форма фигуры.

Задание 2. «Починка телевизора».

Мама заказала телевизор, но не знала, что доставкой займётся почта РФ. И телевизор пришёл немного «помятый». Почините картинку, подобрав размеры, координаты и радиусы скругления фигур. Все значения кратны пяти.

Исходный код

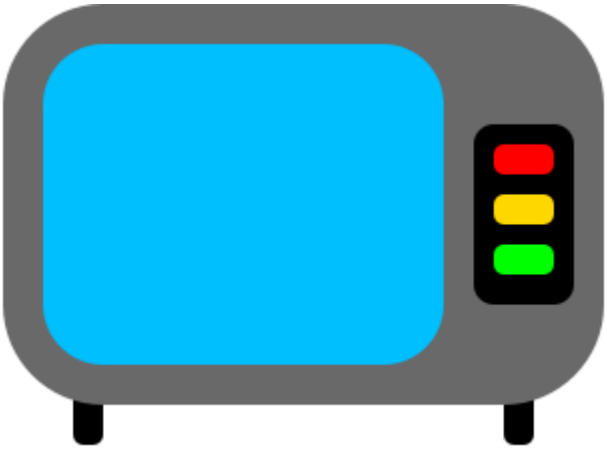
CSS

```
svg {
    border: 1px solid #dddddd;
}
```

HTML

```
<svg height="250" width="330">
  <rect class="leg" width="15" height="30" x="180" y="210" rx="2" fill="black"/>
  <rect class="leg" width="15" height="30" x="5" y="190" rx="10" fill="black"/>
  <rect class="case" width="300" height="200" x="20" y="25" rx="30" fill="dimgray"/>
  <rect class="screen" width="200" height="160" x="30" rx="10" fill="deepskyblue"/>
  <rect class="panel" width="80" height="110" x="200" y="50" fill="black"/>
   <rect class="button" width="20" height="10" x="245" y="70" rx="2" fill="red"/>
   <rect class="button" width="30" height="30" x="265" y="95" rx="4" fill="gold"/>
   <rect class="button" width="40" height="20" x="225" y="120" rx="7" fill="lime"/>
</svg>
```

Результат исправления должен быть таким



Многоугольники

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

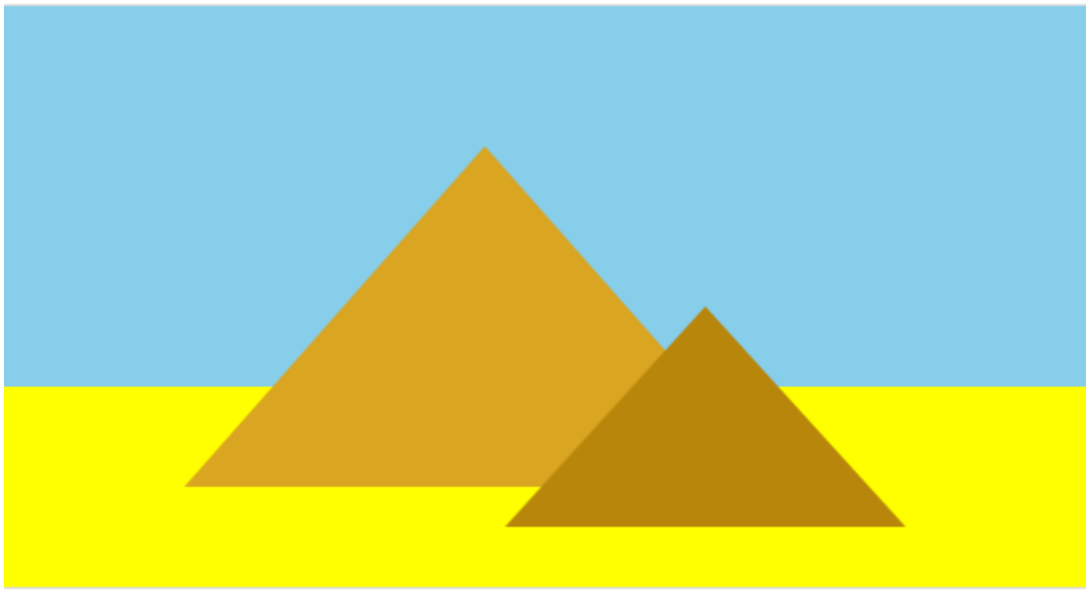
```
<polygon points="70,5 90,41 136,48 103,80 111,126 70,105 29,126 36,80 5,48 48,41"/>
```

И результат:

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

Задание 3. Пирамиды

1. Нарисуйте многоугольник с координатами 5,135 115,5 225,135 и заливкой violet.
2. Нарисуйте пейзаж с пирамидами, используя `rect` и `polygon`. Цвета: `skyblue`, `yellow`, `goldenrod`, `darkgoldenrod`. Все значения кратны десяти.



