Рич Шуп, Зеван Россер

ИЗУЧАЕМ ActionScript 3.0

ОТ ПРОСТОГО К СЛОЖНОМУ

O'REILLY®

Adobe Developer Library

СИМВОЛ®

Learning ActionScript 3.0

A Beginner's Guide

Rich Shupe with Zevan Rosser

O'REILLY®

Мастера FLASH

Изучаем ActionScript 3.0

От простого к сложному

Рич Шуп, Зеван Россер

Символ®

Санкт-Петербург — Москва
2009

Серия «Мастера FLASH»

Рич Шуп, Зеван Россер

Изучаем ActionScript 3.0. От простого к сложному

Перевод А. Минаевой и Н. Шатохиной

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*К. Ковалев*
Редактор	*В. Подобед*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Шуп Р., Россер З.

Изучаем ActionScript 3.0. От простого к сложному. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 496 с., ил.

ISBN 978-5-93286-155-4

«Изучаем ActionScript 3.0» – лучшее руководство по ActionScript для начинающих пользователей, в котором представлено подробное описание механизма работы ActionScript и Flash. Четко и ясно изложены принципы построения кода, обработки событий, отображения контента, применения классов, описаны способы переноса проектов из предыдущих версий ActionScript и многое другое.

Авторы книги, Рич Шуп и Зеван Россер, имеют многолетний опыт не только разработки на ActionScript, но и преподавания этого языка. Они демонстрируют способы практического применения ActionScript для создания собственных проектов. Выполняя по мере прочтения книги предложенные упражнения, легко освоить новые приемы программирования. На сайте книги есть дополнительные материалы и упражнения, а также небольшие тесты для самопроверки.

Это пособие будет в равной степени полезно Flash-дизайнерам, разработчикам и начинающим программистам. Оно послужит незаменимым помощником на нелегком пути освоения ActionScript 3.0.

ISBN 978-5-93286-155-4
ISBN 978-0-596-52787-7 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 25.06.2009. Формат 70×100¹/16. Печать офсетная.
Объем 31 печ. л. Тираж 1500 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Adobe Developer Library (Библиотека Adobe для разработчиков), совместный проект O'Reilly Media, Inc. и Adobe Systems, Inc., – авторитетный источник для разработчиков, использующих технологии Adobe. Их всеобъемлющие ресурсы предлагают учебные решения, помогающие создавать передовые интерактивные веб-приложения, которые могут использоваться практически кем угодно на любой платформе.

Adobe Developer Library из первых рук представляет книги высочайшего качества и инновационные ресурсы, посвященные новейшим инструментальным средствам для разработки насыщенных интернет-приложений. Таким образом, *Adobe Developer Library* способствует подготовке высококвалифицированных специалистов. Область рассмотрения охватывает ActionScript, программное обеспечение Adobe Flex®, Adobe Flash® и Adobe Acrobat®.

Последние новости о книгах, сетевых ресурсах и прочее ищите на *adobedeveloperlibrary.com*.

Оглавление

Предисловие

Вы держите в руках эту книгу, раздумывая, сможет ли она стать вам надежным помощником. В этой связи вам может быть интересно, почему мы, авторы, написали именно *такую* книгу. Дело в том, что мы – не только разработчики, интенсивно применяющие Flash-технологии в своей работе, но и преподаватели. Вместе и порознь мы обучили тысячи студентов в различных университетах, учебных центрах и на конференциях – и на протяжении всей этой деятельности мы снова и снова сталкивались с одним и тем же значимым явлением: люди жаловались нам на отсутствие по-настоящему полного учебного пособия по ActionScript для начинающих.

Поначалу такое единодушие казалось нам удивительным, но мы понимали, что не располагаем достаточной информацией, чтобы составить собственное мнение. При проведении занятий мы опирались на учебный план, разработанный нами самостоятельно, и не использовали никаких пособий, предназначенных для новичков. Желая заполнить данный пробел и создать пособие для студентов, которое можно было бы использовать за пределами учебной аудитории, мы начали собственные изыскания и провели немало часов в беседах со студентами, преподавателями и пользователями, прежде чем смогли составить на этой основе набросок книги.

С появлением ActionScript 3.0 интерес к этой технологии существенно вырос. Объем нововведений, требовавших освоения с нуля, вызвал у пользователей неоднозначную реакцию – от восхищения до нерешительности и смятения. Многочисленные разговоры о разделении приверженцев платформы Flash на два лагеря – Flex-разработчиков и Flash-дизайнеров – внесли в души дизайнеров и начинающих программистов смуту и заставили их сильнее, чем когда-либо, терзаться сомнениями по поводу своего будущего. Выпуск среды разработки Flash CS3 Professional не только не устранил потребность в полноценном руководстве, но в некоторых случаях даже усилил ее, и мы почувствовали, что настало время представить книгу, которую вы держите в руках.

Мы надеемся, что эта книга поможет самым разным пользователям Flash – любопытным и робким, искушенным и начинающим – получить полное представление о возможностях и эффективности ActionScript 3.0. Эта книга станет вашим помощником при переходе от предыдущих версий ActionScript (если вы имеете опыт работы с ними)

к новой версии, в которой архитектура языка претерпела самые кардинальные изменения с момента его рождения.

Для кого предназначена эта книга

Эта книга рассчитана на Flash-дизайнеров и разработчиков, делающих первые шаги в изучении языка ActionScript 3.0, а также начинающих программистов, желающих освежить свои знания в этой области. Хотя основополагающие понятия представлены в книге достаточно подробно, мы все же предполагаем, что читатель знаком с интерфейсом Flash и имеет некоторый опыт написания сценариев.

Мы постарались изложить материал в простой и ясной форме, понятной любому, поэтому, даже если вы новичок в области программирования, эта книга – для вас! Однако если у вас есть пара минут, пролистайте главу 2, чтобы проверить, достаточно ли представленных там сведений об основах программирования для заполнения пробелов в ваших знаниях. Важные синтаксические конструкции подробно поясняются на протяжении всей книги, но первые две главы служат своеобразным фундаментом, на который опирается все последующее изложение.

Если у вас за плечами есть опыт программирования на ActionScript 2.0, вам стоит просмотреть те разделы, которые посвящены интересующим вас вопросам, чтобы решить, подойдет ли вам эта книга. Мы уделяем существенное внимание переходу от ActionScript 2.0 к ActionScript 3.0, но вам следует удостовериться, что принятый нами прямолинейный стиль представления материала окажется для вас подходящим. При работе над каждой главой мы стремились подобрать форму подачи материала, объем и последовательность изложения так, чтобы облегчить освоение основных принципов. Тем не менее вам все же стоит потратить несколько минут на чтение двух следующих разделов, чтобы развеять все сомнения в том, соответствует ли содержание книги вашим целям.

Структура книги

В отличие от других книг, посвященных ActionScript 3.0, в данной книге нет сильного акцента на принципах *объектно-ориентированного программирования (ООП)*. Если это понятие вам незнакомо, не волнуйтесь. Книга, которую вы держите в руках, станет вам надежным проводником, и с каждой следующей главой вы будете узнавать все больше и больше.

Для наглядного представления основных понятий мы используем синтаксис, основанный на выполнении операций в рамках временной диаграммы, попутно вводя ключевые понятия объектно-ориентированного программирования. В первых пяти главах – включая описание средств отображения контента (списка отображения) и новой мо-

дели событий в ActionScript 3.0 – классы и другие понятия объектно-ориентированного программирования упоминаются довольно редко. В главе 6 мы приступаем к более глубокому изучению объектно-ориентированного программирования, начав с самых азов и предлагая специально подобранные примеры с использованием классов или принципов ООП на протяжении всех последующих девяти глав.

