***[SVG на собственной шкуре. Часть 1 — создание SVG файла в CorelDraw](https://blog.g63.ru/other/svg-%d0%bd%d0%b0-%d1%81%d0%be%d0%b1%d1%81%d1%82%d0%b2%d0%b5%d0%bd%d0%bd%d0%be%d0%b9-%d1%88%d0%ba%d1%83%d1%80%d0%b5-%d1%87%d0%b0%d1%81%d1%82%d1%8c-1-%d1%81%d0%be%d0%b7%d0%b4%d0%b0%d0%bd%d0%b8%d0%b5/" \o "Ссылка на запись SVG на собственной шкуре. Часть 1 — создание SVG файла в CorelDraw)***

В интернете полно руководств о том, как внедрить в HTML5 документ SVG графику. Ну конечно не только в HTML5. Во всех этих руководствах / статьях одно и тоже. Но реально они не рассказывают ни чего подробного. И большинство этих статей рассказывают о создание SVG объектов внутри кода страницы. И ни где нет подробного руководства о том, как подключить внешний файл на страницу. При этом чтобы он был как часть страницы, чтобы было полное взаимодействие с DOM, чтобы можно было манипулировать SVG файлом с помощью JS.

Долго изучал данный вопрос и вот моя статья, которая ни в коем случае не претендует на руководство, а просто рассказывает о моих экспериментах и поисках решения.

Свою сказку я пожалуй начну немного из далека

***Создание SVG файла в CorelDraw и параметры при сохранение.***

Я очень давно занимаюсь рекламой — полиграфия и наружка. В этой отрасли векторная графика очень развита в отличие от интернета. Большинство макетов делается непосредственно с применением **CorelDraw** или **Adobe Illustrator**, а макеты в **Photoshop** является плохим тоном.

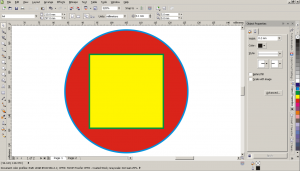
Вместе с [HTML5](http://blog.g63.ru/tag/html5/) и [CSS3](http://blog.g63.ru/tag/css3/) нам теперь доступна и полноценная поддержка векторной графики во всех современных браузерах… Ну ладно, ладно — почти полноценная, но не за горами то время когда она будет полноценной. И уже сейчас смело можно использовать SVG в разработке сайтов, уже сейчас SVG графика может заменить некоторые ниши FLASH… Но это уже не по теме, это очередной холивар для отдельной темы, о которой я возможно пофлужу в отдельной теме.

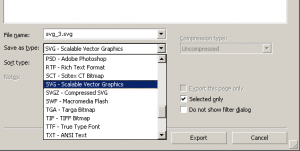
И так, ближе к делу. Как я писал чуть выше — я очень долго работал с векторной графикой и её внедрение в HTML для меня очень радостное событие. Все свои файлы я создаю в **CorelDraw X5**. Он корректно сохраняет SVG. Конечно есть и бесплатные аналоги, такие как **Inkscape**, но:

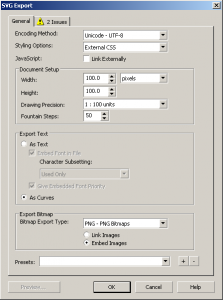
* Я им не пользуюсь. Хоть он и самый мощный аналог Corel’a и Illustrator’а из бесплатных программ и я его уважаю за то, что он OpenSourse, но он еще очень молодой и слабый по сравнению со своими старшими платными собратьями.
* В интернете много написано о том, что Inkscape сохраняет не корректно SVG. Хотя есть инструмент, который исправляет его ошибки правит, но [об этом ниже](https://blog.g63.ru/other/svg-%D0%BD%D0%B0-%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9-%D1%88%D0%BA%D1%83%D1%80%D0%B5-%D1%87%D0%B0%D1%81%D1%82%D1%8C-1-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5/#optimize).