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.

Задание 4. Окружности

1. Нарисуйте окружность с радиусом 10 и центром по горизонтали (**cx**) равным 50, по вертикали (**cy**) — 50%, цвет заливки — crimson.
2. Вторую окружность: радиус — 30, cx = 105, cy = 50%, цвет заливки — orangered.
3. третью: радиус — 50, cx = 200 и cy = 50%, цвет заливки — gold.

Задание 5. Мишень

Нарисуйте мишень, используя `circle` и `rect`. Все размеры и координаты кратны пяти.

Цвета: `yellowgreen`, `skyblue`, `white`, `crimson` и `black`.

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%"/>
```

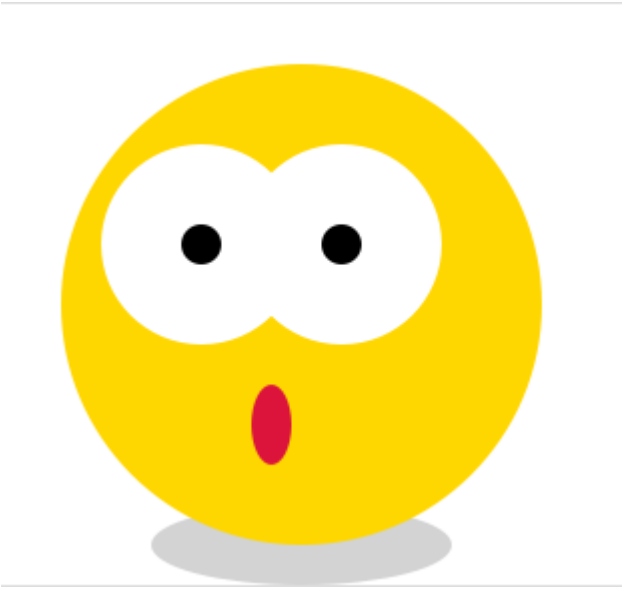
Расположение эллипса, так же, как и для circle, задаётся с помощью **cx** и **cy**.

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

Задание 6. Эллипс. Удивлённый смайлик

1. Нарисуйте эллипс с радиусом по горизонтали равным 40% и радиусом по вертикали — 30.
2. Переместите фигуру, задав cx значение 50% и cy значение 80%.
3. Добавьте зелёную заливку (yellowgreen).
4. Используя круги и эллипсы нарисуйте удивленный смайл.

Цвета: lightgray, gold, white, black, crimson. Все значения кратны пяти.



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.

Задание 7. Нарисуйте линию оранжевого цвета (orange) толщиной 10 пикселей с началом в точке (20,20) и концом в точке (200,100).

Задание 8. арифметические знаки

Используя линии, нарисуйте арифметические знаки.

Цвета: teal, orangered, royalblue, purple и orange.

Все значения кратны пяти.

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):

Задание 8.

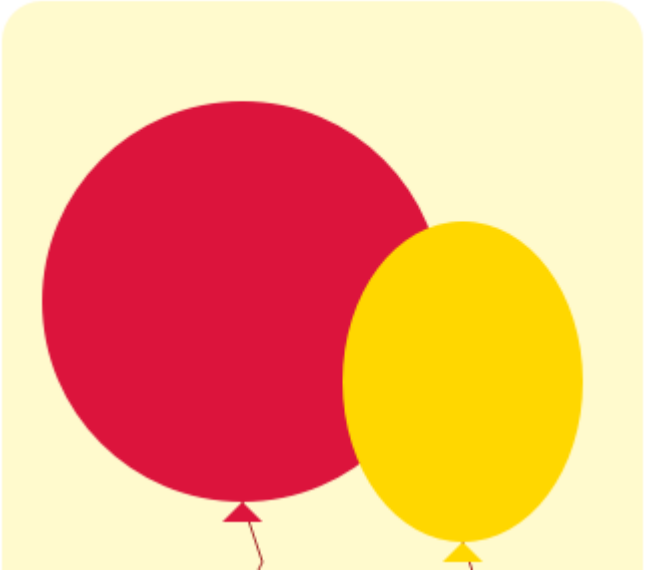
1. Нарисуйте линию с координатами `50,180 100,20 150,180 20,80 180,80 50,180 100,20`.
2. Добавьте ей обводку цвета `red` толщиной `5` пикселей. Чтобы убрать чёрную заливку, задайте атрибуту `fill` значение `none`.

Задание 9. Воздушные шары

Нарисуйте открытку с воздушными шарами,
используя rect, circle, polygon, ellipse и polyline.

Цвета: `lemonchiffon`, `crimson`, `gold`, `brown`.

Все значения кратны пяти.



Заливки

Рассмотрим возможности оформления векторных фигур 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:

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

Вот так будет выглядеть действие этого кода в браузере:

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:

Цвет можно задавать в любом удобном формате.

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:

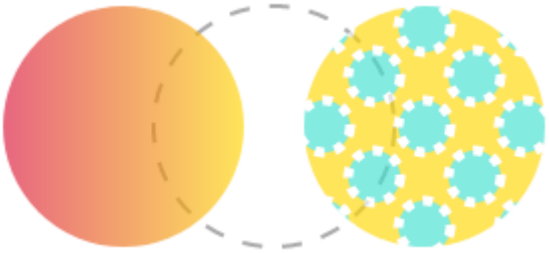
HTML

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

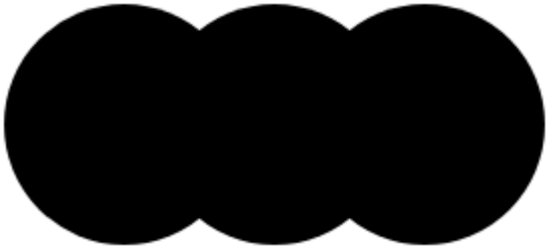
CSS

