

Неофициальный перевод

# **RU Ai SCRIPTING GUIDE CC 2017**



alexmaj

Версия 0.0.1 beta

Данный документ является неофициальным переводом Adobe Illustrator Scripting Guide CC 2017.

С одним ограничением, описание только JavaScript, в нём нет глав для языков AppleScript и VBScript.

Оригинал находится по ссылке на сайте adobe.com

[AI\\_ScriptGd\\_2017.pdf](#)

Web версия оригинала по ссылке

[ai-scripting.docsforadobe.dev](#)

Перевод на Русский на данный момент сырой, возможны ошибки в тексте и неправильная трактовка в нескольких местах.

Также есть вопросы, если в тексте указаны пункты меню программы Illustrator, их оставлять в Оригине не трогая, или же писать так как в Русифицированной версии Иллюстратора.

Если заметили ошибки или неправильную трактовку, прошу сообщить в комментариях к файлу или на почту alexmaj467@yahoo.com

*Есть желание перевести и остальную документацию по Скриптам, но пока точно не понятно, в таком виде + исправление со временем, оно нужно ?*

С любыми предложениями на почту alexmaj467@yahoo.com

Желающие помочь финансово.

***Donate***



		3
1	Вступление	5
	Что такое сценарии (скрипты)?	5
	Зачем использовать сценарии?	5
	Сценарии и Операции	5
	Языки сценариев поддерживаемые в <i>Adobe Illustrator CC 2017</i>	6
	Какое расширение используют файлы сценариев	6
	JavaScript варианты разработки	6
	Примеры сценариев	7
	Просмотр объектной модели	7
	JavaScript объектная модель	8
	AppleScript объектная модель	8
	VBScript объектная модель	8
	Выполнение сценариев	9
	Установка сценариев в меню Scripts	9
	Выполнение сценариев из меню Другие сценарии	9
	Автозапуск сценариев (только .jsx сценарии)	9
	Изменения в CC	10
	Перечисления и константы	10
	Методы и свойства	11
	Ответы на вопросы	11 - 12
2	Объектная модель сценариев Illustrator	12
	Соглашение об именовании объектов	13
	Объекты верхнего уровня	13
	Приложение	13
	Документ	13 - 14
	Слой	14
	Дерево арт - объектов	14
	Графические стили	15
	Цвет	15
	Текст	15 - 16
	Текстовый фрейм	16 - 17
	Объекты предоставляющие текстовое содержимое	17 - 18
	Стили текста	18
	Динамические объекты	18 - 19
	Символы	19
	Трансформация	19
3	Сценарии Illustrator	19
	Запуск и выход из Illustrator посредством скрипта	20
	Запуск и активация Illustrator	20
	Выход из Illustrator	21
	Работа с объектами	21
	Получение самого переднего объекта или слоя	21
	Создание новых объектов	21 - 22
	Объекты коллекции	22
	Выбранные объекты	23
	Примечание по переименованию объектов, хранящихся в панелях приложения	23
	Единицы измерения	23
	Em относительные единицы	24
	Page-item размеры и расположение	24
	Границы арт - объекта	24 - 25
	Пути и фигуры	25
	Уровни взаимодействия с пользователем	25 - 26
	Печать документов	26 - 27

4	Сценарии на языке JavaScript	27
	Для дополнительной информации	27
	Ваш первый сценарий	27
	Добавление функции в "Hello World"	28
	Работа с методами в JavaScript	28 - 29
	Доступ к объектам или ссылки на них	29
	Ссылка на объект приложения	29
	Доступ к объектам коллекции	29 - 30
	Создание новых объектов	30
	Работа с выбранными объектами	31
	Работа с текстовыми фреймами	31
	Связь фреймов в поток	31
	Создание путей и фигур	32
	Путь	32 - 33
	Фигура	33
	Работа с сеткой перспективы	34
	Предварительные настройки	34
	Показать или скрыть сетку	34 - 35
	Установка активной плоскости	35
	Рисуем в сетке перспективы	35 - 36
	Перевести объекты в перспективу	36

# 1 Вступление

В этом руководстве описывается интерфейс сценариев для *Adobe Illustrator CC 2017*. Если вы новичок в написании сценариев и хотите получить базовую информацию о сценариях и о том, как использовать различные языки сценариев см. *Adobe Introduction to Scripting*. (Adobe Введение в создание сценариев).

## Что такое Scripting ?

Сценарий (script) - это серия команд, которые указывают *Illustrator* выполнить одну или несколько задач. Эти задачи могут быть простыми, затрагивающими только один объект в текущем документе, или сложными, влияющими на объекты во всех ваших документах *Illustrator*.

Задачи могут включать даже другие приложения, такие как текстовые редакторы, электронные таблицы, и программы управления базами данных.

По большей части строительные блоки сценариев соответствуют инструментам, меню, панелям и диалоговым окнам *Illustrator*, с которыми вы уже работали в программе. Если вы знаете, что нужно делать в *Illustrator*, вы можете написать сценарий, для этого.

## Зачем использовать сценарии?

Графический дизайн - это область, характеризующаяся творчеством, но аспекты работы совсем не творческие. На самом деле, вы, вероятно, заметили, что время, которое вы тратите на размещение и замену изображений, исправление ошибок в тексте и подготовку файлов, настройку изображений к печати, сокращает время, которое вы могли потратить на творческую работу.

Приложив небольшие затраты времени и усилий, вы можете научиться писать короткие, простые сценарии, выполняющие повторяющиеся задачи за вас. По мере роста ваших навыков написания сценариев вы можете переходить к более сложным сценариям.

Сценарии также могут повысить ваш творческий потенциал, быстро выполняя задачи, на которые у вас не хватало времени. Например, вы можете написать сценарий для систематического создания серии объектов, изменяя новые объекты их позицию, свойства обводки и заливки. Вы также можете написать сценарий, который обращается к встроенным матричным функциям преобразования для растягивания, масштабирования и искажения серии объектов. Без сценариев вы, скорее всего, упустили бы творческий потенциал таких трудоемких методов.

## Сценарии и Операции (Actions)

И операции, и сценарии - это способы автоматизации повторяющихся задач, но они работают по - разному:

- Операции используют пользовательский интерфейс программы для выполнения своей работы. По мере выполнения операции выполняются пункты меню, выбираются объекты, и создаются записанные пути. Сценарии не используют пользовательский интерфейс программы для выполнения задачи, и сценарии могут выполняться быстрее, чем операции.
- Операции имеют очень ограниченные возможности для получения информации и реагирования на нее. Вы не можете добавить условную логику в операцию; следовательно, операции не могут принимать решения на основе текущей ситуации. Например, изменение типа обводки прямоугольников, но не эллипсов. Сценарии могут получать информацию и принимать решения и производить расчеты на основе информации, которую они получают от *Illustrator*.

Сценарий может выполнять операции, но операции не могут выполнять сценарий.

## Языки сценариев поддерживаемые *Adobe Illustrator CC 2017*

*Illustrator* поддерживает *VBScript* и *JavaScript* сценарии для *Windows*, а также *AppleScript* и *JavaScript*, сценарии для *Mac OS*.

### Какое расширение используют файлы сценариев

Чтобы файл был распознан *Illustrator CC 2017* как допустимый файл сценария, он должен иметь правильное расширение имени файла:

Script Type	File type (extension)	Platforms
AppleScript	compiled script ( .sct ) OSAS file (no extension)	Mac OS
JavaScript or ExtendScript	text ( .js or .jsx )	Windows Mac OS
VBScript	text ( .vbs )	Windows

### JavaScript варианты разработки

Вы можете использовать *ExtendScript Toolkit* для создания сценариев *JavaScript* для *Illustrator*, или использовать *Adobe Extension Builder* и *Creative Cloud SDK* для разработки расширений в *ActionScript*. Расширения на основе *Flash (SWF)* потенциально могут работать в различных *Creative Cloud* приложениях.

#### Разработка расширения CC с использованием *ActionScript*

Приложения *Creative Cloud* имеют расширяемую инфраструктуру, которая позволяет разработчикам расширять возможности приложений; инфраструктура основана на технологии *Flash / Flex*, и каждое расширение поставляется в виде скомпилированного файла *Flash (SWF)*. *Creative Cloud* включает в себя *Extension Manager* позволяющий устанавливать расширения.

Примером расширения, поставляемого с указанными продуктами, является *Adobe Kuler*. У *Kuler* есть постоянный пользовательский интерфейс для различных приложений пакета, но имеет разную логику в каждом, адаптированную к ведущему – приложению.

Пользовательский интерфейс расширения написан на *ActionScript* с использованием инфраструктуры *Flex*. Обычно доступ к нему осуществляется через отдельный пункт меню расширения приложения. *Adobe Extension Builder* позволяет интерактивно разрабатывать пользовательский интерфейс с помощью представления «Дизайн» *Flash Builder*. *Creative Cloud SDK* также позволяет разрабатывать всю логику приложения для вашего расширения в *ActionScript*; вы можете разрабатывать и отлаживать свое расширение в знакомой среде *Flash Builder*.

Для разработки логики вашего приложения мы рекомендуем использовать *ActionScript Wrapper Library* (обёрточная библиотека) (**CSAWLib**), которая представляет DOM сценарии каждого ведущего приложения в виде библиотеки *ActionScript*. Это плотно интегрировано со средой *Adobe Extension Builder*, который включает мастера, помогающего вам создать свою базовую структуру вашего расширения, а также запускать и отлаживать свой код с помощью приложений пакета, таких как *Adobe InDesign*, *Photoshop* и *Illustrator*.

Методы, свойства и поведение скриптовой модели DOM описаны в разделе *JavaScript Scripting Reference* для каждой программы пакета. Подробнее об использовании *Adobe Extension Builder* и *wrapper libraries*, см. документацию *Creative Cloud SDK*, доступную из *Adobe Extension Builder*.

#### Сценарии и подключаемые модули (*Scripting plug-ins*)

Интерфейс сценариев CC *JavaScript* допускает ограниченное создание сценариев для подключаемых модулей. Плагин может определять команду с событием и уведомителем, а также обработчик, который выполняет какое-либо действие. Сценарий *JavaScript* может использовать метод `app.sendScriptMessage()` для отправки

параметров в эту команду, определённую подключаемым модулем, и получения ответа, определённого подключаемым модулем.

Например, плагин *Adobe Custom Workspace* определяет команду " *Switch Workspace*". Сценарий может вызвать эту команду с помощью следующего кода:

```
result = app.sendScriptMessage (
    "Adobe Custom Workspace" ,
    "Switch Workspace",
    '<workspace="Essentials" >'
);
```

В этом случае значение, возвращаемое подключаемым модулем, представляет собой строку

```
"<error=errNo>".
```

## Возможности *ExtendScript*

Если вы пишете сценарии для *Illustrator*, которые напрямую используют *DOM JavaScript* *Illustrator*, вы создаёте файлы *ExtendScript*, которые отличаются расширением **.jsx**.

Создание файлов *JavaScript* в формате **.jsx** (а не стандартное **.js** для файла *JavaScript*) позволяет использовать преимущества функций и инструментов *ExtendScript*.

*ExtendScript* предлагает все стандартные функции *JavaScript*, а также среду разработки и отладки *ExtendScript Toolkit (ESTK)*.

*ESTK* устанавливается со всеми скриптовыми приложениями *Adobe* и используется по умолчанию (редактор файлов **JSX**). *ESTK* включает в себя средство просмотра объектной модели, которое содержит полную документацию по методам и свойствам объектов *JavaScript*.

Для получения информации о доступе к *ESTK* и объектной модели, см. "JavaScript объектная модель" стр. 8.