Если вы хотите с самого начала работать с кодом, использующим ООП, вы можете скачать версии приводимых в данной книге примеров, задействующие классы. Эти варианты примеров не только станут для вас трамплином, если вы уже имеете опыт работы с объектно-ориентированными языками, но и послужат прекрасным материалом для самостоятельного изучения в том случае, если вы готовы двигаться с опережением. Более того, новая возможность среды разработки Flash CS3 Professional по созданию *класса документа* (Document Class) позволяет приступить к использованию классов быстрее, чем когда-либо прежде; при этом класс становится неким заменителем главной временной диаграммы любого файла с расширением *.fla* – для этого достаточно ввести имя класса в инспекторе свойств (Property Inspector) среды Flash. (Если вам не терпится узнать об этой возможности подробнее, обратитесь к разделу «Класс документа» в главе 1.)

И наконец, мы разработали масштабный проект, тесно связанный с материалом этой книги. Для каждой главы после главы 6, в которой разъясняются основы ООП, в составе загружаемого исходного кода есть пакет классов. Эти классы включают в себя полезные методы и свойства, используемые в учебном проекте. Освоив синтаксис ActionScript 3.0 и основные принципы объектно-ориентированного программирования, вы можете создать собственный проект, чтобы закрепить полученные знания. Все необходимые файлы находятся на сопроводительном веб-сайте книги, о котором мы расскажем чуть ниже.

Что есть (и чего нет) в этой книге

Мы приложили все усилия к тому, чтобы включить в эту книгу материал обо всех основных особенностях ActionScript, насколько позволят ее объем и тематический охват.

Что здесь есть...

Часть I. В начале начал

Часть I начинается с главы 1, где рассматриваются особенности версий ActionScript 1.0, 2.0 и 3.0, а также принципы их использования в среде разработки Flash CS3 Professional и воспроизведения с помощью Flash Player. За ней следует глава 2, в которой изложены сведения об основных строительных блоках программы – фундаментальных понятиях, не зависящих от используемого языка.

Часть II. Графика и взаимодействие

Часть II – самая объемная в книге – открывается главой 3, где представлены основные понятия языка ActionScript: свойства, методы и события (в том числе сильно измененная в ActionScript 3.0 модель событий). В главе 4 основное внимание уделяется динамическому отображению контента, глава 5 посвящена управлению временной диаграммой, а в главе 6 вводятся понятия ООП. Из главы 7 вы узнаете о том, как посредством ActionScript анимировать объекты, а главы 8 и 9 расскажут вам, как рисовать с помощью программного кода.

Часть III. Текст

Эта часть состоит из единственной главы – главы 10, посвященной форматированию текста, поддержке HTML и использованию каскадных таблиц стилей.

Часть IV. Звук и видео

Эту часть открывает глава 11, в которой обсуждается работа со звуком. Помимо манипуляций со встроенными и внешними звуковыми ресурсами в ней затрагивается тема синтаксического разбора метаданных ID3. В конце главы рассматривается пример визуализации звука, позволяющий «на лету» графически отображать форму звуковой волны в ходе воспроизведения. Завершается эта часть главой 12, в которой продемонстрированы возможности воспроизведения видео как с использованием компонентов, так и без них, а также показано, как снабдить видеоматериал субтитрами, чтобы сделать его доступным для людей с ограниченными возможностями и обеспечить поддержку нескольких языков.

Часть V. Ввод и вывод

Часть V посвящена загрузке ресурсов в Flash и пересылке данных серверу или другому клиенту. В главе 13 рассматриваются вопросы загрузки SWF-файлов, изображений и данных в формате, кодирования URL, описывается взаимодействие между ActionScript 3.0 и SWF-файлами, написанными с использованием предыдущих версий языка, а также кратко обсуждаются вопросы безопасности. Глава 14 посвящена языку XML и новым стандартам, благодаря которым работа с XML становится столь же простой, как манипуляция прочими объектами, методами и свойствами в ActionScript.

Часть VI. Проектирование программных продуктов и информационные ресурсы

Завершает книгу глава 15, в который представлен краткий обзор различных методологических подходов к программированию, рассмотрены объектно-ориентированные шаблоны проектирования и рассказано, где найти дополнительные материалы для изучения.

...и чего нет

Эта книга посвящена языку ActionScript 3.0, который относится к платформе Flash в целом, но здесь рассмотрен в контексте использования внутри среды разработки Flash CS3 Professional. Поэтому мы не касаемся Flex, AIR, Flash Media Server и других развивающихся технологий, основанных на Flash-платформе.

Далее, хотя мы и обсуждаем приемы объектно-ориентированного программирования, изложение не отличается большой глубиной. Более подробно об этом сказано в предыдущем разделе «Структура книги».

Ориентация книги на начинающих пользователей накладывает определенные ограничения на широту охвата материала. Оглавление поможет вам составить достаточно четкое представление о том, какие темы рассматриваются в этой книге, а во многих случаях – и о глубине их изложения. Однако некоторые важные области ActionScript здесь не затрагиваются вообще в силу их сложности. К ним относятся способы связи с базами данных, регулярные выражения, разработка программ для мобильных устройств, веб-сервисы, удаленный вызов методов с использованием формата AMF и создание собственных компонентов.

Эта книга не претендует на роль всеобъемлющего справочника. Если вы опытный программист на ActionScript, желающий быстро освоить версию 3.0, рекомендуем обратиться к книге «ActionScript 3.0 Cookbook» Джои Лотта (Joey Lott), Кейта Питерса (Keith Peters) и Деррона Шалла (Darron Schall).[1] Если вы ищете подробный справочник, охватывающий все аспекты использования языка, вас может заинтересовать книга «Essential ActionScript 3.0» Колина Мука (Colin Moock).[2] Наша книга является прекрасным дополнением к перечисленным выше, особенно если вы – начинающий программист, но не может в полной мере заменить эти книги.

Веб-сайт книги

Все упражнения, предлагаемые в этой книге, можно скачать с веб-сайта, расположенного по адресу *http://www.LearningActionScript3.com*. Там же вы найдете дополнительные материалы – упражнения, тесты для самопроверки, подробные примеры, советы, расширенный список ресурсов, комментарии читателей, список опечаток и многое другое. Мы планируем расширить сайт, добавив возможности для общения пользователей – например, форум, где мы сами также будем выступать участниками. Через этот сайт можно непосредственно связаться с нами.

---

[1] Джои Лотт, Деррон Шалл и Кейт Питерс «ActionScript 3.0. Сборник рецептов». – Пер. с англ. – СПб.: Символ-Плюс, 2007.

[2] Колин Мук «ActionScript 3.0 для Flash. Подробное руководство». – Пер. с англ. – СПб.: Питер, 2009.

Обозначения, принятые в книге

В этой книге мы придерживаемся следующих обозначений:

Рубленый шрифт

Применяется для обозначения заголовков, пунктов, кнопок меню и модификаторов клавиатуры (таких как Alt или Command).

Курсив

Используется для выделения новых понятий, URL-адресов, адресов электронной почты, имен и расширений файлов, каталогов и путей к файлам.

Моноширинный шрифт

Применяется для оформления кода на ActionScript, текста, выводимого при выполнении сценария, тегов XML и HTML, а также содержимого файлов.

Жирный моноширинный шрифт

Выделяет команды или иной текст, который должен быть введен без изменений.

Моноширинный курсив

Отмечает текст, который следует заменить на заданный пользователем.