```
rect {
  fill: gold;
  fill-opacity: 0.5;
}
```

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



Задание 10. В html разметке добавить заливку в элементы сложной фигуры



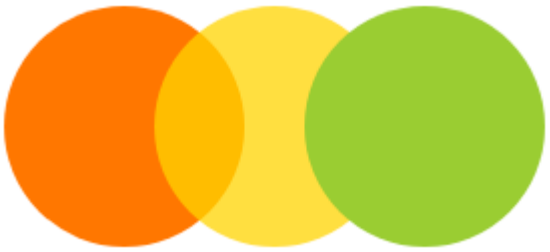
HTML

```
<html>
 <head>   </head>
 <body>
  <svg width="300" height="150">
    <circle r="60" cx="75" cy="50%"></circle>
    <circle r="60" cx="150" cy="50%"></circle>
    <circle class="shape-opacity" r="60" cx="225" cy="50%"></circle>
  </svg>
 </body>
<html>
```

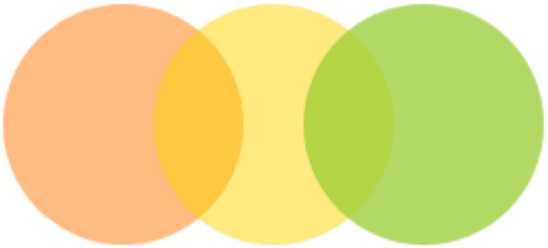
CSS

```
svg {
  border: 1px solid #dddddd;
}
. shape-opacity {
 ??????????????
}
```

1. Первой фигуре задайте заливку атрибутом в формате HEX `fill="#ff7700"`
2. Второй — атрибутом в формате HSLA `fill="hsla(50, 100%, 50%, 0.75)"`
3. Третьей фигуре задайте заливку именованным цветом `fill: yellowgreen` в редакторе CSS.



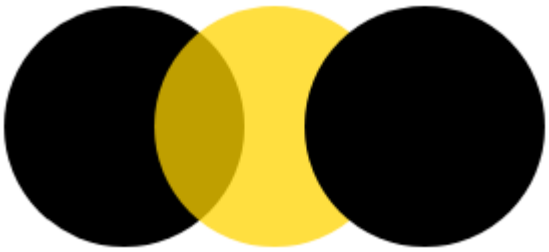
4. Задайте первой фигуре прозрачность с помощью атрибута fill-opacity="0.5"
5. Задайте третьей фигуре прозрачность с помощью CSS `fill-opacity: 0.75`



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.

Задание 11. Уберите заливку фигур в HTML и CSS кодах рисунка



Исходный код

svg2.HTML