Дальше я приведу небольшое руководство по сохранению файла из CorelDraw X5. В Illustrator’е и Inkscap’е я не пробовал сохранять, но процесс наверняка схожий.

*Сначала дам один совет по оптимизации самого изображения — избегать****Behind fill****(контур под объектом) у контуров. Corel при экспорте в SVG будет создавать дубликат объекта. В таких случаях лучше сделать толщину контура в два раза меньше. Если это не возможно, то лучше конвертировать контур в объект и ручками удалить лишние узлы.*

Для примера в CorelDraw X5 я создал два простых примитива — круг и квадрат с контурами.  
[](http://blog.g63.ru/wp-content/uploads/2011/07/svg_1.png)  
Сохраняем нарисованное в SVG. Сохранять можно через «Export» или «Save As». Какой способ выбрать — значения не имеет.  
[](http://blog.g63.ru/wp-content/uploads/2011/07/svg_2.png)

В появившемся диалоговом окне Экспорта / Сохранения выбираем формат SVG.  
[](http://blog.g63.ru/wp-content/uploads/2011/07/svg_3.png)

Жмём «Сохранить» (ну или «Экспорт» — в зависимости от выбранного вами способа) и перед нами появится окно настроек SVG файла.  
[](http://blog.g63.ru/wp-content/uploads/2011/07/svg_4.png)

***Encoding Method***

Оставляем как есть в **Unicode — UTF-8**

***Styling Options***

Первым делом решим — хотим ли мы что бы стили были внутри SVG файла или были вынесены в отдельный файл. Тут принцип работы ни чем не отличается от работы стилей в HTML+CSS.

В большинстве случаев стили лучше встраивать в файл. Для этого необходимо выбрать пункт **Internal Style Sheet** в строке **Styling Options**. Этот параметр создаст SVG файл в хедере которого будут записаны все стили файла — цвета, контуры и их цвета.

Если в документе нет повторяющихся цветов и прочих стилей, то имеет смысл встроить их непосредственно в объекты. Для этого выбирайте пункт **Presentation Attributes**. Оба эти способа помогут сделать файл максимально независимым:

* При передаче — вы будете передавать только 1 файл, что не позволит вам случайно забыть файл стилей и запутаться в файлах.
* При внедрения файла в страницу — файл всегда будет использовать встроенные стили и будет производится только один запрос к серверу — не будет лишних запросов при загрузке файла.

Параметр **External CSS** позволит сохранить стили в отдельный файл. К сожалению, я так и не смог добиться раскрашивания SVG файла с помощью глобальных стилей документа. Но это с одной стороны понятно, а с другой очень огорчает, т.к. SVG файл встраивается в DOM страницы, но и в тоже время не воспринимает глобальные стили, тем самым лишая многих возможностей по оформлению интерфейса и не только. Хотя с помощью JS управлять файлом всё равно можно. Об этом я расскажу в следующей статье. Так что данный параметр полезен разве только для презентаций, когда у нескольких файлов очень много общих стилей.

Если Вы не поняли что я написал в последних  четырёх абзаца — поэкспериментируйте сами с этим параметром и Вы всё сами поймете. SVG файл можно легко открыть в блокноте и посмотреть или отредактировать код. Это же обычный XML файл.

***JavaScript***

Следующий немаловажный пункт (checkbox) — это прикрепление или внедрение JS. Тут всего два параметра «выносить js код в отдельный файл» или «внедрить его в документ». По умолчанию он внедряется в код файла, т.е. галочка снята. Но т.к. в примере я не используем ни каких JS, то этот пункт мне не особо интересен.