Примечание

Так оформляется дополнительная информация (например, ссылка на материалы по теме или более подробное объяснение).

Внимание

Такой текст содержит предупреждение или предостережение.

Использование примеров кода

Эта книга будет служить подспорьем в вашей работе. В целом вы можете свободно использовать приведенный на ее страницах код в своих приложениях или документации. Если объем используемого вами кода невелик, нет необходимости обращаться к нам за разрешением. К примеру, вставка в вашу программу нескольких строчек кода из данной книги разрешения не требует. В то же время продажа или иное распространение CD-дисков с записью примеров, взятых из книг издательства O'Reilly, требует разрешения. Для цитирования материалов этой книги при ответе на вопрос специального разрешения не нужно, но при включении в составляемую вами документацию значительного объема кода из данной книги наше разрешение потребуется.

При использовании наших материалов мы не требуем обязательной ссылки на источник, однако будем вам за нее благодарны. Ссылка на источник, как правило, включает в себя название, имя автора, издательство и ISBN книги, например: «Learning ActionScript 3.0» by Rich Shupe and Zevan Rosser. Copyright 2008 O'Reilly Media, Inc., 978-0596527877.

Если вы чувствуете, что при использовании кода из книги вы вышли за рамки свободного использования, оговоренные выше, пожалуйста, свяжитесь с нами по адресу *permissions@oreilly.com*.

Нам интересно ваше мнение

Пожалуйста, присылайте свои вопросы и комментарии по поводу этой книги издателю по следующему адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (для звонков из США или Канады)
(707) 829-0515 (для международных или местных звонков)
(707) 829-0104 (факс)

На сайте издательства есть посвященная этой книге страница, где приводятся примеры, список опечаток и другие дополнительные сведения. Адрес страницы:

www.oreilly.com/catalog/ 9780596527877

Адрес электронной почты для комментариев и вопросов технического характера, связанных с этой книгой:

booksquestions@oreilly.com

Чтобы получить информацию о наших книгах, конференциях, ресурсных центрах и сети O'Reilly Network, посетите наш веб-сайт, расположенный по адресу:

www.oreilly.com

Благодарности

Рич и Зеван хотели бы выразить свою благодарность несравненной команде O'Reilly: Робину Томасу (Robyn Thomas), Стиву Вайссу (Steve Weiss), Мишель Филши (Michele Filshie), Мэттью Робертсу (Matthew Roberts), Джилл Штайнберг (Jill Steinberg), Джой Дин Ли (Joy Dean Lee), Рону Билодью (Ron Bilodeau), Филу Дэнглеру (Phil Dangler), Линде Зейферт (Linda Seifert), Марку Паглиетти (Mark Paglietti), Карен Монтгомери (Karen Montgomery) и Лори Петрицки (Laurie Petrycki). Эти чудесные люди работали над книгой не покладая рук и не разгибая спины, так что хруст позвонков порой был слышен на другом

конце материка. Лучших сотрудников просто невозможно себе представить. Отдельное спасибо Робину (Robyn) за бесконечное терпение и поддержку.

Зеван благодарит Рича Шупа, Школу изобразительных искусств (The School of Visual Arts), Джесси Резник (Jesse Reznick) и творческую группу SOM, Энн Орен (Ann Oren), своих студентов и членов своей семьи.

Рич благодарит Зевана Россера, Джоди Ротондо (Jodi Rotondo), Салли Шуп (Sally Shupe), Стивена Мэттсона Хейхёрста (Steven Mattson Hayhurst), Томаса Йе (Thomas Yeh), Аарона Крауча (Aaron Crouch), Аниту Рэмруп (Anita Ramroop) и членов своей семьи – без них эта книга никогда не была бы написана.

Рич также хотел бы выразить свою признательность:

- Брюсу Вэндсу (Bruce Wands), Джо Деллинджеру (Joe Dellinger), Рассету Ледермэну (Russet Lederman), Майку Баррону (Mike Barron), Джариду Лаудеру (Jaryd Lowder), Дайане Филд (Diane Field), Школе изобразительных искусств и своим студентам.
- Линде Вайнманн (Lynda Weinmann), Брюсу Хэвину (Bruce Heavin), Тоби Малина (Toby Malina), Кристофу Вайзе (Christoph Weise), Кевину Скоглунду (Kevin Skoglund) и всей команде FlashForward.
- Терри О'Доннеллу (Terry O'Donnell), Расселу Джонсу (Russell Jones) и DevX.com; Карен Шнайдер (Karen Schneider); Полу Кенту (Paul Kent), Кристен Маргулис (Kristen Margulis) и IDG; Джону Дэви (John Davey) и Flash on the Beach; Дэйву Шрёдеру (Dave Schroeder) и Flashbelt; Сьюзен Хоровиц (Susan Horowitz), Уильяму Моррисону (William Morrison) и организаторам программы поддержки Гавайского университета (University of Hawaii's Outreach).
- Майку Дауни (Mike Downey), Майку Чамберсу (Mike Chambers), Ричарду Гэлвану (Richard Galvan), Нивешу Раджбхандари (Nivesh Rajbhandari), Малли Гардинер (Mally Gardiner), Джеффу Камереру (Jeff Kamerer), Майклу Ниннессу (Michael Ninness), Джону Нэку (John Nack), Питу Фалько (Pete Falco) и Adobe.
- Эрал Бэлкан (Aral Balkan), Питу Барр-Уотсону (Pete Barr-Watson), Брендану Дейвсу (Brendan Dawes), Крису Джорджнесу (Chris Georgenes), Марио Клингеманну (Mario Klingemann), Зеб Ли-Делисл (Seb Lee-Delisle), Андре Мишель (AndreTMichelle), Эрику Нацке (Erik Natzke), Киту Питерсу (Keith Peters), Тиму Сэгунсину (Tim Saguinsin), Гранту Скиннеру (Grant Skinner), Крейгу Суонну (Craig Swann), Джарид Тарбелл (Jared Tarbell), Карлосу Уллоа (Carlos Ulloa) и всем, кого я мог случайно забыть, за вдохновение и поддержку.
- ...и, конечно, тебе, Мина! Эта книга – для Салли и?..

В этой части:

Глава 1 «Общий обзор особенностей ActionScript»

Глава 2 «Ключевые понятия языка»

I

В начале начал

Вступительная часть книги включает в себя главы 1 и 2, в которых представлен общий обзор основных понятий языка. Мы начнем с рассмотрения основных черт ActionScript, уделяя внимание особенностям новой версии, поговорим о различиях между процедурным и объектно-ориентированным подходами к программированию и сделаем несколько важных замечаний о структуре этой книги.

Завершается эта часть обзором фундаментальных принципов ActionScript, большинство из которых не изменяются от одной версии языка к другой. Тем, кто только начинает знакомство с языком ActionScript, этот материал послужит введением в тему, а опытным пользователям он поможет освежить свои знания, с тем чтобы дальше мы могли сосредоточиться на особенностях синтаксиса версии 3.0.