```
<html>

<head>   </head>
<body>
<svg width="300" height="150">
  <circle r="60" cx="75" cy="50%"></circle>

  <circle r="60" cx="150" cy="50%" fill="hsla(50, 100%, 50%, 0.75)"></circle>
  <circle class="shape-transparent" r="60" cx="225" cy="50%"></circle>
</svg>
</body>
<html>
```

svg2.CSS

```
svg {
  border: 1px solid #dddddd;
}

.shape-transparent {

}
```

1. Задайте первой фигуре отсутствие заливки с помощью `fill="none"`.
2. Сделайте то же самое для третьей фигуры, но через CSS, используя `fill: none`.

Результат исправления должен быть таким

Задание 12. разноцветные квадраты

Раскрасьте квадраты, используя цвета `tomato`, `lightseagreen` и `#ffd700`.

Заливка бирюзового квадрата должна быть полупрозрачной, нижний правый квадрат нужно сделать полностью прозрачным.

svg3.HTML

```
<html>
 <head>   </head>
 <body>
 <svg width="290" height="290">
   <rect class="rect-1" width="42.5%" height="42.5%" x="5%" y="5%"></rect>
   <rect class="rect-2" width="42.5%" height="42.5%" x="52.5%" y="5%"></rect>
   <rect class="rect-3" width="42.5%" height="42.5%" x="5%" y="52.5%"></rect>
   <rect class="rect-4" width="42.5%" height="42.5%" x="52.5%" y="52.5%"></rect>
 </svg>
 </body>
<html>
```

svg3.CSS

```
/* Используемые цвета: tomato, lightseagreen, #ffd700
*/
svg {
  border: 1px solid #dddddd;
}
```

Результат исправления должен быть таким

Обводки

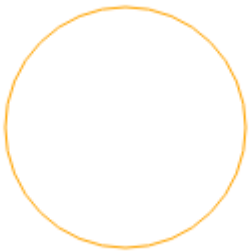
Обводки задаются с помощью нескольких атрибутов, причём цвет и толщина обводки задаются отдельно. Цвет задаётся атрибутом **stroke**:

```
<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;
}
```

Результат:

Короткой записи нет, поэтому цвет и толщина всегда задаются отдельно.

Если задавать значение в процентах, они будут рассчитываться не от размеров фигуры, а относительно размеров всего SVG, что может давать непредсказуемый результат.

Если обводке задана толщина, но не задан цвет, обводка не отобразится.

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;
}
```

Задание 13.

1. Задайте первой фигуре обводку атрибутом `stroke="darkorange"`.
2. Задайте второй фигуре обводку через CSS `stroke: teal`.

svg4.HTML

```
<html>
 <head>   </head>
 <body>
  <svg width="300" height="150">
    <circle r="60" cx="75" cy="50%" fill="gold"></circle>
    <circle class="shape-stroke" r="60" cx="225" cy="50%"></circle>
  </svg>
 </body>
<html>
```

svg4.CSS

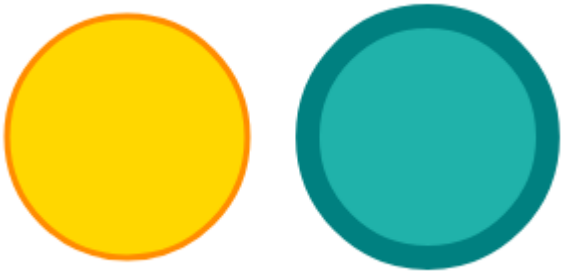
```
svg {
  border: 1px solid #dddddd;
}

.shape-stroke {
```

```
  fill: lightseagreen;
}
```

Результат исправления должен быть таким

3. Задайте первой фигуре толщину обводки с помощью атрибута `stroke-width="3"`.
4. Задайте второй фигуре толщину обводки через CSS `stroke-width: 5%`.