*ExtendScript* также предоставляет различные инструменты и утилиты, в том числе следующие:

- Утилита локализации.
- Инструменты, позволяющие комбинировать сценарии и направлять их в конкретные приложения.
- Независимое от платформы представление файлов и папок.
- Инструменты для создания пользовательских интерфейсов к вашим скриптам.
- Платформа обмена сообщениями, которая позволяет отправлять и получать сценарии и данные среди поддерживающих сценарии приложений *Adobe*.

Все эти функции доступны независимо от того, используете ли вы *DOM* напрямую с файлом *JSX* или косвенно через библиотеку - оболочку *ActionScript* и *Adobe Extension Builder*.

Подробнее об этих и других функциях см. *JavaScript Tools Guide*.

## Примеры сценариев

*Adobe* предоставляет образцы сценариев для объектов, свойств и методов в *Illustrator CC DOM*.

Их можно просмотреть в двух местах:

- */Scripting/Sample Scripts* - папка установки *Illustrator CC*
- В справочнике по сценариям *Adobe Illustrator CC* для вашего языка сценариев, который можно загрузить <http://www.adobe.com/devnet/illustrator/scripting/>

## Просмотр объектной модели

Каждый из поддерживаемых языков сценариев предоставляет возможность просмо-



тра объектов сценариев, со справочной информацией, определенной *Illustrator*.

## JavaScript объектная модель

Чтобы просмотреть объектную модель *JavaScript* для *Illustrator*, выполните следующие действия:

При установке *Adobe* по умолчанию *ESTK* находится в следующем месте:

Windows	<code>\system drive\Program Files\Adobe\Adobe Utilities CC\ExtendScript Toolkit CC</code>
Mac OS	<code>\system drive\Applications\Utilities\Adobe Utilities CC\ExtendScript Toolkit CC</code>

1. Старт *ESTK*.
2. В *ESTK*, выберите *Help > Object Model Viewer*.
3. В окне *Object Model Viewer*, выберите *Adobe Illustrator CC Type Library* в выпадающем списке проводника.

Несколько расширенных примеров сценариев доступны в папке */Scripting/Sample Scripts* каталог установки *Illustrator CC*.

Вы также можете просматривать образцы сценариев и информацию об отдельных классах, объектах, свойствах, методах, и параметры в *Adobe Illustrator CC Scripting Reference: JavaScript*, которые можно скачать <http://www.adobe.com/devnet/illustrator/scripting/>.

## AppleScript объектная модель

*Apple* предоставляет редактор сценариев для всех систем *Mac OS*. Вы можете использовать редактор сценариев для просмотра *AppleScript* словаря, описывающего объекты и команды *Illustrator*.

Дополнительные сведения, см. *Script Editor Help*.

### Примечание:

При установке *Mac OS* по умолчанию редактор сценариев находится в *Applications:AppleScript:Script Editor*. Если вы не можете найти приложение *Script Editor*, вам необходимо переустановить его из вашей системы *Mac OS CD*.

1. *Start Script Editor*.
2. Выберите *File > Open Dictionary*. В редакторе сценариев отображается диалоговое окно *Open Dictionary*.
3. *Open Dictionary dialog*, найдите и выберите *Adobe Illustrator CC*, нажать *Open*. *Script Editor* отображает список объектов и команд *Illustrator*, которые включают свойства и элементы, связанные с каждым объектом, и параметры для каждой команды.

Несколько расширенных примеров сценариев находятся в папке *:Scripting:Sample Scripts* каталог установки *Illustrator CC*.

Вы также можете просматривать образцы сценариев и информацию об отдельных классах, объектах, свойствах, методах, и параметры в *Adobe Illustrator CC Scripting Reference: AppleScript*, которые можно скачать <http://www.adobe.com/devnet/illustrator/scripting/>.

## VBScript объектная модель

*VBScript* предоставляет библиотеку типов, которую вы можете использовать для просмотра свойств и методов объектов *Illustrator*. Эта процедура объясняет, как просматривать библиотеку типов с помощью любой программы *Microsoft Office*. Ваш *VBScript* редактор, предоставляет доступ к библиотеке. Для получения информации см. Справку вашего редактора.

1. В любом приложении *Microsoft Office*, choose *Tools > Macro > Visual Basic Editor*.
2. В *Visual Basic Editor*, выберите *Tools > References*.



3. В появившемся диалоговом окне установите флажок для *Adobe Illustrator CC Type Library*, нажмите OK.
4. Выберите *View > Object Browser, to display the Object Browser window*.
5. Choose “*Illustrator*” from the list of open libraries in the top-left pull-down menu of the Object Browser window.

Несколько расширенных примеров сценариев находятся в папке */Scripting/Sample Scripts* каталог установки *Illustrator CC*.

Вы также можете просматривать образцы сценариев и информацию об отдельных классах, объектах, свойствах, методах, и параметры в *Adobe Illustrator CC Scripting Reference: VBScript*, которые можно скачать <http://www.adobe.com/devnet/illustrator/scripting/>.

## Выполнение сценариев

Интерфейс *Illustrator* включает меню «Сценариев» (*File > Scripts*), которое обеспечивает быстрый и легкий доступ к сценариям.

Сценарии могут быть перечислены непосредственно как пункты меню, запускаемые при их выборе. см. “Установка сценариев в меню *Scripts*” стр. 9.

Вы можете перейти из меню к любому сценарию в вашей файловой системе, а затем запустить его. см. “Выполнение сценариев из меню *Другие сценарии*” стр. 9.

У вас также могут быть сценарии *JavaScript* с расширением **.jsx**, которые запускаются автоматически при запуске приложения. Для получения информации. см. “Автозапуск сценариев (только **.jsx** сценарии)” стр. 9.

## Установка сценариев в меню *Scripts*

Чтобы включить сценарий в меню «Сценариев» (*File > Scripts*), сохраните сценарий в папке «Сценарии» */Illustrator CC/Presets* в каталоге установки *Illustrator CC*.

Имя файла сценария, минус расширение файла, отображается в меню «Сценарии».

Сценарии, которые вы добавляете в папку *Scripts* во время работы *Illustrator*, не отображаются в меню *Scripts*, пока не перезапустите *Illustrator*.

В меню «Сценарии» можно установить любое количество сценариев. Если у вас много сценариев, используйте вложенные папки для упорядочивания сценариев в меню сценариев. Каждая под папка отображается как отдельное подменю, содержащее сценарии в этой под папке.

## Выполнение сценариев из меню *Другие сценарии*

Пункт *Другие сценарии* в конце меню (*File > Scripts > Other Scripts*) предназначен для выбора сценариев из другой папки.

При выборе «Другие сценарии» отображается диалоговое окно «Обзор», которое вы используете для перехода к файлу сценария. Когда вы выбираете файл, сценарий выполняется.

В диалоговом окне просмотра отображаются только файлы одного из поддерживаемых типов. Подробнее см. “Языки сценариев поддерживаемые в *Adobe Illustrator CC 2017*” стр. 6.

## Автозапуск сценариев (только **.jsx** сценарии)

Сценарии *JavaScript* с расширением файла **.jsx** можно установить в одну из двух папок, поэтому сценарии запускаются автоматически при запуске *Illustrator* и каждый раз при запуске сценария. Папки:

- Папка сценариев запуска для конкретного приложения, которая содержит сценарии для *IllustratorCC*.
- Общая папка сценариев запуска, которая содержит сценарии, которые запускаются автоматически при запуске любого приложения *Creative Cloud*.

## Папка сценариев запуска для конкретных приложений

Вы должны поместить сценарии запуска для конкретного приложения в папку с именем *Startup Scripts*, которую вы создаете в каталоге установки *Illustrator*.

Например, если *Illustrator*CC установлен в расположение по умолчанию, вы должны создать *Startup Scripts* папку в следующем месте:

Windows	C:\Program Files\Adobe\Adobe IllustratorCC\Startup Scripts\
Mac OS	/Applications/Adobe Illustrator CC/Startup Scripts/

*JavaScript* сценарии с расширением **.jsx** помещённые в *Startup Scripts* папку, запускаются в двух случаях:

- Приложение запущено.
- Любой файл *JavaScript* выбирается в меню «Сценарии» (*File > Scripts*).

## Папка общих сценариев запуска

Папка общих сценариев запуска содержит сценарии, которые запускаются автоматически при запуске любого приложения *Creative Cloud*. Вы создаете папку в следующем месте:

Windows	Program Files/Common Files/Adobe/Startup Scripts CC/Illustrator
Mac OS	::Library:Application Support:Adobe:Startup Scripts CC:Illustrator

Если сценарий в общей папке автозагрузки предназначен для выполнения только в *Illustrator*, сценарий должен включать директиву *ExtendScript #target* (**#target illustrator**) или следующий код:

```
if( BridgeTalk.appName == "illustrator" ) {
    //continue executing script (продолжаем выполнение скрипта)
}
```

Подробнее см. *JavaScript Tools Guide*.

## Изменения CC

В этом разделе перечислены изменения, внесенные в объектную модель для поддержки функций в *Illustrator* CC. Подробно описания можно найти в справочных документах по сценариям для каждого языка сценариев.

### Перечисления и константы

Новое значение перечисления и значение по умолчанию (для совместимости с *AutoCad*):

AppleScript	JavaScript	VBScript
<b>auto cad compatibility.</b> new values <b>auto cad release 21</b> <b>auto cad release 24</b> (default)	<b>AutoCADCompatibility.</b> new values <b>AutoCADRelease21</b> <b>AutoCADRelease24</b> (default)	<b>AutoCADCompatibility.</b> new values <b>aiAutoCADRelease21</b> <b>aiAutoCADRelease24</b> (default)

Новое значение и значение по умолчанию для совместимости версий при сохранении в формате EPS или AI:

AppleScript	JavaScript	VBScript
<b>compatibility.</b> new values <b>illustrator 21</b>	<b>Compatibility.</b> new values <b>Illustrator21</b>	<b>Compatibility.</b> new values <b>aiIllustrator21</b>

Новая константа для стилей макета нескольких документов:

AppleScript	JavaScript	VBScript
<code>document layout style</code> values <div> <div><code>cascade</code></div> <div><code>horizontal tile</code></div> <div><code>vertical tile</code></div> <div><code>float all</code></div> <div><code>consolidate all</code></div> </div>	<code>DocumentLayoutStyle.</code> values <div> <div><code>CASCADE</code></div> <div><code>HORIZONTALTILE</code></div> <div><code>VERTICALTILE</code></div> <div><code>FLOATALL</code></div> <div><code>CONSOLIDATEALL</code></div> </div>	<code>DocumentLayoutStyle.</code> values <div> <div><code>aiCASCADE</code></div> <div><code>aiHORIZONTALTILE</code></div> <div><code>aiVERTICALTILE</code></div> <div><code>aiFLOATALL</code></div> <div><code>aiCONSOLIDATEALL</code></div> </div>

## Методы и свойства

Новые методы / команды приложения управляют рабочими пространствами:

AppleScript	JavaScript	VBScript
<code>save workspace</code>	<code>saveWorkspace ()</code>	<code>SaveWorkspace ()</code>
<code>switch workspace</code>	<code>switchWorkspace ()</code>	<code>SwitchWorkspace ()</code>
<code>delete workspace</code>	<code>deleteWorkspace ()</code>	<code>DeleteWorkspace ()</code>
<code>reset workspace</code>	<code>resetWorkspace ()</code>	<code>ResetWorkspace ()</code>

Новый метод / команда документа определяет стиль макета для нескольких документов:

AppleScript	<code>arrange [document layout style]</code>
JavaScript	<code>arrange (layoutStyle)</code>
VBScript	<code>Arrange (layoutStyle as DocumentLayoutStyle)</code>

Новые методы / команды текстового фрейма преобразуют объекты типа области и типа точки:

AppleScript	<code>convert area object to point object</code>
	<code>convert point object to area object</code>
JavaScript	<code>convertAreaObjectToPointObject ()</code>
	<code>convertPointObjectToAreaObject ()</code>
VBScript	<code>ConvertAreaObjectToPointObject ()</code>
	<code>ConvertPointObjectToAreaObject ()</code>

Новые возможности экспорта SVG:

AppleScript	JavaScript	VBScript
<code>save multiple art boards</code>	<code>saveMultipleArtBoards</code>	<code>SaveMultipleArtBoards</code>
<code>artboard range</code>	<code>artboardRange</code>	<code>ArtboardRange</code>
<code>include unused styles</code>	<code>includeUnusedStyles</code>	<code>IncludeUnusedStyles</code>

## Ответы на вопросы

Сценарии, которые создают, сохраняют и закрывают файлы *Illustrator*, должны периодически завершать работу и перезапускаться. Рекомендуемое максимальное количество файлов для обработки перед выходом и перезапуском. *Illustrator* это:

- *Windows 500 files*
- *Mac OS 1000 files*

Для получения дополнительной информации о выходе и перезапуске *Illustrator* см. “Запуск и активация *Illustrator*” стр. 20 и “Выход из *Illustrator*” стр. 21.