В этой главе:

- Что такое ActionScript 3.0?
- Платформа Flash
- Процедурный и объектно-ориентированный подходы к программированию
- Класс документа
- Совместимость с кодом для ранних версий

1

Общий обзор особенностей ActionScript

Скорее всего, вы уже знакомы с языком ActionScript, и вам не терпится начать работать с его новой версией. Краткий экскурс в историю языка поможет вам понять принципы его использования – в частности, в том, что касается работы с Flash Player и поддержки в нем различных версий ActionScript Эта краткая вводная глава позволит вам окинуть взглядом те области, в которых ActionScript 3.0 может стать подходящим инструментом для решения ваших задач. В ней рассматриваются следующие вопросы:

- **Что такое ActionScript 3.0?** Вполне естественно ожидать появления в новой версии языка новых возможностей. Но в версии 3.0 язык ActionScript был не просто обновлен – он был переписан с нуля и даже на этапе исполнения программы обрабатывается отличным от предыдущих версий образом. Такой намеренный «раскол» в среде Flash Player позволяет существенно поднять производительность, но накладывает определенные ограничения на взаимодействие различных версий ActionScript.
- **Платформа Flash**. Когда писались эти строки, ActionScript 3.0 являлся внутренним языком программирования для Flex и AIR (среда Adobe Integrated Runtime). Различия в процессе компиляции и специфические аспекты среды препятствуют исполнению кода на ActionScript 3.0 в других областях разработки на платформе Flash, однако ключевые понятия языка в большинстве своем остаются неизменными.
- **Процедурный и объектно-ориентированный подходы к программированию**. Мы много внимания уделяем объектно-ориентированным

возможностям ActionScript 3.0, и в этой области многочисленные достоинства и эффективность языка налицо. Однако вас наверняка обрадует тот факт, что для использования ActionScript 3.0 вовсе не обязательно становиться экспертом в области ООП. Проекты попрежнему можно создавать, используя структурированный набор функций, – в стиле, характерном для процедурного подхода к программированию. Flash CS3, как и раньше, допускает работу с кодом непосредственно на временной диаграмме, не требуя непременного использования внешних классов. Если же вы предпочитаете объектно-ориентированный подход, то усовершенствованная инфраструктура ООП в версии ActionScript 3.0, с точки зрения функциональности, ставит этот язык в один ряд с другими важными объектно-ориентированными языками программирования (такими, как Java), что помимо прочего облегчает переход с одного языка на другой.

- **Класс документа**. Не все стремятся овладеть объектно-ориентированным подходом к программированию, но если вы поставили перед собой такую цель, то благодаря появлению в Flash CS3 класса документа сможете с легкостью создавать объектно-ориентированные приложения. Класс документа является атрибутом панели инспектора свойств (Properties Inspector), поэтому достаточно лишь указать, какой внешний класс станет отправной точкой, – и необходимость писать сценарий временной диаграммы отпадает.
- **Совместимость с кодом для ранних версий**. Разработка проектов, поддерживающих код, написанный на предыдущих версиях языка, – задача весьма непростая, поскольку нельзя смешивать в одном файле код на ActionScript 3.0 и код для прежних версий. Мы вкратце остановимся на основных аспектах этой проблемы, которые затем более подробно обсудим в одной из последующих глав.

Что такое ActionScript 3.0?

Хотя большинство характерных черт очередной версии внутреннего языка сценариев Flash будут знакомы тем, кто имеет опыт работы с предыдущими версиями, по ряду вполне убедительных причин ActionScript 3.0 стоит рассматривать как совершенно новый язык. К ним прежде всего относятся существенные отличия модели событий и способа отображения графических ресурсов. Кроме того, язык претерпел множество менее заметных изменений, которые требуют определенного времени для привыкания. Сюда относятся, например, небольшие изменения в названиях некоторых свойств.

Однако важнее всего тот факт, что код, отвечающий за функционирование ActionScript 3.0, был полностью переработан. Произведенная таким образом оптимизация обеспечила существенный прирост производительности, но в то же время привела к тому, что код на ActionScript 3.0 нельзя смешивать с кодом, написанным для предыдущих версий этого языка, в пределах одного файла.

Однако бояться всех этих нововведений не стоит. Безусловно, ActionScript 3.0 стал более трудным в освоении по сравнению с предыдущими версиями языка, но это обусловлено скорее широтой его возможностей, нежели сложностью. Привыкание к новым особенностям языка просто потребует некоторого времени.

Чтобы все сказанное выше не вызвало у вас чувства растерянности, рассмотрим основные нововведения в версии ActionScript 3.0. Вам будет проще принять и освоить эти изменения, если вы будете ясно понимать, какие новые возможности они открывают, особенно в тех случаях, когда то или иное нововведение кажется излишним или чересчур усложненным. Среди основных новых функциональных возможностей:

Более подробный отчет об ошибках

Использование ActionScript 3.0 требует строгой типизации переменных, аргументов, возвращаемых функцией значений и т. д. Об этом подробнее говорится в главе 2, но по сути дела все сводится к тому, что компилятору необходимо знать, с каким типом данных вы работаете в каждый конкретный момент времени. Контроль типов данных был впервые введен в версии ActionScript 2.0, но до сих пор не носил обязательного характера. Более строгие требования к типизации данных помогают успешно выявлять ошибки и позволяют получить больше информации, необходимой для их исправления в коде. В дополнение к этому ActionScript 3.0 требует статической типизации на этапе исполнения. Это повышает достоверность типа данных во время исполнения, способствует увеличению производительности и сокращает объем требуемой памяти, поскольку при хранении информации о типах данных в машинном коде нет необходимости запрашивать ее динамически.

Синтаксические улучшения

Синтаксический строй языка в третьей версии был унифицирован и упорядочен. В частности, как вы увидите в главе 3, стали более ясными и непротиворечивыми названия некоторых свойств, что было достигнуто путем удаления начальных символов подчеркивания, использовавшихся несистематически. Были приведены также к общему знаменателю различные способы решения одинаковых или похожих задач, в том числе загрузка внешних ресурсов (более подробно об этом рассказано в главе 13) и обращение к URL-адресу (о чем вы узнаете дальше по ходу изложения).

Новые методы отображения данных

Многочисленные методы, которые в предыдущих версиях языка использовались для динамического добавления отображаемых данных, теперь сведены воедино. Благодаря новому списку отображения этот процесс стал гораздо проще; кроме того, теперь у разработчиков есть простой метод упорядочения визуального наложения отображаемых объектов, а также изменения иерархических отношений

родительских, дочерних и одноуровневых объектов. Это весьма важное нововведение ActionScript 3.0, и мы поговорим о нем более подробно в главе 4.

Новая модель событий

Все события теперь отслеживаются с помощью обработчиков, которые «прислушиваются» к событиям определенного типа и реагируют на них соответствующим образом. Это еще одно улучшение ActionScript 3.0, способствующее стабильности работы. Новая модель событий расширяет возможности разработчика, позволяя событиям клавиатуры или мыши распространяться по цепочке объектов в списке отображения. Особенности новой модели обработки событий подробно рассматриваются в главе 3.

Усовершенствованные методы работы с XML

Работа со сложными XML-документами, которая прежде была тягостной и трудоемкой, при использовании ActionScript 3.0 превратилась в удовольствие. Благодаря поддержке стандарта, известного как E4X, ActionScript позволяет работать с объектами XML гораздо более тонкими и удобными методами. Новый подход дает возможность использовать знакомый «точечный» синтаксис для вычленения сходных объектов в отдельную структуру.

Больше возможностей для обработки текстовых данных

Новые методы обработки текстовых данных позволяют точнее контролировать операции над текстом. Теперь можно найти текст в определенной строке текстового поля, вычислить количество символов в этой строке и определить символ, расположенный в заданном месте (например, под указателем мыши). Можно также получить индекс первого символа абзаца в тестовом поле и даже определить минимальный размер ограничивающего прямоугольника для любого символа. Наличие таких возможностей не только упрощает работу с текстовыми полями, но и позволяет выстраивать более тесное взаимодействие между строками и символами в текстовом поле и окружающими их элементами на сцене. Методы работы с текстовыми данными рассматриваются в главе 10.