Обратите внимание, что, в отличие от HTML, рамка не влияет на положение фигуры в пространстве или на её размеры.



Размеры в 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.

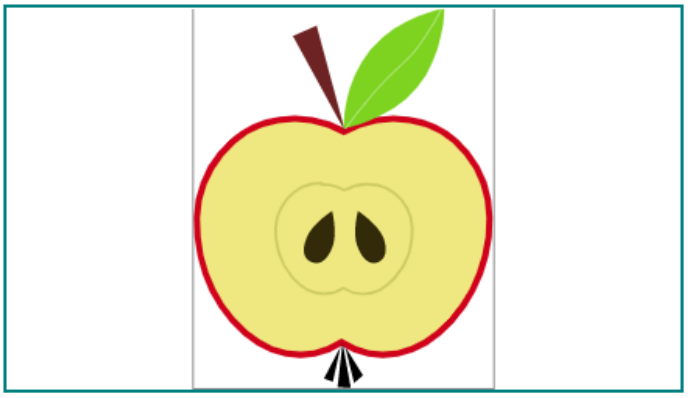
Атрибут viewBox

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.



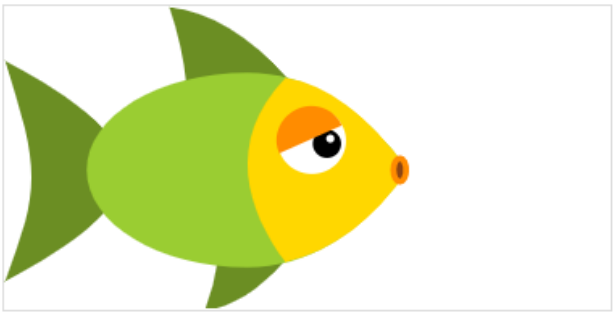
С вьюбоксом содержимое масштабируется, чтобы поместиться целиком в контейнер, и выравнивается по центру.

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.

Задание 14. Рыбка

Исходные размеры изображения — 200 на 150 пикселей.



```
<svg>
  <g id="fish">
    <path fill="olivedrab" d="M81.592.37c19.9 2.102 39.47 13.838 58.707 35.207L90.11 38
.462C88.937 25.884 86.097 13.186 81.592.37zM99.354 150c12.867-1.36 25.852-9.268 38.954-
23.727l-32.45-1.865c-1.423 8.774-3.59 17.305-6.504 25.592zM59.08 81.25c0 27.163-58.707
53.685-58.707 54.327C17.8 89.394 17.765 81.25.373 26.923c0 .952 58.707 27.164 58.707 54
.327z"/>
    <path fill="yellowgreen" d="M120.398 128.846c29.31 0 55.843-16.025 79.602-48.077-23
.76-32.052-50.293-48.078-79.602-48.078-43.963 0-79.602 21.525-79.602 48.077 0 26.55 35.
64 48.076 79.602 48.076z"/>
    <path fill="gold" d="M139.13 126.32c23.436-5.678 43.726-20.86 60.87-45.55-20.02-27.
17-40.293-42.348-60.818-45.537-30.57 31.603-18.83 65.88-.052 91.086z"/>
    <g transform="translate(129.353 43.27)">
      <ellipse cx="22.886" cy="23.077" fill="#FFF" rx="16.915" ry="16.346"/>
      <circle cx="30.348" cy="24.519" r="7"/>
      <circle r="2" cx="32" cy="22" fill="white"/>
      <path fill="darkorange" d="M37.683 15.49C34.01 7.243 24.116 3.637 15.58 7.437c-8.
533 3.8-12.475 13.566-8.803 21.813l30.906-13.76z"/>
    </g>
    <path fill="saddlebrown" stroke="darkorange" stroke-width="3" d="M195.788 86.538c1.
728 0 3.13-2.583 3.13-5.77 0-3.185-1.402-5.768-3.13-5.768-1.73 0-3.13 2.583-3.13 5.77 0
 3.185 1.4 5.768 3.13 5.768z"/>
  </g>
</svg>
```