*An Illustrator error occurred: 1346458189 (“PARM”)* может появиться при неправильном написании. Скрипты многократно запускаются в *Illustrator* из *ESTK*. Сценарии должны быть осторожны с инициализацией переменных и конфликтом пространства имен при повторном выполнении в *Illustrator* через *ESTK* за один сеанс. Каждый запуск сценария выполняется в рамках одного и того же постоянного движка *ExtendScript* в *Illustrator*.

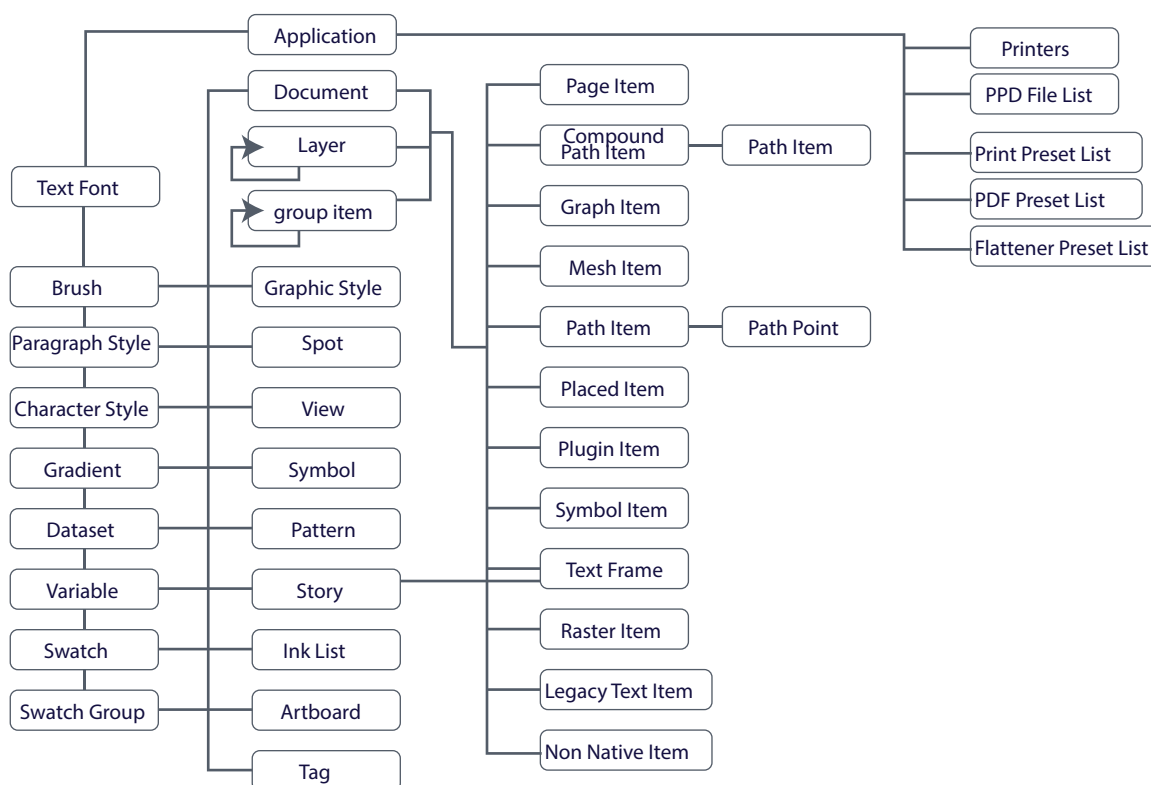
Отладчик *ESTK* использует *BridgeTalk* для связи с *Illustrator*. Единый глобальный, стойкий, движок *ExtendScript* внутри *Illustrator* обрабатывает все коммуникации *BridgeTalk*. Эффект заключается в том, что состояние механизма *ExtendScript* является совокупным для всех сценариев, которые выполнялись ранее. Проблемы с кодом сценариев, которые могут вызвать эту проблему:

- Чтение не инициализированных переменных.
- Конфликты глобальных пространств имен, например, когда два глобальных объекта из разных сценариев сбивают друг друга.

Если вы создаете более одного арт-объекта в *AppleScript* и назначаете каждый из них переменной, все переменные устанавливаются на последний элемент. Это означает, что ранее созданные элементы недоступны.

## 2 Объектная модель сценариев *Illustrator*

Хорошее понимание объектной модели *Illustrator* улучшит ваши навыки написания сценариев. На рисунке показана иерархия вложенности объектной модели, начиная с объекта **application**. Примечание, что классы элементов **layer** и **group item** могут содержать вложенные объекты одного и того же класса, которые, в свою очередь, могут содержать дополнительные вложенные объекты.



В дополнение к этой объектной модели, зависящей от приложений, *JavaScript* предоставляет определенные служебные объекты, такие как объекты `file` и `folder`, которые обеспечивают независимый от операционной системы доступ к файловой системе. Подробности см. *JavaScript Tools Guide*.

## Соглашение об именовании объектов

Для интерфейса сценариев *Illustrator* существует одна объектная модель, но фактические имена объектов немного различаются на разных языках сценариев:

- Имена *AppleScript* пишутся строчными буквами, а отдельные слова разделяются пробелом; Например: `graphic style`
- Имена *VBScript* начинаются с заглавной буквы, и дополнительные слова в имени обозначаются инициалом в верхнем регистре письма; Например: `GraphicStyle`
- Имена *JavaScript* начинаются со строчных букв, а дополнительные слова в имени обозначаются прописными начальными буквами; Например: `graphicStyle`

В этой главе используются общие имена объектов и свойств, но вы можете легко применить эти соглашения и определить какому языку соответствуют данные имена. В этом документе названия свойств, методов и объектов набраны шрифтом *monospaced*.

## Объекты верхнего уровня

Используйте эти объекты для доступа к глобальной информации о приложении *Illustrator* или отдельного документа.

### Приложение (*Application*)

Свойства объекта `application` предоставляют вашему сценарию доступ к глобальным значениям, например:

- ▶ `preferences`, которые пользователь устанавливает в интерактивном режиме в приложении *Illustrator* с помощью диалогового окна настроек (*Edit > Preferences*).
- ▶ Системная информация, свойства установленных шрифтов (`text fonts`) и принтеров (`printer list`).

Кроме того, есть свойства, которые предоставляют информацию о конкретном приложении и информацию более высокого уровня о любых открытых документах:

- ▶ Информация о приложении установки `path`, `version`, и является ли *illustrator* `visible`.
- ▶ `current active` документ; то есть арт - полотно, которое отображается и принимает вводимые пользователем данные.
- ▶ Все открытые `documents`.

Метод или команда объекта `application` позволяет сценарию выполнять действия в масштабах всего приложения; Например:

- ▶ `Open files`
- ▶ `Undo` и `redo transactions`
- ▶ `Quit Illustrator`

### Документ (*Document*)

В объекте `document`, который ваши сценарии могут создавать или получать доступ через объект `application`, представляется художественный холст или загруженный файл *Illustrator*. Свойства объекта `document` дает вам доступ к содержимому документа; Например:

- ▶ `selection`. Текущее выделение или арт-объекты, выбранные пользователем в документе.
- ▶ `page items`. Все содержащиеся арт-объекты, называемые элементами страниц, составляющие дерево произведений.
- ▶ `symbols` и `text frames`. Арт - объекты определенных типов, такие как символы и текстовые фреймы.
- ▶ `layers`, `active layer`. Все слои или текущий активный слой.

Свойства документа также говорят вам о состоянии самого документа; Например:

- ▶ `ruler units`. Пользовательские настройки документа, например единицы измерения.
- ▶ `saved`. Был ли документ сохранен с момента последнего изменения содержания.
- ▶ `path`. Путь к соответствующему файлу.

Методы объекта `document` позволяют сценариям воздействовать на документ; Например:

- ▶ `save`, `save as` сохранить в файл *illustrator* или сохранить в различных поддерживаемых форматах файлов.
- ▶ `activate` или `close a document` (открыть или закрыть документ).
- ▶ `print`, `print options`, `printer list`. Распечатать документ. Ваши сценарии могут выбрать принтер, ссылаясь на объект параметров печати, или они могут ссылаться на доступные принтеры через свойство списка принтеров объекта приложения.

## Слой (Layer)

Объект `layer` обеспечивает доступ к содержимому или дереву иллюстраций определенного слоя. Вы получаете доступ к объекту `layer` через объект `document`. Свойства объекта `layer` обеспечивает доступ к слою или информацию о нём, например:

- ▶ `visible` или `locked`. Виден или заблокирован слой.
- ▶ `opacity` (общая прозрачность) и `z order position` (положение в порядке укладки).
- ▶ `artwork knockout` и `blending mode`. Настройки художественного оформления для слоя, такие как выделение иллюстраций и режим наложения.

## Дерево арт - объектов (The artwork tree)

Содержимое документа Illustrator называется artwork tree. Дерево арт-объектов представлено следующими объектами:

- |                                   |  |
|-----------------------------------|--|
| ● <code>compound path item</code> | ● <code>path item</code>                           |
| ● <code>graph item</code>         | ● <code>placed item</code>                         |
| ● <code>group item</code>         | ● <code>plugin item</code>                         |
| ● <code>legacy text item</code>   | ● <code>raster item</code>                         |
| ● <code>mesh item</code>          | ● <code>symbol item</code> (см. "Dynamic objects") |
| ● <code>non native item</code>    | ● <code>text frame</code>                          |

Ваши сценарии могут получать доступ к арт - объектам и управлять ими через коллекции в объекте `document` и `layer`. Коллекции арт-объектов бывают двух типов:

- ▶ `graph items` или `mesh items`. Объекты коллекции, которые соответствуют каждому отдельному типу арт - объекта, например, элементы графика или элементов сетки.
- ▶ `page items` включающий арт-объекты всех типов.



`group item`. Кроме того, вы можете использовать объект группового элемента для ссылки на сгруппированный набор арт — объектов.