Новые регулярные выражения

Возможности обработки текстовых данных стали еще шире благодаря поддержке регулярных выражений. Регулярные выражения – это работа с текстом «на стероидах». Вместо управления заранее известными строками символов вы можете теперь оперировать текстом, задавая символы замены, тип символов (цифры, буквы, знаки пунктуации и т. п.), специальные символы (пробелы, символы табуляции, символы новой строки), повторяющиеся группы символов и т. п. Простой пример использования регулярных выражений представлен в главе 10.

Дополнительные возможности для работы со звуком

Новые возможности ActionScript 3.0 в области работы со звуком сразу же привлекают к себе внимание. С практической точки зрения они облегчают доступ как к отдельным звуковым фрагментам, так и ко всем воспроизводимым звукам. Звуковые фрагменты теперь помещаются в индивидуальные каналы, что позволяет одновременно работать с отдельными звуками, а также управлять всеми используемыми звуками совместно, используя микшер. В процессе воспроизведения звуковых фрагментов можно получить данные об их амплитуде и частототном спектре. Работа со звуком подробно обсуждается в главе 11.

Новый метод доступа к необработанным данным

Для выполнения некоторых сложных операций теперь можно получить доступ к бинарным данным на этапе исполнения. В частности, вы можете прочитать отдельные байты данных при загрузке, воспроизведении звука или в процессе различных операций с растровыми изображениями. Их можно сохранить для быстрого и удобного доступа к ним в дальнейшем. Пример использования приема такого рода показан в главе 11 при обсуждении визуализации звука.

Автоматическое определение области видимости

Областью видимости в программировании иногда называют область, которой принадлежит объект. Некоторый Flash-ресурс, скажем клип (movie clip), может быть доступен в одной части Flash-ролика, но недоступен в другой. Рассмотрим, к примеру, дочерний клип, вложенный внутрь одного из двух клипов в составе основной временной диаграммы. Такой вложенный клип будет существовать в рамках одного клипа, но не второго. Тем самым его область видимости ограничена родительским элементом. Программные структуры тоже обладают ограниченной областью видимости, и одна из трудностей программирования – необходимость следить за тем, находитесь ли вы в области видимости той или иной структуры при обращении к ней. ActionScript 3.0 облегчает жизнь разработчика, автоматически отслеживая область видимости в процессе написания программы.

Улучшенная поддержка ООП

Объектно-ориентированные программные конструкции в ActionScript 3.0 также были значительно усовершенствованы. Среди улучшений необходимо упомянуть, среди прочего, запечатанные классы и новые пространства имен. Основные аспекты ООП мы обсудим в данной главе и в главе 6, а на протяжении всей книги будем демонстрировать примеры, основанные на использовании классов. В ActionScript 3.0 все классы по умолчанию являются запечатанными; это означает, что на этапе исполнения существуют только те свойства и методы класса, которые были определены на этапе разработки.

Если необходимо иметь возможность изменять класс в процессе выполнения программы – например, если может возникнуть потребность добавить какие-либо свойства, – это по-прежнему можно обеспечить, объявив класс динамическим. Наконец, благодаря пространствам имен (в том числе возможности определения собственных пространств имен) оперирование классами и XML-данными стало еще более тонким и точным.

Платформа Flash

Темой данной книги является разработка приложений на языке ActionScript 3.0 с использованием интегрированной среды разработки (IDE – integrated development environment) Flash CS3 Professional. Однако ActionScript 3.0 служит языком программирования и для других областей применения платформы Flash; здесь прежде всего следует упомянуть среды Flex- и AIR (Adobe Integrated Runtime – кросс-платформенная среда для запуска приложений).

Примечание

Проекты AIR могут включать в себя элементы HTML, JavaScript и PDF, но их привлекательность в большой степени обусловлена использованием ActionScript 3.0, и именно этот язык нас интересует в контексте данной книги.

Это означает, что приобретенные вами навыки написания сценариев с помощью Flash CS3 во многих случаях можно применять в других областях разработки под Flash-платформу, что расширяет ваши возможности как программиста. При этом необходимо учитывать некоторые важные различия, которые сразу становятся заметными при более широком взгляде на разработку сценариев для различных приложений. Для этого вкратце остановимся на нескольких аспектах проблемы.

Прежде всего, Flash и Flex используют различные компиляторы, и нет никакой гарантии того, что ваш проект можно будет успешно скомпилировать в обеих средах. Flex Builder (компилятор Flex) позволяет скомпилировать код на «чистом» ActionScript без использования особенностей именно Flex и загрузить скомпилированный SWF-файл в проект, созданный в Flash CS3. Аналогичным образом можно загружать SWF-файлы, скомпилированные с помощью Flash CS3, в проекты Flex. Однако как только базовых функций языка для ваших нужд становится недостаточно, возникают осложнения.

К примеру, Flex не имеет доступа к средствам Flash IDE, используемым для создания визуальных ресурсов (таких, как клипы), а Flash, в свою очередь, не поддерживает использование тега Embed, предназначенного для включения таких ресурсов во Flex-приложение. Это означает, что один и тот же код вряд ли получится беспрепятственно использовать и здесь, и там, если в приложение включены ресурсы такого рода.

В компонентной архитектуре обнаруживаются аналогичные расхождения, в том числе несоответствие форматов и наборов компонентов.

Этот вопрос некоторое время вызывал оживленные дискуссии, но постепенно острые углы проблемы сглаживаются. Adobe создала «заплатку» для Flex 2, а Flex 3, в котором совместимость компонентов существенно улучшена, в момент написания этой книги проходил публичное тестирование.[1] Однако потребуется, вероятно, еще немало времени, чтобы использование одного и того же кода для этих приложений стало простым и удобным делом (если это вообще возможно). В то же время приложения, разрабатываемые под AIR, быстро обретают черты универсальности. Adobe продолжает совершенствовать рабочий цикл создания AIR-приложений в среде Flash CS3.[2]

Тем не менее навыки программирования на ActionScript 3.0 существенно упростят вам процесс перехода из одной среды, использующей Flash-платформу, в другую, даже если в процессе разработки вам придется иметь дело с отличающимся инструментами и компиляторами.

Процедурный и объектно-ориентированный подходы к программированию

Преимуществам и недостаткам процедурного и объектно-ориентированного подходов к программированию было посвящено немало дискуссий. Мы вкратце рассмотрим эту тему в свете истории развития языка ActionScript.

Изначально язык ActionScript подразумевал *последовательное* программирование, то есть написание сценария сводилось к созданию линейной последовательности пошаговых инструкций для Flash. Такой способ программирования не отличался гибкостью и не способствовал повторному использованию фрагментов кода.

По мере своего развития ActionScript приобретал черты *процедурного* языка программирования. Как и последовательное программирование, процедурный подход основан на пошаговой последовательности команд, но отличается более высокой степенью структурированности, модульности сценариев. *Процедуры*, также называемые функциями (а иногда – подпрограммами), могут выполняться многократно и вызываться из различных частей проекта; при этом нет необходимости копировать соответствующий фрагмент кода в текущую последовательность инструкций. Деление кода на модули позволяет повторно

---

[1] Когда шла работа над русским переводом книги, Flex 3 уже вышел и были доступны предварительные версии Flex 4. – *Примеч. науч. ред.*

[2] К моменту завершения работ над русским переводом книги среда Flash была полностью приспособлена для создания AIR-приложений. – *Примеч. науч. ред.*

использовать определенные фрагменты кода, упрощает процесс редактирования программы и делает работу более эффективной.