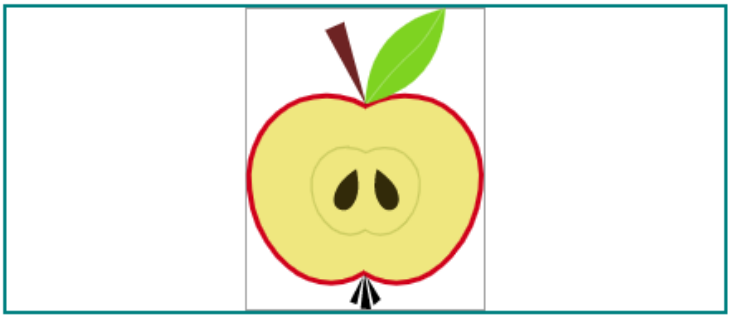
style.css

```
svg {
  background-color: #ffffff;
  border: 1px solid #dddddd;
}
```

Увеличьте рыбку до 400 на 300.

Атрибут preserveAspectRatio

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

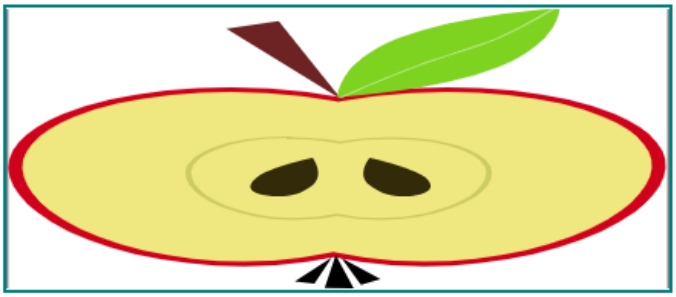


С помощью свойства preserveAspectRatio это поведение можно изменять: например, значение none указывает, что сохранять пропорции не нужно:

```
<svg viewBox="0 0 237 300" preserveAspectRatio="none">
  …
</svg>
```

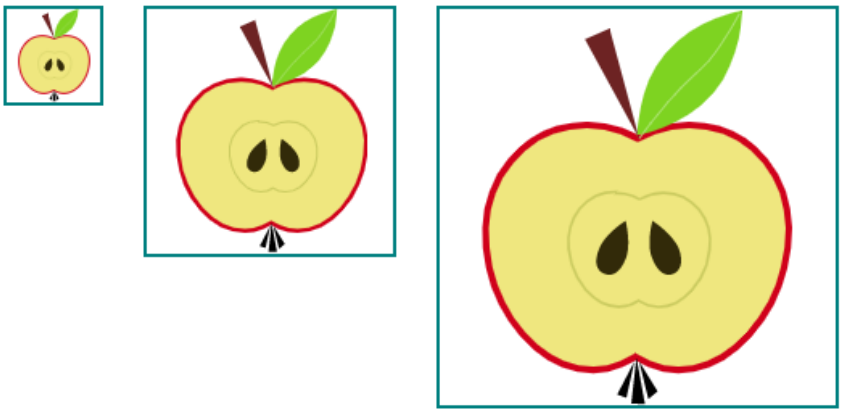
В этом случае область, размеры которой заданы вьюбоксом, растягивается на всё доступное пространство вьюпорта:

```
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:

Задание 15. резиновые полосы

Исходные размеры изображения — 140 на 140. Добавьте <svg> нужные атрибуты, чтобы при текущих размерах SVG-элемента полосы заполняли всё доступное пространство.