Вы можете создавать новые арт-объекты командой `make` (AppleScript) или методом `add` графического объекта коллекции. Например чтоб создать новый объект `path item`:

AppleScript	<code>set myPathItem to make new path item in current document</code>
JavaScript	<code>var myPathItem = activeDocument.pathItems.add();</code>
VBScript	<code>Set myPathItem = appRef.ActiveDocument.PathItems.Add()</code>

Следующие коллекции арт - объектов не позволяют создавать новые объекты с помощью команды `make` или метода `add`:

- `graph items`
- `mesh items`
- `plugin items`
- `legacy text items`

Подробнее о создании объектов этих типов см. *Adobe Illustrator CC Scripting Reference* для вашего языка программирования.

## Графические стили (Art styles)

Ваш скрипт может применить графический стиль к арт - объекту используя объект `graphic style`. Чтобы применить графический стиль, используйте свойство `graphic styles` у объекта `document`. Для доступа к объекту `graphic style` используйте метод `apply to`.

Точно так же объект `brush` позволяет указать кисть для применения к иллюстрации. Вы получаете доступ к любой кисти через объект коллекции `brushes`, который является свойством объекта `document`.

## Цвет (Color objects)

Ваш скрипт может применить цвет, узор или градиент к объекту `path item`, используя свойства `fill color` или `stroke color`:

- Сценарии могут устанавливать новые образцы цветов с помощью команды `make` или метода `add` к объекту `swatches`. Ваш скрипт также может создать новый плашечный цвет, используя команду `make` или `add` свойство объекта `spots`.
- Вы можете определить атрибуты объекта `ink`, используя объект `ink info`, который является свойством объекта `ink`. Вы получаете доступ к объекту `ink` через `ink list` свойство объекта `document`.

Следующие объекты позволяют создавать цвета в определенных цветовых пространствах:

- `RGB color` используя диапазон от 0,0 до 255,0 для каждого из трех отдельных значений цвета.
- `CMYK color` используя процентное соотношение значений от 0.0 до 100,0 для каждого из четырех отдельных значения цвета.
- `grayscale color` или `LAB color` используя один и тот же диапазон и количество значений, которые вы используете в приложении *Illustrator*.

## Текст (Text objects)

Когда вы печатаете текст в документ *Illustrator*, тип автоматически становится объектом `text frame` и, в тоже время, объектом `story`.

Чтобы убедиться в этом, откройте новый документ в *Illustrator* и используйте инструмент горизонтального текста, чтобы ввести текст, затем используйте инструмент вертикального текста, чтобы ввести ещё текст.

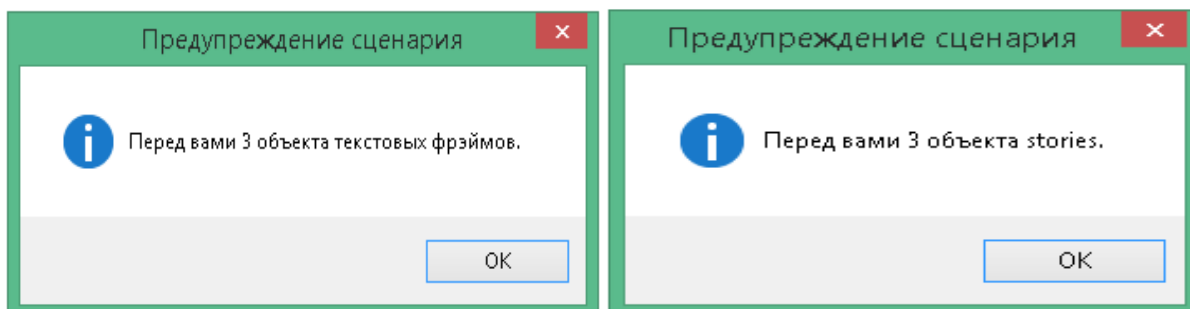


Наконец, создайте прямоугольник и введите текст внутри него.

Теперь запустите следующий сценарий JavaScript:

```
var myDoc = app.activeDocument
alert("Перед вами " + myDoc.textFrames.length + " объекта текстовых фреймов.")
alert("Перед вами " + myDoc.stories.length + " объекта stories.")
```

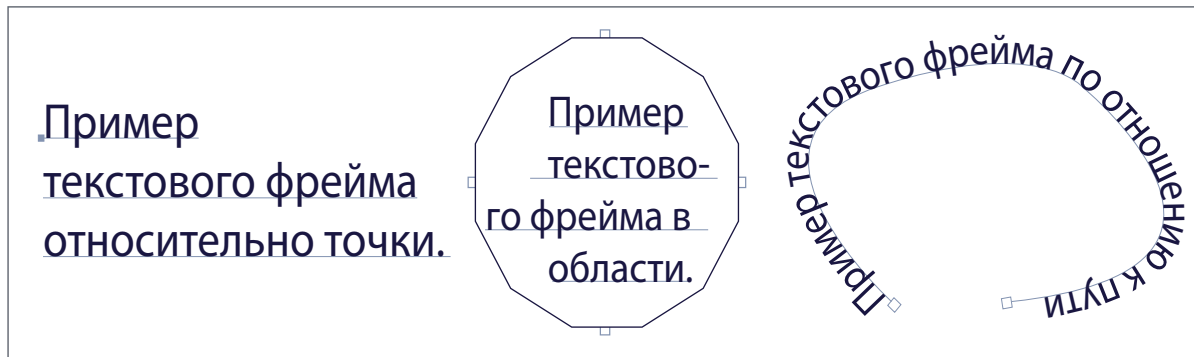
Работа скрипта заключается в подсчёте количества текстовых фреймов и объектов stories в открытом файле и вывода результата посредством диалогового окна.



## Текстовый фрейм (*Text frames*)

Есть три типа текстовых фреймов:

- point (точка)
- area (область)
- path (путь)



Чтобы создать текстовый фрейм определенного типа, используйте свойство `kind` объекта `text frames` в *AppleScript*. Объекты `text frames` в *JavaScript* и *VBScript* содержат специальные методы для создания текстовых фреймов области и текстовых фреймов путей.

Как и в приложении *Illustrator*, вы можете создавать текстовые фреймы в области или пути.

Чтобы связать существующие текстовые фреймы, используйте свойство `next frame` или `previous frame` у объекта `text frame`. Связанные фреймы составляют единый объект `story`.

Для получения информации о создании или потоковой связи текстовых фреймов см. главу «Связь фреймов в поток» стр.31 для языка JS или оригинал для других языков.

## Геометрия текста

Хотя три типа текстовых фреймов имеют общие характеристики, такие как `orientation`, у каждого из них есть типовые качества, отраженные в свойствах объекта `text frame`. Например:

- В текстовом фрейме области могут быть строки и столбцы, доступ к которым осуществляется через свойство `row count` и `column count`.
- Текст путей имеет свойства, `start T value` и `end T value` которые указывают, где на пути начинается и заканчивается текст.
- Текстовые фреймы области и пути связаны с объектом `text path`, и его свойством `text frame`. Текстовый путь определяет положение текстового фрейма и ориентацию (горизонтальная или вертикальная) на монтажной панели (в то время как свойство `orientation` объекта `text frame` определяет ориентацию текста в текстовом фрейме). Свойство `text path` недопустимо для точечного текста, потому что положение и ориентация точечного текста заданы и полностью определяется свойствами самого текстового фрейма.

#### Примечание:

Текстовый путь (**text path**) - это не то же самое, что путь арт - элемента (**path art item**). Но они связаны на панели Оформления (*Appearance*) с Графическими стилями (*Graphic Styles*), где можно изменить их внешний вид.

## Объекты представляющие текстовое содержимое

К фактическому содержимому текстового фрейма или story можно получить доступ как и к любому из следующих объектов:

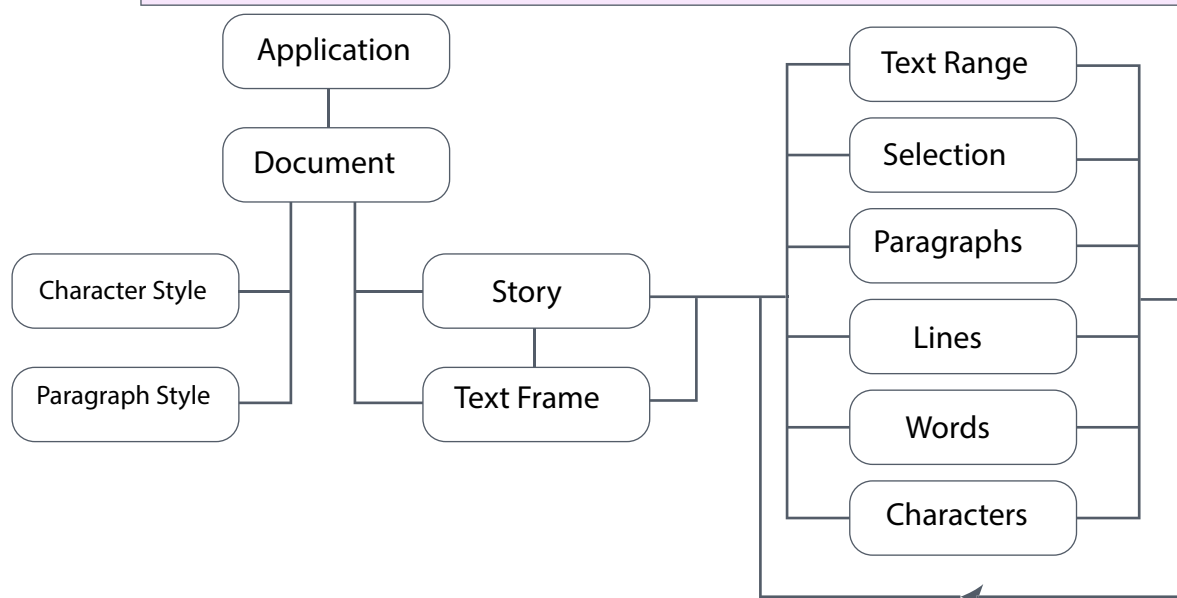
- `characters`
- `words`
- `paragraphs`
- `lines`

Объект `line` это все символы, которые помещаются на одной строке в объекте `text frame` или объекте `story`. Все текстовые элементы имеют хотя бы одну строку текста, определенную как объект `line`.

Текстовый - арт может состоять из нескольких строк текста, если текст содержит жесткие разрывы строк или его символы переходят в новую строку, потому что они не помещаются в ширину фрейма. Доступ к текстовым объектам и их идентификация осуществляется коллекциями внутри объектов `text frame` и `story`;

Пример:

```
textFrame("My Text Frame").paragraphs
story("My Story").paragraphs
```



Объекты `text frame` и `story` имеют свойства `insertion point` и `text selection`. Свойства объекта `text frame` также включают в себя определяющие функции текстового фрейма, такие как:

- *The frame* `width`, `height`, and `position` - ( ширина, высота, положение )
- *Whether the frame is* `hidden` or `locked` - ( скрыт или заблокирован фрейм )
- *Whether the text is* `editable` - ( есть ли возможность редактирования текста )

#### Примечание :

Объект `line` не может быть создан в сценарии. Ваш скрипт может создавать `character`, `paragraph`, и `word` объекты.

## Текстовый диапазон (Text ranges)

Различные текстовые объекты в текстовом фрейме или story также представлены вместе в объекте `text range`. Например, символ - это текстовый диапазон длиной 1, а слово - это текстовый диапазон, имеющий пространство перед ним (пробел).

Вы можете установить содержимое объекта `text range` передав строку с помощью свойства `contents`.

## Стили текста (Text styles)

Элементы стиля текста, такие как `font`, `capitalization`, и `justification`, представлены объектами `paragraph attribute` и `character attribute`.

Эти объекты, атрибутами которых являются свойства объектов `paragraph style` и `character style`.