В попытках довести принцип деления программы на модули и возможность повторного использования кода до совершенства программисты пришли к созданию *объектно-ориентированного* подхода в программировании. Программы, написанные на объектно-ориентированных языках, представляют собой совокупность объектов. Объекты являются экземплярами *классов* – блоков самодостаточного кода, которые не могут изменить или разрушить друг друга. Разбиение кода на небольшие независимые части, известное как *инкапсуляция,* является одним из отличительных признаков объектной ориентированности языка. Еще одним важным признаком ООП служит наличие механизма *наследования*, т. е. возможности создания классов на основе родительских классов путем передачи определенных свойств последних.

Классическим примером, раскрывающим суть ООП и в частности принципов наследования, является описание набора транспортных средств. Начнем с общего класса `Vehicle` (транспортное средство), обладающего чертами, общими для всех транспортных средств (например, основными физическими законами движения). На его основе можно создать три подкласса: `GroundVehicle` (наземное транспортное средство), `WaterVehicle` (водное транспортное средство) и `AirVehicle` (воздушное транспортное средство). Эти классы соответствующим образом изменяют свойства (или вводят новые), чтобы отразить специфику перемещения по земле, по воде и по воздуху соответственно, но их еще недостаточно для определения настоящего транспортного средства. Далее можно создать классы `Car` (автомобиль) и `Motorcycle` (мотоцикл) на основе класса `GroundVehicle`, `Boat` (лодка) и `Submarine` (подводная лодка) на основе класса WaterVehicle, а также `Plane` (самолет) и `Helicopter` (вертолет) на основе класса `AirVehicle`. В зависимости от сложности вашей программы данный процесс можно продолжить, создав модели с индивидуальным набором свойств, касающихся потребления топлива, сил трения и т. п.

Нетрудно заметить, что такой подход к разработке программ обладает еще большей гибкостью и открывает еще больше возможностей, в том числе для повторного использования кода. Благодаря указанным преимуществам (а также многим другим плюсам), объектно-ориентированное программирование нередко становится наилучшим подходом к решению задачи. Однако некоторые программисты склонны считать этот метод лучшим (или даже, сверх того, единственным) решением всех проблем. Это заблуждение.

Объектно-ориентированный подход к программированию, как правило, весьма эффективен в случае крупных проектов или при работе команды программистов, но для небольших проектов зачастую является излишним. Кроме того, если вы не знакомы с ООП, изучение этого подхода может существенно увеличить продолжительность обучения

и отвлечь внимание от других ключевых понятий. В общем, ООП вовсе не является универсальным инструментом, подходящим для решения любой задачи. Процедурный подход к программированию по-прежнему имеет право на существование, и при использовании Flash CS3 можно изучать и применять оба подхода к разработке проектов.

В этой книге используются как процедурный, так и объектно-ориентированный подходы к программированию, в зависимости от характера решаемых задач. Освоение объектно-ориентированных приемов программирования – достойная цель, и мы держим ее в поле зрения. Однако мы стараемся в каждой главе первым делом уделять внимание основной теме, особенностям синтаксиса и способам применения рассматриваемого материала при написании кода.

В целом до главы 6 мы будем отдавать предпочтение процедурному подходу к программированию; глава 6 послужит своего рода мостиком, позволяющим перейти с берега процедурного программирования на берег ООП. В главе 7 и следующих за ней начало каждой главы отводится под обсуждение определенной темы без погружения в структуру соответствующих классов, а в завершающей части главы (там, где это уместно) приводится практический пример с использованием приемов ООП.

Мы предпочитаем именно такой подход, комбинирующий оба метода программирования – процедурный и объектно-ориентированный. Это позволяет представить материал в форме, доступной любому пользователю. Мы надеемся, что вне зависимости от ваших навыков и опыта вы сможете быстро освоить предложенные нами темы и вскоре начнете создавать собственные программы в том стиле, который вам ближе – с помощью временной диаграммы или же с использованием классов.

Класс документа

Если вам не терпится погрузиться в мир объектно-ориентированного программирования, мы покажем вам, как сделать первые шаги в этом направлении. В Flash CS3 появилась новая возможность, упрощающая связывание главного класса (или точки входа приложения) с вашим FLA-файлом. Она называется *классом документа* и отвечает за выполнение всех необходимых действий по работе с главным классом. Это позволяет полностью обойтись без написания кода на временной диаграмме; а для редактирования программы использовать не только Flash, но и любой внешний текстовый редактор или среду разработки на ваше усмотрение.

Примечание

Если вы предпочитаете не касаться методов ООП до тех пор, пока мы не дойдем до соответствующей темы, можете просто пропустить данный раздел, поскольку мы вернемся к этому вопросу в главе 6. Здесь представлен минимум информации, необходимый для того, чтобы вы могли приступить к использованию

класса документа; в последующих главах эта возможность будет рассмотрена гораздо подробнее.

Начнем с учебного примера, который можно использовать на временной диаграмме. Он выполняет единственное действие – вызывает метод `trace()` для отображения одного слова на панели Output, предназначенной исключительно для разработчика и принимающей текст, выводимый программой.

```
trace("Flash");
```

Чтобы получить класс документа, необходимо создать некую обертку для вызова метода `trace()` в соответствии с синтаксическими правилами языка.

Создайте новый файл ActionScript (не FLA-документ) и вставьте в него следующий обертывающий код:

```
1    package {
2
3      import flash.display.MovieClip;
4
5      public class Main extends MovieClip {
6
7        public function Main() {
8
9        }
10
11     }
12   }
```

В первой строке содержится определение *пакета* класса, которое заканчивается фигурной скобкой в строке 12. Наличие определения пакета класса обязательно, поскольку позволяет объяснить компилятору, с чем он имеет дело. Далее необходимо импортировать классы, которые будут использованы в пакете.

Класс документа по сути дела служит методом быстрого создания экземпляра клипа или спрайта (новый объект Flash, представляющий собой попросту однокадровый клип) и добавления его в список отображения Flash Player. (Это верно и в том случае, когда отображаемых объектов нет, как в нашем примере. О работе со списком отображения мы поговорим в главе 4.).

Любой класс документа должен быть производным от класса `MovieClip` или класса `Sprite`. (Прочие создаваемые пользователем классы, не являющиеся классами документа, не обязаны быть расширением классов `MovieClip` или `Sprite`, если это не соответствует целям программы.) Поскольку в данном примере используется класс `MovieClip`, его необходимо импортировать, что и происходит в строке 3.

В строке 5 расположено описание класса, которое завершается фигурной скобкой в строке 11. Имя класса может быть произвольным, но должно следовать следующим простым правилам и соглашениям: оно должно состоять из одного слова, не используемого в ActionScript, начинаться с буквы (а не цифры или иного символа) – обычно заглавной. Класс должен быть публичным – то есть другие классы должны иметь доступ к его конструктору, и он должен расширять класс `MovieClip` или `Sprite`, как было сказано выше.

В строке 7 начинается *конструктор* класса, завершающийся закрывающей фигурной скобкой в строке 9. Это основная функция, запускаемая автоматически при создании экземпляра данного класса. Она также должна быть публичной и называться тем же именем, что и класс. Остальные функции (если они имеются) могут и должны иметь уникальные имена. В нашем примере остается лишь добавить единственный требуемый вызов метода. Конструктор класса должен обеспечить отображение слова «Flash» на панели Output. Для этого в строку 8 добавьте следующий код:

```
7        public function Main() {
8          trace("Flash");
9        }
```

По завершении необходимо сохранить файл в той же папке, где будет расположен FLA-файл. (Чуть позже вы узнаете о том, как размещать файлы классов в других местах.) Этому файлу нужно дать то же имя, которым назван класс, и добавить расширение имени *.as*.