***Document Setup***

Далее идут настройки документа — **Document Setup**.  
Ну про ширину и высоту думаю рассказывать не имеет смысла. Тут и так всё понятно. Главное не забудьте выставить единицу измерения **pixsels**  
  
А вот **Drawing Precision** очень полезная штука. Тут мы задаем масштаб в котором будет обрабатываться документ, главным образом координаты узлов и объектов. В этом параметре я всегда выставляю разное значение. Вы можете поиграться и выставить свои. Но желательно добиться что бы у вас координаты были типа ***12.435567*** или ***213.078381***, т.е. 2 или 3 разряда. Зачем? Про это будет [рассказано чуть ниже](https://blog.g63.ru/other/svg-%D0%BD%D0%B0-%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9-%D1%88%D0%BA%D1%83%D1%80%D0%B5-%D1%87%D0%B0%D1%81%D1%82%D1%8C-1-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5/#optimize).

Следующий пункт **Fourtain Steps** оставим по умолчанию.

***Export Text***

Далее нам предлагается выбрать, как мы желаем сохранить текст — **Export Text**  
В моем примере текста нет, поэтому он меня не касается, но всё же поясню.

Если у Вас в документе есть текст и он использует стандартные системные шрифты, то текст стоит сохранить «как текст» (**As Text**) со всеми снятыми галочками у этого параметра.

Если же вы используете не стандартные шрифты, то рекомендую сохранить текст в кривых (**As Curves**). Это позволит избавится от лишних файлов со шрифтами.

***Export Bitmap***

Следующим пунктом идёт **Export Bitmap**, в котором вы можете выбирать формат растровых изображений, если конечно они имеются. Здесь выбирайте на свой вкус. Моё мнение такое

* Если в файле много фотографий, то следует выбрать формат JPG. Хотя я не представлю зачем в SVG файле может быть использовано много фотографий. Если для галереи или чего то подобного, то это не оправдано, учитывая что сейчас для таких целей существует множество разных галерей на JS и фрейморках.
* Для остальных целей необходимо использовать PNG или GIF. Хотя такие картинки чаще проще отрисовать. Не спроста же мы решили использовать SVG.
* Если вы используете много эффектов, которые Corel будет растрировать при сохранение, то стоит выбрать PNG.

Так же выбирайте на своё усмотрение прикреплять файлами картинки или внедрять (**Link Images** / **Embed Images**). Если в файле много картинок и все они крупные, то лучше будет прикреплять их файлами. Если в файле несколько картинок или чуть больше чем несколько, но они все маленькие, то лучше внедрить.

Хотелось бы напомнить, что SVG файл легко может редактироваться даже в блокноте. Хотя не рекомендую обычный виндовский блокнот, лучше использовать редактор **Notepad++** или аналог в котором используется подсветка кода и который может корректно работать с **UTF-8**. И если вы прикрепите изображения как файлы, а не внедрите, то их потом лучше сохранить отдельно в соответствующие форматы, используя советы выше — фотки в JPG, а остальные картинки в PNG и GIF и подправить пути до файлов ручками.

***Presents***

И последний пункт в окне параметров сохранения SVG — это **Presents**. В этом пункте разработчики CorelDraw создали для нас несколько предустановок настроек, а также предоставили возможность сохранить свои часто используемые настройки.

***Оптимизация SVG файла и его сжатие***

Не буду изобретать велосипед и рассказывать то, что уже есть в интернете.  
На сайте **xiper.net** есть хороший материал на эту тему: [Оптимизация SVG. Компрессия кода.](http://www.xiper.net/collect/web-without-flash/svg/compression-SVG-code.html). Там же есть [браузерная версия SVG-оптимизатора](http://www.xiper.net/collect/services/graphics/svg-compressor.html). Она то нам и понадобится.

В принципе далее будет описана процедура сжатия, которую я уже описал в комментариях выше озвученной статьи. Но всё же повторю её здесь.