Объекты `paragraph style` и `character style` имеют методы `apply to` и `remove`, которые позволяют вашему сценарию назначать или удалять атрибуты в определенном абзаце, символе или текстовом диапазоне.

Вы можете изменить свойства отображения текстового диапазона, применив соответствующий стиль или предоставив локальное переопределение атрибутов на уровне текста или абзаца:

- `character style` применяются к наборам из одного или нескольких символов. Они контролируют такие особенности как `font`, `alignment`, `leading`, `language` и `capitalization`, которые являются свойствами объекта `character attribute`.
- `paragraph style` применяется к абзацам. Они управляют функциями абзаца, такими как `first line indent`, `left indent`, и `right indent`, которые являются свойствами объекта `paragraph attribute`.

## Динамические объекты (Dynamic Objects)

Создавая динамические объекты, вы можете создавать графику, управляемую данными. В приложении *Illustrator* вы используете Панель переменных для создания или редактирования переменных, таких как данные графика, связанные файлы, текстовая строка и видимость или переменные, чей тип не указан.

В сценариях вы используете объект `variable` для представления переменной этого типа.

Свойство `kind` указывает тип динамических данных, которые хранятся в объекте `variable`.

`variable` - это объекты уровня документа; вы создаете их в объекте `document`.

**Примечание:**

Не путайте объекты `variable` с переменными сценария. Подробнее о переменных *Illustrator* см. динамические объекты и графику, управляемую данными, в справке *Illustrator*.

Наборы данных, которые собирают переменные и связанные с ними динамические данные в один объект, представлены в сценариях с помощью объекта `dataset`. Объект `dataset` предоставляет методы для обновления и удаления объекта `dataset` в ваших скриптах.

## Символы (Symbols)

В *Illustrator* символы - это арт элементы, которые хранятся на панели «Символы». Ваши скрипты могут создавать, удалять и дублировать объекты `symbol`.

Когда вы создаете объекты `symbol` в своем скрипте, *Illustrator* добавляет их в Панель символов для целевого документа.

`symbol item` является экземпляром объекта `symbol` в документе. Каждый `symbol item` связан со своим символом определением, поэтому изменение определения символа обновляет все экземпляры символа.

Ваш сценарий может создавать, удалять и дублировать элементы символов. Элементы-символы - это арт элементы *Illustrator*; следовательно, с ними можно обращаться так же, как и с другими арт элементами или элементами страницы.

Вы можете вращать, изменять размер, выбирать, блокировать, скрывать и выполнять другие операции с элементами символа.

## Трансформация (Transformations)

Объект `matrix` обеспечивает доступ к мощи матриц геометрического преобразования. Матрицы трансформации в *Illustrator* хранят настройки операции, которая масштабирует, вращает или перемещает (переводит) объект на странице. Преимущества использования матриц:

- Сохраняя значение преобразования в объекте `matrix`, вы можете многократно использовать значения для разных объектов в вашем скрипте.
- Объединяя матрицы вращения, сдвига и / или масштабирования и применяя полученную матрицу, вы можете выполнять множество геометрических преобразований с помощью только одного оператора скрипта.
- Вы можете инвертировать значения матрицы.
- Вы можете сравнить значения двух матриц.

Объект `application` имеет команды или методы для создания, получения, инвертирования, сравнения или объединения матриц.

Команда или метод, используемый для применения матрицы, - это команда `transform`, которая принадлежит к любому типу объекта, для которого могут быть выполнены преобразования.

## 3 Scripting Illustrator

Эта глава представляет собой обзор того, как использовать объекты сценария для программирования *Illustrator* CC 2017. Конкретные примеры для поддерживаемых языков сценариев находятся в следующих главах.

## Запуск и выход из Illustrator посредством скрипта

Ваши сценарии могут контролировать активацию и завершение работы Illustrator.

### Запуск и активация Illustrator

#### AppleScript

В *AppleScript* для нацеливания на *Illustrator* используется оператор **tell**. Команда *activate* активирует *Illustrator*, если он еще не активен.

```
tell application "Adobe Illustrator"
  activate
end tell
```

#### JavaScript

Как правило, вы запускаете сценарии *JavaScript* из меню сценариев приложения (*File > Scripts*) или из папки запуска, поэтому нет необходимости запускать *Illustrator* из вашего скрипта.

Информация о запуске *Illustrator* на *JavaScript* выходит за рамки этого руководства.

Для получения подробной информации см. "*interapplication messaging*" или "*JavaScript messaging framework*" в *JavaScript Tools Guide*.

#### VBScript

В *VBScript* есть несколько способов создать экземпляр *Illustrator*:

- **CreateObject** запускает *Illustrator* как невидимое приложение, если оно еще не запущено. Если *Illustrator* запущено как невидимое приложение, вы должны вручную активировать приложение, чтобы оно стало видимым:

```
Set appRef = CreateObject("Illustrator.Application")
```

Если у вас установлено несколько версий *Illustrator* на одном компьютере и вы используете метод **CreateObject** для получения ссылки на приложение, с помощью "**Illustrator.Application**" создает ссылку до последней версии *Illustrator*. Чтобы специально настроить таргетинг на более раннюю версию, используйте идентификатор версии в конец строки:

Для Illustrator 10, используйте	"Illustrator.Application.1"
Для Illustrator CS, используйте	"Illustrator.Application.2"
Для Illustrator CS2, используйте	"Illustrator.Application.3"
Для Illustrator CS3, используйте	"Illustrator.Application.4"
Для Illustrator CS4, используйте	"Illustrator.Application.CS4"
Для Illustrator CS5, используйте	"Illustrator.Application.CS5"
Для Illustrator CS6, используйте	"Illustrator.Application.CS6"
Для Illustrator CC, используйте	"Illustrator.Application.CC"
Для Illustrator CC 2014, используйте	"Illustrator.Application.CC2014"
Для Illustrator CC 2015, используйте	"Illustrator.Application.CC2015"
Для Illustrator CC 2017, используйте	"Illustrator.Application.CC2017"

Используйте оператор **New** если вы добавили в проект ссылку на библиотеку типов *Illustrator*. Например, следующая строка создает новую ссылку на объект **Application**:

```
Set appRef = New Illustrator.Application
```

## Выход из *Illustrator*

### AppleScript

Используйте команду **quit**:

```
tell application "Adobe Illustrator"
quit
end tell
```

### JavaScript

Используйте метод **app.quit()**:

```
app.quit()
```

### VBScript

Используйте объект **Application** с методом **Quit**:

```
Set appRef = CreateObject("Illustrator.Application")
appRef.Quit
```

## Работа с объектами

### Получение самого переднего документа или слоя

Для того, чтобы обратиться к выбранному документу, используйте свойство **current document** у объекта **application** в *AppleScript* или свойство **active document** в *JavaScript* или *VBScript*.

Также вы можете использовать объект **document** со свойством **current layer** или **active layer** для ссылки на выбранный слой.

Существуют и другие типы «активных» или «текущих» свойств объекта, например, **active dataset** или **active view**.

Для подробных деталей см. *Adobe Illustrator CC 2017 Scripting Reference* для вашего языка программирования.

### Создание новых объектов

Некоторые объекты (кроме объекта **application**) не могут быть получены из контейнеров или родительских объектов. Ваш сценарий должен создавать эти объекты напрямую.

Следующие объекты должны быть созданы явно:

- **CMYK color**
- **document preset**
- **EPS save options**
- **export options AutoCAD**
- **export options Flash**
- **export options GIF**
- **export options JPEG**
- **export options Photoshop**
- **export options PNG8**
- **export options PNG24**
- **export options SVG**
- **export options TIFF**
- **file**
- **folder**
- **gradient color**
- **gray color**
- **Illustrator save options**
- **ink**
- **ink info**
- **lab color**

- `matrix`
- `MXG save options`
- `no color`
- `open options`
- `open options AutoCAD`
- `open options FreeHand`
- `open options PDF`
- `open options Photoshop`
- `paper info`
- `Pattern color`
- `PDF save options`
- `PPD file`
- `PPD file info`
- `print color management options`
- `print color separation options`
- `print coordinate options`
- `printer`
- `printer info`
- `print flattener options`
- `print font options`
- `print job options`
- `print options`
- `print page marks options`
- `print paper options`
- `print postscript options`
- `raster effect options`
- `rasterize options`
- `screen`
- `screen spot function`
- `RGB color`
- `spot color`
- `tracing options`

Объекты `file` и `folder` это устройства *Adobe ExtendScript*, предназначено для обеспечения независимого от платформы доступа к базовой файловой системе. Для получения информации об использовании этих объектов см. *JavaScript Tools Guide*. Информацию о явном создании объекта см. В главе, посвященной вашему языку сценариев.

## Объекты коллекции (*Collection objects*)

Большинство объектов коллекции должно быть получено из контейнера. Например, объект коллекции `path items` может содержаться в объекте `document` или объекте `layer`; чтобы получить в коллекции `path items` обратитесь к любому из этих объектов. Например, см. Разделы, относящиеся к конкретному языку, ниже.

### AppleScript

Обратиться к объекту `path items` в документе, второй вариант в слое:

```
path item 1 in document 1
```

```
path item 1 in layer 1 in document 1
```

### JavaScript

Обратиться к объекту `path items` в документе, второй вариант в слое:

```
documents[0].pathItems[1]
```

```
documents[0].layers[0].pathItems[0]
```

### VBScript

Обратиться к объекту `path items` в документе, второй вариант в слое:

```
Documents(1).PathItems(1)
```

```
Documents(1).Layers(1).PathItems(1)
```



## Выбранные объекты

Иногда вам нужно написать сценарии, которые воздействуют на текущий выбранный объект или объекты. Например, вам может потребоваться применить форматирование к выделенному тексту или изменить фигуру выделенного контура.

### Выделить текст

Чтобы выделить текст, используйте команду `select` или метод объекта `text range`.

### Выделить арт - объект

Вы можете выбрать арт-объект (например, графические элементы, сеточные элементы, растровые элементы и элементы символов), установив его свойство `selected` = `true`. (В *AppleScript*, `selected` это свойство объекта `page items`.)

### Ссылка на выделенные арт - объекты

Для обозначения всех выбранных объектов в документе, используйте объект `document` свойство `selection`. Для работы с объектами в массиве, вы должны определить их тип, чтобы вы знали, какие свойства и методы или команды, вы можете использовать для них.

В *JavaScript* и *VBScript* каждый тип арт - объекта имеет свойство `typename` доступное только для чтения, которое можно использовать для определения типа объекта. В *AppleScript* используйте свойство `class`.

### Примечания по переименованию объектов, хранящихся в панелях приложения

Можно переименовать несколько объектов; то есть их свойство доступное для записи `name`. Следующие типы объектов могут быть отсортированными в алфавитном порядке на соответствующей панели *Illustrator*. Если скрипт изменяет имя такого объекта, ссылки на этот объект по индексу могут стать недействительными.

- `Brush`
- `Gradient`
- `Graphic Style`
- `Pattern`
- `Swatch`
- `Symbol`
- `Variable`

## Единицы измерения

*Illustrator* использует точки в качестве единицы измерения почти для всех расстояний. Один дюйм равен 72 точкам. Исключением являются значения таких свойств, как  `Kerning`, `tracking`, `aki` (используются для композиции текста на японском языке), в которых используются единицы em (см. "Em относительные ед." стр. 24.)

*Illustrator* использует точки при взаимодействии с вашими скриптами независимо от текущих единиц измерения линейки. Если ваш сценарий зависит от сложения, вычитания, умножения или деления конкретных значений измерения для единиц кроме точек, он должен выполнять любые преобразования единиц, необходимые для представления ваших измерений в виде точек.