Таким образом, наш файл будет называться *Main.as*. Теперь создайте новый FLA-файл, указав в качестве версии используемого языка программирования ActionScript 3.0, и сохраните его в той же папке, что и созданный ранее файл класса. Имя FLA-файла не имеет принципиального значения.

В заключение откройте инспектор свойств (Properties Inspector) и укажите в поле Document Class имя вашего класса документа (но не самого файла). В нашем примере это будет Main, но не Main.as (рис. 1.1).

Теперь откройте свой файл в режиме предварительного просмотра. При этом будет создан и добавлен в список отображения экземпляр

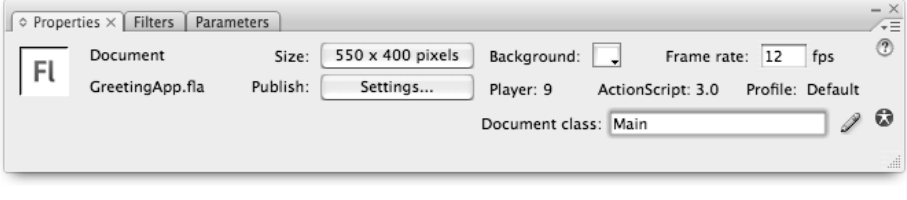

Рис. 1.1. *Добавление класса документа в FLA-файл*

класса `Main` (который ведет себя как клип, поскольку основан на классе `MovieClip`). Класс отобразит текст «Flash» на панели вывода, и выполнение тестового приложения будет на этом закончено.

Теперь вы сможете самостоятельно добавлять в класс документа любой код временной диаграммы, приведенный в нашей книге. Поначалу у вас могут возникнуть затруднения с тем, какие классы необходимо импортировать или как внести некоторые изменения в переменные и подобные структуры в соответствии с правилами синтаксиса для класса. Однако к коду всех предлагаемых в книге примеров прилагаются соответствующие файлы класса для тестирования. Вы можете спокойно пользоваться этими файлам, пока не привыкнете к формату класса документа.

Совместимость с кодом для ранних версий

В заключение главы хотелось бы сделать небольшое предупреждение: в пределах одного SWF-файла нельзя использовать код на ActionScript 3.0 совместно с кодом, написанным для предыдущих версий языка – ActionScript 1.0 или 2.0. Если вы только начинаете знакомство с ActionScript, это вряд ли будет для вас актуальным, но при обновлении существующих проектов путем добавления кода на ActionScript 3.0 об этом необходимо помнить, чтобы не столкнуться с проблемами.

Если у вас возникла необходимость совместно использовать ActionScript 3.0 и предыдущие версии языка (например, в том случае, когда нужно отобразить уже имеющийся файл в интерфейсе нового ролика), вы можете сделать это путем загрузки в приложение SWF-файла. В файл, написанный на ActionScript 3.0, можно загрузить SWF-файл, созданный с использованием ActionScript 1.0 или 2.0, однако при этом код на ActionScript 3.0 не сможет обращаться к переменным и функциям из старого SWF-файла. В обратном направлении этот прием не действует ни при каких условиях: в файлы, написанные на предыдущих версиях языка, нельзя загрузить SWF-файл на ActionScript 3.0.

В главе 13 мы обсудим специальные методы взаимодействия между такими SWF-файлами. На данный момент просто примите к сведению, что код на языке ActionScript 3.0 нельзя смешивать с кодом для ранних версий в пределах одного файла.

Что дальше?

Теперь, когда первое знакомство с ActionScript 3.0 и платформой Flash состоялось, самое время рассказать о ключевых понятиях языка. Мы начнем с основ, не зависящих от версии языка, а в последующих главах сконцентрируемся на новом синтаксисе версии 3.0. Если вы имеете богатый опыт работы с ActionScript 1.0 или 2.0, можете бегло пролистать этот раздел.

В следующей главе мы обсудим:

- Основные понятия, необходимые для того, чтобы приступить к работе как можно скорее, в том числе метод `trace()`, который используется в качестве диагностического инструмента, позволяющего немедленно получить ответную реакцию приложения.
- Использование переменных для хранения данных, включая массивы и произвольные объекты, что позволяет оперировать более чем одним значением одновременно, а также типизацию этих данных, позволяющую эффективно выявлять ошибки.
- Важные логические структуры, такие как условные операторы для выбора выполняемых действий и циклы, упрощающие выполнение повторяющихся задач.
- Использование функций для выделения фрагментов кода в отдельные блоки, которые выполняются строго при вызове соответствующей функции.
- Способы обращения к объектам Flash посредством ActionScript, в том числе использование относительных и абсолютных путей и идентификатора `this`.

2

В этой главе:
- Некоторые общие принципы
- Переменные и типы данных
- Условные операторы
- Циклы
- Массивы
- Функции
- Объекты, создаваемые пользователем
- this
- Абсолютный и относительный пути

Ключевые понятия языка

ActionScript 3.0 – кардинальным образом переработанная версия внутреннего языка сценариев Flash, и за ее фасадом находится код, существенно отличающийся от кода предыдущих версий. Однако сейчас эти подробности являются для нас второстепенными – более важно то, что все существующие на данный момент версии ActionScript обладают рядом сходных черт.

Этого следовало ожидать, поскольку в основе всех версий ActionScript лежит один и тот же стандарт языков сценариев (называемый ECMA-262), который получил широкое распространение благодаря успеху JavaScript, и текущие версии ActionScript во многом сохраняют обратную совместимость для поддержки проектов, выполненных в предыдущих версиях.

Это вовсе не означает, что язык остановился в своем развитии. Безусловно, с каждой следующей версией в ActionScript, как и в других языках программирования, появляется множество новых возможностей. А поскольку последняя версия была переписана заново, возник хороший повод избавиться от ряда досадных мелочей, доставшихся в наследство от прежних версий, а именно – потребовать более строгого соблюдения принципов, ранее представленных в качестве необязательных, и реструктурировать модель событий и систему отображения данных.

Эти изменения, однако, не затронули стандарт, лежащий в основе ActionScript, и большинство основных принципов языка остались неизменными. В дальнейшем мы вплотную займемся новыми возможно-

стями ActionScript, но прежде нам хотелось бы рассмотреть упомянутые выше общие понятия. Естественно, в последующих главах мы еще будем возвращаться к ним, но, поскольку эти понятия являются ключевыми, мы постарались дать развернутые объяснения здесь, чтобы в дальнейшем уделять больше внимания другим темам.

Если вы уже хорошо знакомы с ActionScript и используете данную книгу в качестве трамплина для освоения новой версии, можете бегло пролистать эту главу или даже пропустить ее вовсе. Она ни в коем случае не претендует на роль полноценного вводного курса. При ее написании мы не предполагали, что у читателя есть опыт работы с какой-либо из предыдущих версий ActionScript, но, учитывая цель и объем книги, подразумевали, что читатель имеет общее представление об основных идеях создания сценариев. Если в вашем случае это не так, обратитесь к предисловию, где подробно разъясняется, для кого предназначена данная книга, а также приводятся ссылки на дополнительные материалы, которые помогут вам заполнить пробелы в ваших знаниях.