```
<svg width="500" height="250">
  <g id="colored-stripes">
    <path fill="tomato" d="M0 0h20v140H0z"/>
    <path fill="lightseagreen" d="M20 0h20v140H20z"/>
    <path fill="gold" d="M40 0h20v140H40z"/>
    <path fill="cornflowerblue" d="M60 0h20v140H60z"/>
    <path fill="darkorange" d="M80 0h20v140H80z"/>
    <path fill="yellowgreen" d="M100 0h20v140h-20z"/>
    <path fill="purple" d="M120 0h20v140h-20z"/>
  </g>
</svg>
```

style.css

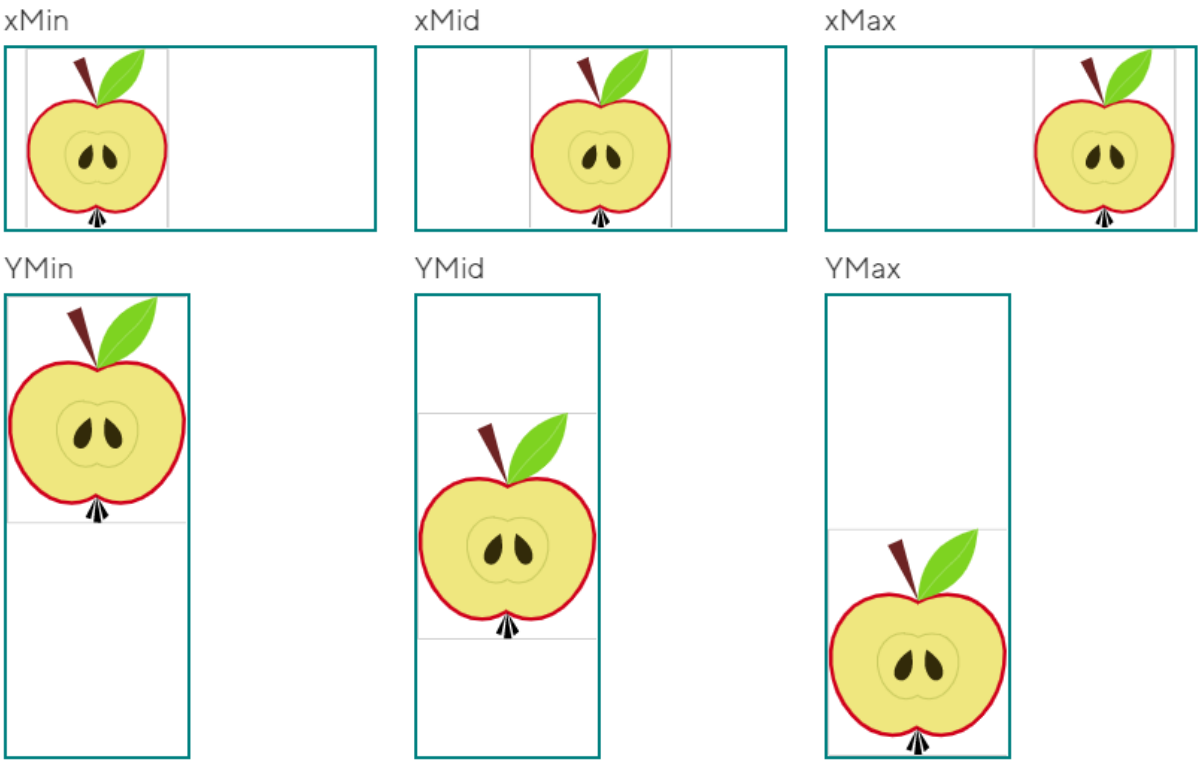
```
svg {
  background-color: #ffffff;
  border: 1px solid #dddddd;
}
```

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:



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.

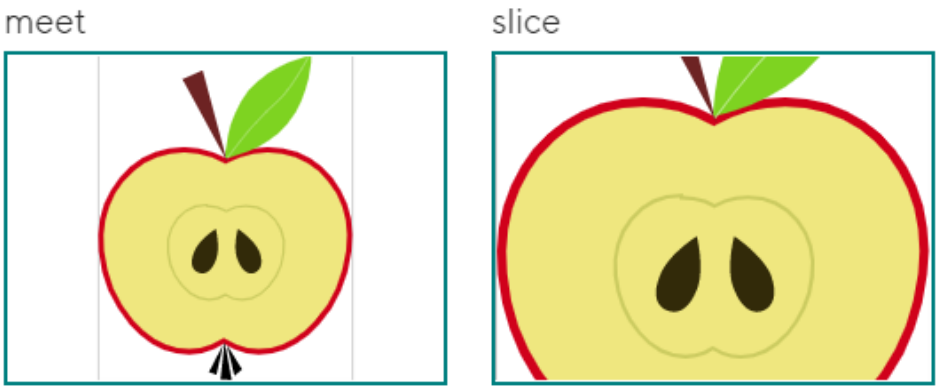
Также preserveAspectRatio не работает, если содержимое отрисовывается без полей (то есть соотношения сторон вьюпорта и вьюбокса совпадают), тогда в нём просто нет необходимости.

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 — 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.

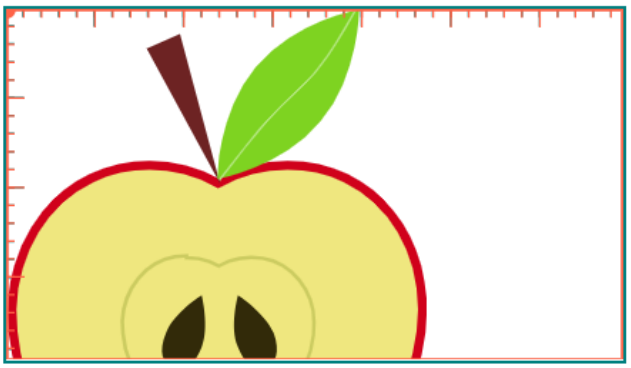
Системы координат

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:

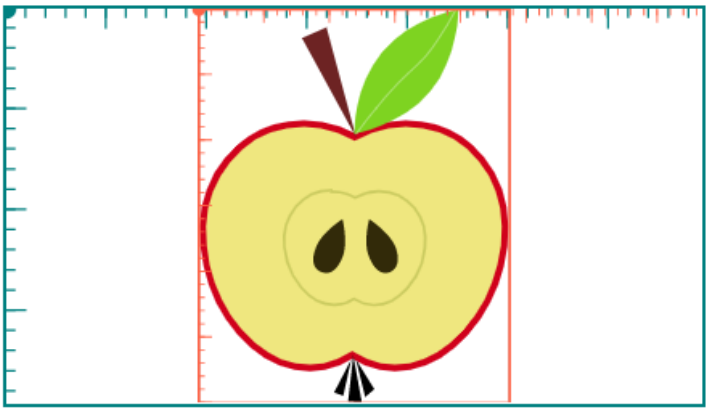
```
<svg width="350" height="200">
  …
</svg>
```



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.

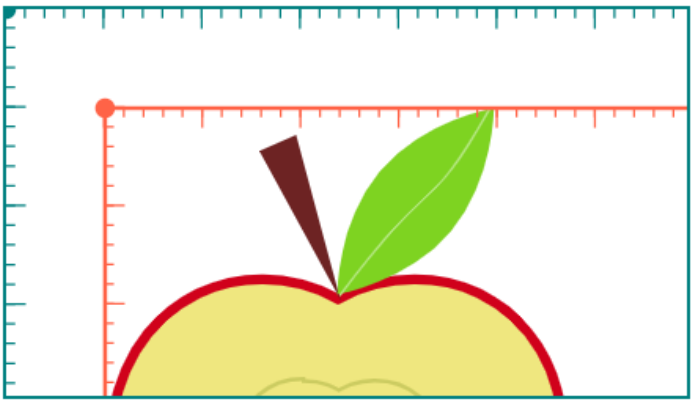
Трансформации тоже создают свою систему координат. Чтобы применить трансформацию ко всему содержимому, обернём его в группу (элемент `<g>`):