Например, чтобы использовать дюймы для координат или единиц измерения, необходимо умножить все значения в дюймах на 72 при вводе значений в ваш скрипт.

В следующей таблице приведены формулы преобразования для различных единиц измерения:

Единицы измерения	Формула преобразования
centimeters	28.346 points = 1 centimeter
inches	72 points = 1 inch
millimeters	2.834645 points = 1 millimeter
picas	12 points = 1 pica
Qs	0.709 point = 1 Q (1 Q equals 0.23

JavaScript предоставляет объектный тип *UnitValue*, который предлагает утилиты для преобразования единиц измерения. Подробнее см. *JavaScript Tools Guide*.

## Em относительные единицы (*Em space units*)

Значения, в которых вместо точек используются единицы em, измеряются в тысячных долях em. Em пропорционален текущему размеру шрифта. Например, в шрифте из 6 пунктов 1 em равен 6 пунктам; в шрифте 10 пунктов, 1 em равняется 10 пунктам. Для шрифта размером 10 пунктов значение кернинга в 20 единиц em эквивалентно:

$$(20 \text{ units} \times 10 \text{ points}) / 1000 \text{ units/em} = 0.2 \text{ points}$$

## Page-item расположение и размеры

Иллюстратор использует простую, двухмерную геометрию в виде точек, чтобы записать **position** объекта **page item** в документе. Каждый объект **page item** в документе имеет свойство **position** которое определяет фиксированную точку как пару координат страницы в формате **[x, y]**. Фиксированная точка - это верхний левый угол ограничивающей рамки объекта.

Для получения информации о типах объектов, составляющих коллекцию **page items**, см. "Дерево арт - объектов" стр. 14.

Точка обозначается парой координат:

- Горизонтальное положение, **x**
- Вертикальное положение, **y**

Эти координаты можно увидеть на панели «Информация», когда вы выбираете или создаете объект в *Illustrator*.

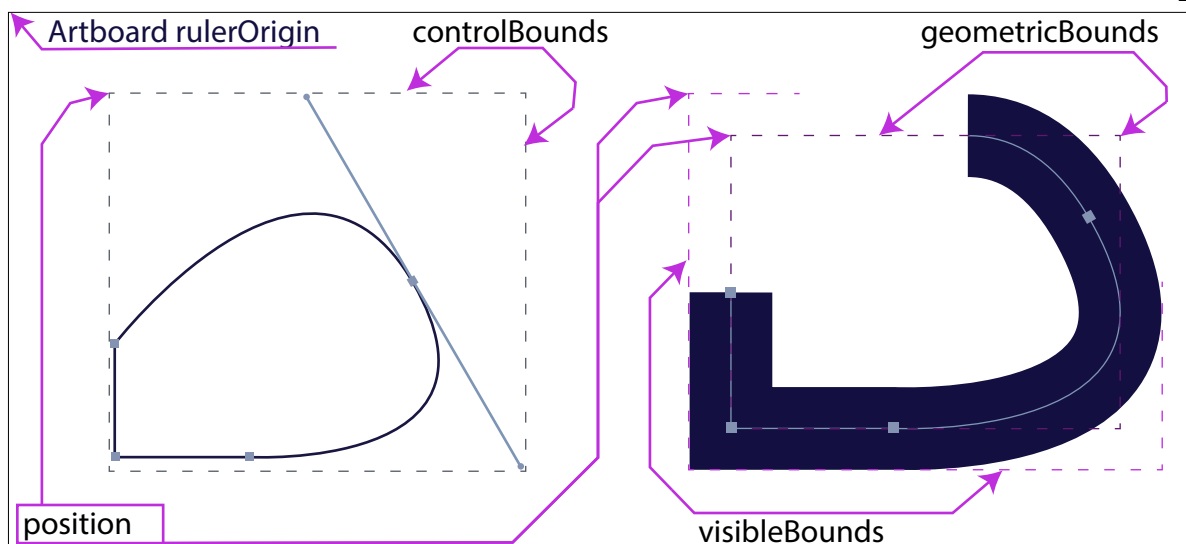
Для монтажной области исходной точкой координат по умолчанию является верхний левый угол (0,0), отраженный в исходной точке **ruler origin** свойства объекта **artboard**. Значения координаты **X** увеличиваются слева направо, а значения **Y** увеличиваются сверху вниз. Это изменилось в версии CS5; для обеспечения совместимости со сценарием, документ созданный сценарием, по-прежнему использует старую систему с исходной точкой в левом нижнем углу монтажной области и значением **Y**, увеличивается снизу вверх. Свойства **page origin** объекта **document** определяет нижний левый угол области печати документа как фиксированная точка. Каждый объект **page item** имеет свойства **width** и **height**. Максимальное допустимое значение ширины или высота элемента страницы - 16348 пунктов.

## Art item границы

Каждый объект **page item** имеет три свойства, которые имеют фиксированные прямоугольные области для описания общего объёма объекта:

- ▶ **geometric bounds** элемент равный геометрической форме объекта, без учета ширины обводки.
- ▶ **visible bounds** элемент равный геометрической форме объекта, включая ширину обводки.
- ▶ **control bounds** определяет прямоугольные размеры объекта, в том числе точки входа и выхода манипуляторов.

На следующем рисунке эти свойства показаны с использованием соглашений об именах *JavaScript*.



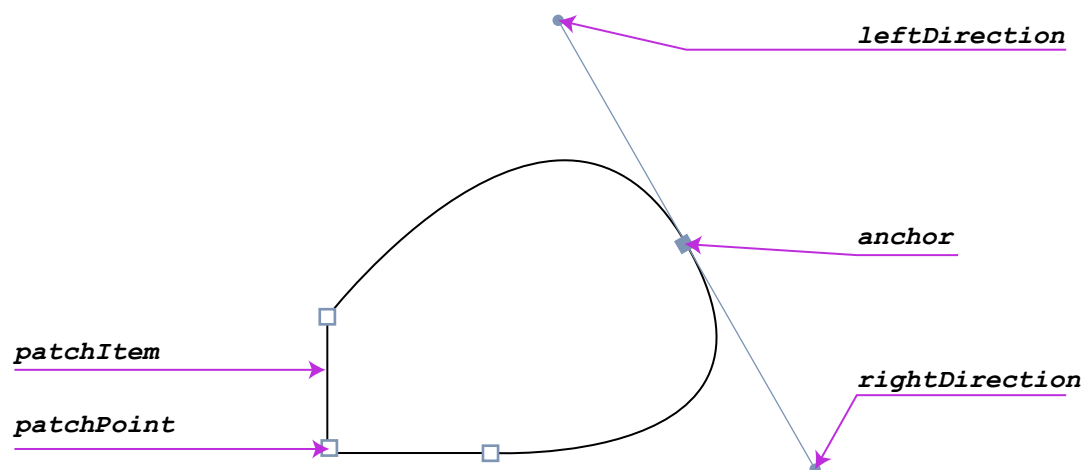
`position` и границы элемента рисунка, в интерфейсе Illustrator панель Трансформирование, вы управляете только в границах `geometricBounds`.

## Путь и фигуры (Paths and shapes)

Пути представлены в модели *Illustrator DOM* объектом `path item`. Элементы пути включают все иллюстрации, содержащие пути, такие как прямоугольники, эллипсы и многоугольники, а также пути произвольной формы.

Путь произвольной формы состоит из ряда точек пути. Точку пути можно указать двумя способами:

- В виде массива координат страницы `[x,y]`.
- В качестве объекта `path point`, который определяет точку привязки и две точки направления или ручки, определяющие кривую сегмента пути:



Подробные сведения, примеры и информацию о создании фигур см. в главе, посвящённой вашему языку сценариев.

## Уровни взаимодействия с пользователем

Когда требуется обратная связь с пользователем, приложение обычно представляет диалоговое окно. Это называется взаимодействием с пользователем. Это полезно и ожидаемо, когда вы напрямую взаимодействуете с приложением; однако, когда сценарий взаимодействует с приложением, диалог останавливает выполнение скрипта до тех пор, пока диалог не будет закрыт. Это может быть серьёзной проблемой в среде автоматизации, где нет никого, кто мог бы иметь дело с диалогами.

Объект `application` содержит свойство `user interaction level`, которое позволяет управлять уровнем взаимодействия, разрешенного во время выполнения скрипта. Вы можете подавить взаимодействие в среде автоматизации или разрешить какое-либо взаимодействие, когда сценарии используются в более интерактивной манере.

### AppleScript

Используя *AppleScript*, можно отправлять команды с одной машины на другую, поэтому возможны дополнительные типы взаимодействия. В *AppleScript*: существует четыре возможных значения для свойства `user interaction level`:

Свойство	Результат
<code>never interact</code>	Никакое взаимодействие не допускается.
<code>interact with self</code>	Взаимодействие только со сценариями, ( <i>File &gt; Scripts</i> ).
<code>interact with local</code>	Взаимодействие со сценариями, на локальном компьютере (вкл. Себя).
<code>interact with all</code>	Взаимодействие со всеми сценариями.

Четыре значения позволяют управлять взаимодействием в зависимости от источника команд сценария. Например, если приложение действует как сервер для удаленных пользователей, удаленному пользователю будет сложно отклонить диалог, но это не будет проблемой для человека, сидящего перед машиной. В этом случае уровень взаимодействия, < взаимодействие с локальным > не позволит диалоговым окнам останавливать удаленные сценарии, но позволит диалоговым окнам быть представленными для локальных сценариев.

### JavaScript

В *JavaScript* есть два возможных значения для свойства `app.userInteractionLevel`:

Свойство	Результат
<code>DISPLAYALERTS</code>	Взаимодействие разрешено.
<code>DONTDISPLAYALERTS</code>	Никакое взаимодействие не допускается.

### VBScript

В *VBScript*, есть два возможных значения для свойства `UserInteractionLevel` у объекта `Application`:

Свойство	Результат
<code>aiDisplayAlerts</code>	Взаимодействие разрешено.
<code>aiDontDisplayAlerts</code>	Никакое взаимодействие не допускается.

## Печать документов

Используя функцию сценария `print options`, вы можете фиксировать и автоматизировать части рабочего процесса печати. Сценарии раскрывают все возможности печати *Illustrator*, некоторые из них могут быть недоступны через пользовательский интерфейс приложения.

*Illustrator* поддерживает не более одного сеанса печати за раз из-за ограничений в текущей архитектуре печати.

Команда `document` или метод `print` принимает один необязательный параметр, который позволяет указать объект `print options`.

Объект `print options` позволяет вам определять параметры печати, такие как *PPD*, параметры *PostScript*, параметры бумаги и управление цветом и т. д. Объект `print options` имеет свойство `print preset`, которое позволяет вам указать набор настроек для определения вашего задания на печать.

При определении свойств объекта `print options`, вы можете узнать, какие принтеры, PPD-файлы наборы настроек печати и другие элементы доступны, с помощью свойств «списка» объекта `application` таких как `printer list`, `PPD file list`, `print presets list` доступных только для чтения.

## 4 Сценарии на языке JavaScript

В этой главе используются примеры сценариев и пояснения, которые помогут вам познакомиться с *Illustrator*, создание сценариев с использованием *JavaScript*.

### Для дополнительной информации

Несколько расширенных примеров сценариев находятся в папке `/Scripting/Sample Scripts` каталог установки *Illustrator CC 2017*.

Для получения информации об отдельных классах, объектах, свойствах, методах и параметрах, а также образцы, демонстрирующие, как использовать многие из этих элементов, см. в *Adobe Illustrator CC 2017 Scripting Reference: JavaScript*, `/Scripting/Documentation` каталоге установки вашего *Illustrator CC 2017*.