На момент написания статьи оптимизатор с сайта xiper.net не работал корректно, поэтому необходимо действовать так:

* Сохраняем SVG файл из Corel’а, но не забудьте настроить масштаб — **Drawing Precision**. Именно теперь она нам понадобиться и будет играть одну из ключевых ролей.
* Далее переходим на страницу [оптимизатора](http://www.xiper.net/collect/services/graphics/svg-compressor.html):
  + В поле «Исходный файл» выбираем созданный нами файл
  + В поле «Точность чисел в атрибуте «d» тега «path»» ставим 0. Это нам позволит оптимизировать файл за счет удаления всех значений после точки. **Именно для этого я советовал, выше в статье,** [отстроить масштаб (Drawing Precision)](https://blog.g63.ru/other/svg-%D0%BD%D0%B0-%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9-%D1%88%D0%BA%D1%83%D1%80%D0%B5-%D1%87%D0%B0%D1%81%D1%82%D1%8C-1-%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5/#drawingPrecision).
  + Поставьте галочку «Форматировать XML», а **галочку с «Сжать gzip» снимите**, если она стоит. Дело в том, что **на момент написания статьи, оптимизатор работал некорректно**. Возможно к моменту прочтения статьи оптимизатор уже будет исправлен (я автору сообщил в комментариях к его статьи об ошибках оптимизатора :-)) и тогда можете поставить галочку «Сжать gzip» и пропустить следующий пункт.
* Жмем «Получить файл» и сохраняем его.
* Открываем наш файл и удаляем в нём ошибку, которую подарил нам оптимизатор:
* <**br** />

<**b**>Notice</**b**>: Undefined index: P in <**b**>/home/clients/ksayri\_ftp0/domains/xiper.net/html/php/services/svg-compressor/scripts/svgcrush.php</**b**> on line <**b**>251</**b**><**br** />

* Следующим шагом у нас будет уже сжатие (архивирование) файла. Да-да, именно архивирование, т.к. все браузеры которые поддерживают **SVG** — поддерживают и **SVGZ** — сжатые **SVG** с помощью **GZip**. Только обращаю Ваше внимание, что **FireFox пока что не может открывать SVGZ локально!**.

На этом всё — файл готов к употреблению. В случае правильно выполненных советов файл может потерять до 90% от своего веса 🙂

***UPDATE (02.11.2011)***

Я сразу не обратил внимания — оказывается CorelDraw умеет сохранять сразу SVGZ. Я бы всё же не рекомендовал этого делать из-за приведенных мной выше нюансов. Ручками все же надежнее корректировать выходной файл. Но если вам надо быстро вывести файл, например для тестов, то SVGZ в CorelDraw самое то, ни надо ни каких заморочек. Однако файл, повторю, по выше предложенной схеме получается меньшего веса, а иногда и в несколько раз.

***UPDATE 2 (02.11.2011)***

Как известно — без мозгоёбства ни чего не бывает. В этот раз сюрприз преподнёс FF. Он не захотел отображать корректно **SVGZ**, а точнее просто ругался и выдавал ошибку

*Ошибка синтаксического анализа XML: некорректно  
Адрес: http://demo.g63.ru/animation\_css3\_and\_svg/animation.svgz  
Строка 1, символ 1:  
­*

Поэтому на данный момент рекомендация следующая:

* Не использовать выше предлагаемый онлайн оптимизатор для сжатия в **SVGZ**
* Использовать локальный упаковщик **GZIP** (GZ). Не путать с **ZIP**. Например **TotalCommander** или **7Zip**. Только не забыть переименовать расширение файла в **SVGZ**.
* Или использовать прямое сохранение из **CorelDraw**. В **InkScape** так же есть такая возможность. Однако файл будет немного больше по размеру из-за отсутствия ручной оптимизации.
* Для надежности создать на сервере файл **.htaccess** или добавить в него, если этот файл есть, следующие параметры
* **AddType** image/svg+xml svg svgz

**AddEncoding** gzip svgz

***UPDATE 3 (26.12.2011)***

В FF также появляется выше приведенная ошибка, если в файле нет отбивки после первой строки, объявляющей формат файла.  
Т.е. вот так не правильно:

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

А вот так правильно:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"