```
<svg width="350" height="200">
  <g>
    …
  </g>
</svg>
```

И добавим трансформацию:

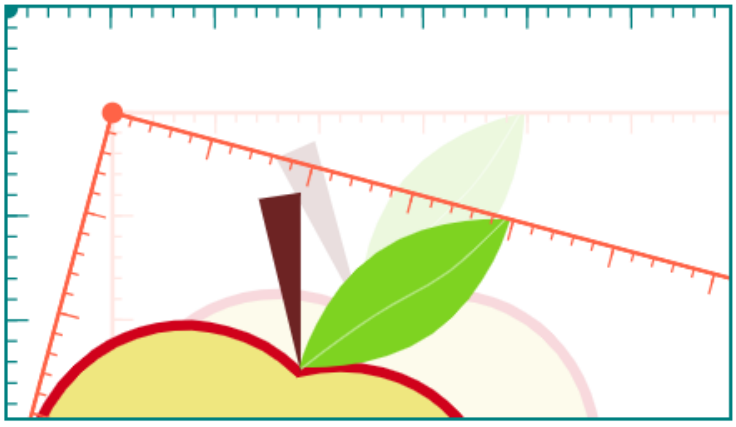
```
<svg width="350" height="200">
  <g transform="translate(50, 50)">
```

```
    …
  </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.

Домашнее задание. Решения всех заданий (1:15) необходимо собрать на один “пен” в своем аккаунте на codepen.io в виде отдельных блоков на веб-странице, и ссылку на этот документ прикрепить в соответствующее задание в курсе