Вы также можете использовать словарь *Illustrator*, доступ к которому можно получить из средства просмотра объектной модели в *ESTK*. Для получения информации об использовании *ExtendScript Toolkit* и средства просмотра объектной модели см. "JavaScript объектная модель" стр. 8 или *JavaScript Tools Guide*.

Если вы не понимаете концепции и термины, используемые в этой главе, прочтите *Adobe Introduction to Scripting*.

### Ваш первый сценарий

Традиционный первый проект на любом языке программирования отображает сообщение «Hello World!» В этом примере, вы создаете новый документ *Illustrator*, а затем добавляете текстовый фрейм, содержащий это сообщение. Следуйте этим шагам:

1. С помощью любого текстового редактора (включая *Adobe InDesign* или *ESTK*), введите следующий текст:

```
//Hello World!
//создаём новый документ и присваиваем его переменной "myDocument"
var myDocument = app.documents.add();
//создаём новый текстовый фрейм и присваиваем его переменной "myTextFrame"
var myTextFrame = myDocument.textFrames.add();
// устанавливаем содержание и положение текстового фрейма
myTextFrame.position = [0,300];
myTextFrame.contents = "Hello World!"
```

Для получения информации о том, как найти *ExtendScript Toolkit*, см. "JavaScript объектная модель" стр. 8.

2. Чтобы протестировать сценарий, выполните одно из следующих действий
  - Если вы используете *ESTK*, выберите *Adobe Illustrator CC 2017* из раскрывающегося списка в левом верхнем углу. В углу выберите «Да», чтобы запустить *Illustrator*, затем выберите *Debug > Run* в *ESTK* чтоб запустить скрипт.

- Если вы используете текстовый редактор, отличный от *ESTK*, сохраните файл как текстовый с расширением **.jsx** в папке по вашему выбору, затем запустите *Illustrator*. В *Illustrator* выберите «File»> «Scripts»> «Others Scripts», и перейдите к файлу сценария и запустите его.

**TIP:**

чтобы добавить сценарий в меню «Сценарии *Illustrator*» («File»> «Scripts»), сохраните сценарий в папке «Scripts». Сценарий появится в меню при следующем запуске *Illustrator*. Подробнее см. «Установка сценариев в меню Scripts» стр. 9.

## Добавление функции в «Hello World»

Затем мы создаем новый скрипт, который вносит изменения в документ *Illustrator*, который вы создали с помощью вашего первого скрипта. Наш второй пример демонстрирует, как:

- Получить активный документ.
- Получить ширину активного документа.
- Изменить размер текстового фрейма в соответствии с шириной документа.

Если вы уже закрыли документ *Illustrator*, запустите свой первый скрипт еще раз, чтобы создать новый документ, прежде чем продолжать это упражнение.

Следуйте этим шагам:

1. Выберите «Файл»> «Создать» в текстовом редакторе, чтобы создать новый скрипт.
2. Введите следующий код:

```
var docRef = app.activeDocument;
var docWidth = docRef.width
var frameRef = docRef.textFrames[0]
frameRef.width = docWidth
```

3. Запускаем скрипт.

## Работа с методами в JavaScript

Когда вы работаете с методами, имеющими несколько параметров, вы можете опустить необязательные параметры в конце списка параметров, но вы не можете пропустить параметры в середине списка. Если вы не хотите их изменять, укажите параметр в середине списка, на значение **undefined**, чтобы использовать значение параметра по умолчанию. Например, следующее определение описывает метод **rotate()** для арт-объекта.

```
rotate
  (angle
    [changePositions]
    [changeFillPatterns]
    [changeFillGradients]
    [changeStrokePattern]
    [,rotateAbout])
```

В определении, взятом из *Adobe Illustrator CC 2017 Scripting Reference: JavaScript*, необязательные параметры заключены в квадратные скобки (**[]**).

Чтобы повернуть объект на 30 градусов и изменить **fillGradients**, вы должны использовать следующий сценарий:

```
myObject.rotate(30, undefined, undefined, true);
```



Вы должны указать `undefined` для таких параметров как `changePositions` и `changeFillPatterns`. Вам не нужно ничего указывать для двух необязательных параметров после `changeFillGradients`, поскольку они находятся в конце списка параметров.

## Доступ к объектам или ссылки на них

Когда вы пишете сценарий, вы должны сначала решить, с каким файлом или `document`, сценарий должен работать. Через объект `application` сценарий может создать новый документ, открыть существующий документ, или действовать на документ, который уже открыт

Сценарий может создавать новые объекты в документе, работать с объектами, выбранными пользователем, или работать на объектах в одной из коллекций объектов. В следующих разделах показаны различные методы доступа, ссылки на них и управление ими.

### Ссылка на объект приложения

Чтобы получить ссылку на конкретный объект, вам необходимо перемещаться по иерархии вложения. Так как все скрипты *JavaScript* выполняются из приложения *Illustrator* конкретная ссылка на объект `application` не требуется. Например, чтобы назначить активный документ в *Illustrator* переменной `frontMostDocument`, вы можете сослаться на свойство `activeDocument` объекта `application` а именно:

```
var frontMostDocument = activeDocument;
```

Допускается использование объекта `application` в ссылке. Чтобы сослаться на объект `application`, используйте `app` глобальную переменную. Следующие два оператора выглядят идентичными движку *JavaScript*:

```
var frontMostDocument = activeDocument;
var frontMostDocument = app.activeDocument;
```

### Доступ к объектам коллекции

Все открытые документы, а также объекты в документе, собираются в объекты коллекции для каждого типа. Объект коллекции содержит массив объектов, к которым вы можете получить доступ по индексу или имени. Объект коллекции принимает форму множественного числа от имени объекта. Например, объект коллекции для `document` будет объект `documents`.

Следующий пример скрипта получает все объекты `graphic style` в `graphic styles` коллекциях; то есть получает все графические стили, доступные для активного документа:

```
var myStyles = app.activeDocument.graphicStyles;
```

Все числовые ссылки на коллекции в *JavaScript* начинаются с нуля: первый объект в коллекции имеет `index [0]`.

Как правило, номера индексов *JavaScript* не меняются при добавлении объекта в коллекцию. Существует одно исключение: `documents [0]` всегда является активным или самым передним документом.

Чтобы получить доступ к первому стилю в коллекции `graphic styles`, вы можете использовать переменную, объявленную в предыдущем примере скрипта, или вы можете использовать иерархию включения для ссылки на коллекцию:



- Использование переменной **myStyles**:

```
var firstStyle = myStyles[0];
```

- Использование иерархии включения:

```
var firstStyle = app.activeDocument.graphicStyles[0];
```

Следующие операторы присваивают переменной имя первого графического стиля в коллекции. Вы можете использовать эти утверждения как синонимы.

```
var styleName = myStyles[0].name
var styleName = firstStyle.name
var styleName = app.activeDocument.graphicStyles[0].name
```

Чтобы получить общее количество объектов в коллекции, используйте свойство **length**:

```
alert ( myStyles.length );
```

Индекс последнего графического стиля в коллекции - **myStyles.length-1** (-1, потому что коллекция начинает подсчет индекса с 0, а свойство **length** отсчитывается с 1):

```
var lastStyle = myStyles[ myStyles.length - 1 ];
```

Обратите внимание что выражение, представляющее значение индекса, заключено в квадратные скобки ( **[]** ), а также в кавычки. Если вы знаете имя объекта, вы можете получить доступ к объекту в коллекциях, используя имя в окружении квадратных скобок; Например:

```
var getStyle = myStyles["Ice Type"];
```

Каждый элемент в коллекции является объектом одинакового типа, и вы можете получить доступ к его свойствам через коллекцию. Например, чтобы получить имя объекта, используйте свойство **name**:

```
var styleName = app.activeDocument.graphicStyles[0].name;
```

Чтобы применить **lastStyle** к первому **pageItem** в документе, используйте метод **applyTo()**:

```
lastStyle.applyTo( app.activeDocument.pageItems[0] );
```

## Создание новых объектов

Вы можете использовать сценарий для создания новых объектов. Для создания объектов, доступных из объектов коллекции, или контейнеров используйте метод **add()**:

```
var myDoc = app.documents.add()
var myLayer = myDoc.layers.add()
```

Некоторые типы объектов недоступны из контейнеров. Чтобы создать объект этого типа, определите переменную, затем используйте оператор **new** с конструктором объектов, чтобы присвоить объекту значение. Например, чтобы создать новый объект **CMYKColor** используйте имя переменной **myColor**:

```
var myColor = new CMYKColor()
```

## Работа с выбранными объектами

Когда пользователь делает выбор в документе, выбранные объекты сохраняются в свойстве документа `selection`. Чтобы получить доступ ко всем выбранным объектам в активном документе:

```
var selectedObjects = app.activeDocument.selection;
```

Значение свойства `selection` может быть массив арт-объектов любого типа, в зависимости от того, какие типы объектов выбраны. Чтобы получить или изменить свойства выбранных арт-объектов, вы должны извлечь отдельные элементы из массива. Чтобы узнать тип объекта, используйте свойство `typename`.

В следующем примере получается первый объект в массиве, а затем отображается тип объекта:

```
var topObject = app.activeDocument.selection[0];
alert(topObject.typename)
```

Первый объект в массиве выбора - это выбранный объект, который был добавлен на страницу последним, а не последний выбранный объект.

### Выбор арт - объектов

Чтобы выбрать арт-объект, используйте свойство `selected`.

## Работа с текстовыми фреймами

Чтобы создать текстовый фрейм определенного типа в *JavaScript*, используйте свойство `kind` объекта `text frame`; Например:

```
var rectRef = docRef.pathItems.rectangle(700, 50, 100, 100);
// используем метод areaText для создания текстового фрейма
var areaTextRef = docRef.textFrames.areaText(rectRef);
```

## Связь фреймов в поток (Threaded frames)

Как и в приложении *Illustrator*, вы можете использовать текстовые фреймы области или текстовые фреймы контура. Чтобы связать существующие текстовые фреймы, используйте свойство `nextFrame` или `previousFrame` объекта `text frame`. При копировании следующего сценария в *ESTK* поместите значение свойства `contents` в одну строку.

```
var myDoc = documents.add();
var myPathItem1 = myDoc.pathItems.rectangle(244, 64, 82, 76);
var myTextFrame1 = myDoc.textFrames.areaText(myPathItem1);
var myPathItem2 = myDoc.pathItems.rectangle(144, 144, 42, 116);
var myTextFrame2 = myDoc.textFrames.areaText(myPathItem2);
// используем свойство nextFrame для создания потока текстового фрейма
myTextFrame1.nextFrame = myTextFrame2;
var sText = "This is two text frames linked together as one story, with text flowing from the first to the last. This is two text frames linked together as one story, with text flowing from the first to the last. This is two text frames linked together as one story.";
myTextFrame1.contents = sText;
redraw();
```

### Threaded frames make a single story object

Потоки фреймов составляют единый объект `story`. Чтобы увидеть это, запустите следующий *JavaScript* после запуска скрипта из *"Threaded frames" cmp. 31*.

```
var myDoc = app.activeDocument
alert("There are " + myDoc.textFrames.length + " text frames.")
alert("There are " + myDoc.stories.length + " stories.")
```

## Создание путей и фигур

В этом разделе объясняется, как создавать элементы, содержащие пути.

### Путь

Чтобы создать путь произвольной формы, укажите серию точек пути в виде серии координат (X, Y) или объект `pathPoint`.

Использование координат (X, Y) ограничивает путь прямыми сегментами. Чтобы создать изогнутый путь, вы должны создать объекты `pathPoint`. Ваш путь может состоять из комбинации координат и объектов `pathPoint` на странице.

#### Указание серии координат (X, Y)

Чтобы задать путь, на странице используя пары координат, используйте метод `setEntirePath()` объекта `pathItems`. Следующий скрипт задает три пары координат (X, Y) для создания пути с тремя точками:

```
var myDoc = app.documents add();
var myLine = myDoc.pathItems.add();
//устанавливаем обводку в true, чтобы мы могли видеть путь
myLine.stroked = true;
myLine.setEntirePath([[220, 475], [375, 300], [200, 300]]);
```

#### Использование объектов `pathPoint`

Когда вы создаете объект `pathPoint`, вы определяете три значения для точки:

- Фиксированная точка привязки `anchor`, точка на пути.
- Пара точек направления — `left direction` и `right direction` — которые позволяют контролировать кривую сегмента пути.

Вы определяете каждое свойство как массив координат страницы в формате [x, y].

- Если все три свойства объекта `pathPoint` имеют одинаковые координаты, а свойства следующего `pathPoint` в линии равны друг другу, вы создаете отрезок прямой.
- Если два или более свойств в объекте `pathPoint` имеют разные значения, сегмент, подключенный к точке изогнут.

Чтобы создать путь или добавить точки к существующему пути с помощью объектов `pathPoint`, создайте объект `pathItem`, затем добавьте точки пути как дочерние объекты в `pathItem`:

```
var myDoc = app.documents add();
var myLine = myDoc.pathItems.add();

// устанавливаем обводку в true, чтобы мы могли видеть путь
myLine.stroked = true;

var newPoint = myLine.pathPoints.add();
newPoint.anchor = [220, 475];
// присвоение указателям направления того же значения, что и
// точка привязки, создаём отрезок прямой линии
newPoint.leftDirection = newPoint.anchor;
newPoint.rightDirection = newPoint.anchor;
newPoint.pointType = PointType.CORNER;

var newPoint1 = myLine.pathPoints.add();
newPoint1.anchor = [375, 300];
newPoint1.leftDirection = newPoint1.anchor;
newPoint1.rightDirection = newPoint1.anchor;
newPoint1.pointType = PointType.CORNER;
```

```
var newPoint2 = myLine.pathPoints.add();
newPoint2.anchor = [220, 300];
// присваиваем точкам направления разные значения
// чем точка привязки, создаёт кривую
newPoint2.leftDirection = [180, 260];
newPoint2.rightDirection = [240, 320];
newPoint2.pointType = PointType.CORNER;
```

### Объединение типов точек пути (*Combining path point types*)

В следующем примере скрипта создается путь с тремя точками:

```
var myDoc = app.activeDocument;
var myLine = myDoc.pathItems.add();
myLine.stroked = true;
myLine.setEntirePath( [[220, 475], [375, 300]]);

// Добавляем еще одну точку к линии
var newPoint = myLine.pathPoints.add();
newPoint.anchor = [220, 300];
newPoint.leftDirection = newPoint.anchor;
newPoint.rightDirection = newPoint.anchor;
newPoint.pointType = PointType.CORNER;
```

## Фигуры (Shapes)

Чтобы создать фигуру, используйте метод `pathItems`, соответствующий имени фигуры (например, `ellipse`, `rectangle`, или `polygon`), и используйте свойства, чтобы указать положение, размер и другую информацию фигуры как количество сторон в многоугольнике.

Помните:

- Все измерения и координаты страницы обрабатываются механизмом скриптов как точки. Подробности см. “Единицы измерения” стр. 23.
- Координаты (*X*, *Y*) отсчитываются от нижнего левого угла документа, как указано в информационной панели в приложении Illustrator. Дополнительные сведения см. В разделе “Page-item размеры и расположение” стр. 24.

### Создание прямоугольника

Рассмотрим следующий пример

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var rect = artLayer.pathItems.rectangle( 144, 144, 72, 216 );
```

В примере используется метод `rectangle()` объекта `pathItems` для создания прямоугольника со следующими свойствами:

- Верх прямоугольника находится на расстоянии 2 дюймов (144 точки) от нижнего края страницы.
- Левый край находится на расстоянии 2 дюймов (144 точки) от левого края страницы.
- Прямоугольник имеет ширину 1 дюйм (72 точки) и длину 3 дюйма (216 точек).

### Создание многоугольника

Рассмотрим следующий пример:

```
var myDocument = app.documents.add()
var artLayer = myDocument.layers.add()
var poly = artLayer.pathItems.polygon( 144, 288, 72.0, 7 );
```

В примере используется метод `polygon()` для создания многоугольника со следующими свойствами:

- Центральная точка объекта находится на расстоянии 2 дюйма (144 точки) по горизонтальной оси и 4 дюйма (288 точек) по вертикальной оси.
- Длина радиуса от центральной точки до каждого угла составляет 1 дюйм (72 точки).
- У многоугольника 7 сторон.

## Работа с сеткой перспективы

Сетка перспективы - это новая функция в *Illustrator CC 2015*, которая позволяет создавать и управлять арт-объектами в пространственной среде, используя установленные законы перспективы. Включите сетку перспективы с помощью *View>Perspective Grid* или инструменты перспективы на панели инструментов (*Shift+P*). SDK предоставляет API для программной работы с сеткой перспективы, и ваши сценарии имеют некоторый доступ к этому API. Сценарий может:

- Устанавливать параметры сетки по умолчанию, используя предварительно заданные значения.
- Показать или скрыть сетку.
- Устанавливать активную плоскость.
- Рисовать объект в перспективе на активной плоскости.
- Переводить объект в перспективу.

## Предварительные настройки

*Illustrator* предоставляет стандартные наборы параметров сетки для одноточечной, двухточечной и трехточечной перспективы. Предусстановки (пресет) называются "[1P-NormalView]", "[2P-NormalView]", и "[3P-NormalView]". Этот сценарий показывает, как программно выбрать один из трёх вариантов настроек по умолчанию:

```
//Устанавливаем пресет одноточечной перспективы по умолчанию
app.activeDocument.selectPerspectivePreset("[1P-Normal View]");

//Устанавливаем пресет двухточечной перспективы по умолчанию
app.activeDocument.selectPerspectivePreset("[2P-Normal View]");

//Устанавливаем пресет трехточечной перспективы по умолчанию
app.activeDocument.selectPerspectivePreset("[3P-Normal View]");
```

Вы можете создавать новые шаблоны настроек перспективы, экспортировать шаблоны настроек в файлы и импортировать шаблоны настроек из файлов.

Эти сценарии показывают, как экспортировать и импортировать предустановки :

```
//Создаем новый документ
var mydoc = app.documents.add();

//Экспорт пресетов перспективы в файл
var exportPresetFile = new File("C:/scripting/PGPresetsExported")
mydoc.exportPerspectiveGridPreset(exportPresetFile);

//Создаем новый документ
var mydoc = app.documents.add();

//Импортируем предустановки перспективы из файла
var importPresetFile = new File("C:/scripting/PGPresets")
mydoc.importPerspectiveGridPreset(importPresetFile);
```

## Показать или скрыть сетку

Этот сценарий программно показывает или скрывает сетку перспективы:

```
//Показать сетку перспективы, определенную в документе
app.activeDocument.showPerspectiveGrid();

//Скрыть сетку перспективы, определенную в документе
mydoc.hidePerspectiveGrid();
```

## Установка активной плоскости (*Set the active plane*)

Типы плоскости сетки перспективы:

Left plane (левая)	<i>PerspectiveGridPlaneType.LEFTPLANE</i>
Right plane(правая)	<i>PerspectiveGridPlaneType.RIGHTPLANE</i>
Floor plane (надир)	<i>PerspectiveGridPlaneType.FLOORPLANE</i>
Invalid plane (зенит)	<i>PerspectiveGridPlaneType.NOPLANE</i>

Для одно точечной перспективной сетки допустимы только левая плоскость и *floor* (надир).

Этот скрипт устанавливает активную плоскость перспективы:

```
//Установить левую плоскость как активную
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);

//Установить правую плоскость как активную
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.RIGHTPLANE);

//Установить надир в качестве активной плоскости
app.activeDocument.setPerspectiveActivePlane(PerspectiveGridPlaneType.FLOORPLANE);
```

## Рисуем в сетке перспективы

Когда сетка перспективы включена, методы рисования позволяют рисовать объекты в перспективе или работать с ними. Этот сценарий создает новый документ, показывает сетку перспективы с двумя точками и рисует арт-объекты на левой плоскости.

```
//Создаем новый документ
var mydoc = app.documents.add();

//Выбираем предустановку двухточечной перспективы по умолчанию
mydoc.selectPerspectivePreset("[2P-Normal View]");

//Отображаем сетку перспективы, определенную в документе
mydoc.showPerspectiveGrid();

//Проверяем, установлена ли активная плоскость влево; если нет, устанавливаем влево
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE) {
    mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}

//Рисуем прямоугольник в перспективе, затем изменяем его размер до 200% и
//перемещаем
var myrect = mydoc.pathItems.rectangle(30, -30, 30, 30, false);
myrect.resize(200, 200, true, false, false, false, 100, Transformation.TOPLEFT);
myrect.translate (-420, 480);

//Рисуем эллипс в перспективе
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);

//Рисуем прямоугольник с закругленными углами в перспективе
var myrrect = mydoc.pathItems.roundedRectangle(90, -90, 30, 30, 10, 10, false);
```

```
//Рисуем многоугольник в перспективе
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);
//Рисуем звезду в перспективе
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);
//Рисуем путь в перспективе
var newPath = mydoc.pathItems.add();
var lineList = new Array(4);
lineList[0] = new Array(0,0);
lineList[1] = new Array(60,0);
lineList[2] = new Array(30,45);
lineList[3] = new Array(90,110);
newPath.setEntirePath(lineList);
```

## Перевести объекты в перспективу

Если арт-объект не в перспективе, используйте метод `bringInPerspective()` чтобы отобразить его в перспективе и поместите его на плоскость. Этот сценарий создает новый документ, рисует арт-объекты и отображает их в перспективе по трех точечной схеме сетки перспективы:

```
//Создаём новый документ
var mydoc = app.documents.add();
//Рисуем эллипс
var myellipse = mydoc.pathItems.ellipse(60, -60, 30, 30, false, true);
//Рисуем многоугольник
var mypoly = mydoc.pathItems.polygon(-105, 105, 15, 7, false);
//Рисуем звезду
var mystar = mydoc.pathItems.star(-135, 135, 15, 10, 6, false);
//Выбираем предустановку трехточечной перспективы по умолчанию
mydoc.selectPerspectivePreset("[3P-Normal View]");
//Отображаем сетку перспективы, определенную в документе
mydoc.showPerspectiveGrid();
//Проверяем, установлена ли активная плоскость влево; если нет, установите его влево
if (mydoc.getPerspectiveActivePlane() != PerspectiveGridPlaneType.LEFTPLANE)
{
mydoc.setPerspectiveActivePlane(PerspectiveGridPlaneType.LEFTPLANE);
}
//Переносим эллипс в активную плоскость (левую плоскость)
myellipse.bringInPerspective(-100,-100,PerspectiveGridPlaneType.LEFTPLANE);
//Перемещаем многоугольник в правую плоскость
mypoly.bringInPerspective(100,-100,PerspectiveGridPlaneType.RIGHTPLANE);
//Приносим звезду к плоскости надир
mystar.bringInPerspective(100,100,PerspectiveGridPlaneType.FLOORPLANE);
```