Эту главу можно рассматривать как своего рода справочник: вы всегда можете обратиться к ней, если вам потребуется прояснить для себя какие-либо из ключевых понятий программирования. Здесь обсуждаются следующие темы:

- **Некоторые общие принципы.** Есть ряд приемов и принципов, которые применяются на протяжении всей книги, но при этом не настолько сложны, чтобы отводить под каждый из них целый раздел. С их рассмотрения мы и начинаем эту главу.
- **Переменные и типы данных.** В тех случаях, когда данные могут понадобиться программе позже, их сохраняют в контейнерах, называемых переменными. Заранее задав для каждой переменной тип данных, которые будут в ней храниться, вы поможете программе Flash отслеживать ошибки в процессе разработки.
- **Условные операторы.** При выполнении сценария часто возникает необходимость сделать выбор. Для этого используется условный оператор, который принимает решение в зависимости от определенного набора условий. Мы рассмотрим условные операторы `if` и `switch`.
- **Циклы.** Если команду необходимо выполнить несколько раз, удобно использовать циклическую структуру. Мы остановимся на наиболее часто используемой структуре `for`, но не обойдем вниманием также и варианты, альтернативные явно заданным циклам, в том числе события кадра и таймера.
- **Массивы.** Обычная переменная может содержать лишь одно значение, но часто бывает удобно (а порой даже необходимо) хранить в переменной несколько значений. В качестве аналогии из жизни можно привести список покупок, в котором несколько предметов перечислены на единственном листке бумаги. Массив – это структура

данных, позволяющая хранить несколько значений в одной переменной.

- **Функции.** Функции – непременный элемент практически любого языка программирования. Их использование позволяет выполнять определенный фрагмент кода многократно и только тогда, когда это нужно.
- **Объекты, создаваемые пользователем.** Такой объект похож на сложную переменную для хранения большого количества данных с логичной и легкой системой доступа к ним. Объекты можно эффективно использовать для передачи в функцию большого набора необязательных значений, что значительно упрощает данную задачу.
- **this.** Ключевое слово `this` используется в качестве сокращенной ссылки, указывающей на текущий объект или область видимости сценария. В конкретном контексте это понятие становится более ясным, но если вы разберетесь, как оно используется, это сэкономит вам массу нажатий на клавиши и упростит ссылки в ваших сценариях.
- **Абсолютный и относительный пути.** Путь к объектам в ActionScript может быть абсолютным, т. е. начинаться от корневого уровня временной диаграммы и включать в себя все промежуточные объекты в иерархии вплоть до целевого, или же относительным, который указывает, что попасть, скажем, к «сестринскому» элементу можно, поднявшись на уровень выше, к родительскому элементу, а затем опустившись на уровень вниз.

Еще раз повторим: если у вас нет никакого опыта работы с ActionScript, вам не стоит полагаться на эту главу как на единственный источник базовых знаний. Скорее всего, вы найдете в ней большую часть необходимых сведений, но некоторые основополагающие принципы – например, информация о том, где сценарии хранятся в интерфейсе Flash, – не были включены сюда в силу ограниченного объема книги.

Для знакомства с интерфейсом Flash прекрасно подойдет книга «Flash CS3 Professional, The Missing Manual», опубликованная издательством O'Reilly. Что касается полноценного справочного руководства по ActionScript, то, как уже говорилось в предисловии, мы без раздумий рекомендуем книгу Колина Мука «Essential ActionScript 3.0».[1] Последняя книга рассчитана главным образом на пользователей со средним и высоким уровнем подготовки, однако ее объем почти в три раза превышает объем данной книги, что дает ей возможность играть роль полноценного справочника.

В целом информации, содержащейся в этой главе, а также дополнительных объяснений, представленных в последующих главах, должно

[1] Колин Мук «ActionScript 3.0 для Flash. Подробное руководство». – Пер. с англ. – СПб.: Питер, 2009.

быть достаточно, чтобы вы смогли понять рассматриваемые темы и заставить работать учебные примеры, приведенные в книге.

Некоторые общие принципы

Существует несколько основополагающих принципов, о которых нельзя не упомянуть, поскольку они будут постоянно использоваться на протяжении всей книги. В то же время посвящать каждому из них отдельный раздел представляется нецелесообразным. Чтобы вы могли составить представление о некоторых из них, здесь собраны несколько примеров.

Порядок выполнения операций

Как правило, операции в программе на ActionScript выполняются по принципу «сверху вниз и слева направо», то есть каждая строка интерпретируется слева направо, а затем осуществляется переход к следующей строке. В некоторых ситуациях такой порядок может быть незначительно изменен, но обычно это правило можно считать незыблемым. К примеру, если в середине выполнения программы запущена та или иная внешняя подпрограмма, это приведет к паузе в выполнении основного сценария. По завершении подпрограммы выполнение сценария продолжится с того места, где была сделана пауза. О порядке выполнения еще будет сказано при рассмотрении практических примеров, представленных в книге.

Использование точки с запятой (;)

Согласно правилам ActionScript точка с запятой используется для разделения нескольких команд, расположенных в одной строке. В реальных сценариях этот прием используется довольно редко; мы рассмотрим его подробнее при разговоре о циклах. Помимо этого точка с запятой применяется для обозначения окончания строки. Это не обязательно, но желательно, поскольку вносит большую ясность в код и облегчает возможный переход к изучению других языков, в которых это правило соблюдается строго.

Интерпретация выражений

Стоит упомянуть, что выражение, состоящее из двух частей, разделенных знаком равенства, не следует расценивать как уравнение. К примеру, когда вы видите что-то вроде **x = x + 1**, это не означает, что вам нужно вычислить значение **x**. В действительности в данной строке происходит присваивание переменной **x** нового значения путем увеличения ее предыдущего значения на единицу.

Использование команды `trace`

Команда `trace` очень удобна в качестве инструмента быстрого получения информации в учебных примерах, а также для тестирования и отладки сценариев. Она выводит указанный текст на панель Output, входящую в состав интерфейса Flash. Эта возможность доступна

только во время разработки – на этапе исполнения приложения в ней нет смысла.

Переменные и типы данных

Проще всего думать о переменной как о контейнере, в который вы помещаете информацию, чтобы обратиться в ней позже. Представьте себе, что у вас не было бы возможности сохранять информацию для дальнейшего использования. Тогда вы не могли бы сравнивать текущие значения (например, имя пользователя или пароль) с данными, предоставленными ранее, ваши сценарии выполнялись бы гораздо дольше, поскольку много раз повторяли бы одни и те же вычисления, и у вас не было бы возможности использовать полученные результаты для решения дальнейших задач. Иными словами, вы не могли бы выполнять никакие операции, требующие предварительного «запоминания» данных.

Все перечисленное (а также многое другое) становится возможным благодаря переменным. Их применение в некотором смысле самоочевидно. Вам необходимо всего лишь создать переменную с уникальным именем, позволяющим отличить ее от других переменных и элементов самого языка ActionScript, а затем присвоить ей значение. Простой пример присваивания переменной значения 1 приведен ниже:

```
myVariable = 1;
```

При задании имени переменной следует придерживаться нескольких простых правил. Имя должно состоять из одного слова и может включать в себя только буквы и цифры, а также знак доллара ($) или символ подчеркивания (_). Оно не может начинаться с цифры и не должно совпадать с ключевыми или зарезервированными словами ActionScript.

Чтобы обеспечить правильное использование переменных, ActionScript следит за ними и выводит предупреждение, когда вы пытаетесь совершить неприемлемые действия с переменными или как-либо еще нарушить правила их использования. К примеру, если вы попытаетесь выполнить математическую операцию над текстовыми данными, Flash выведет соответствующее предупреждение, чтобы вы могли исправить свою ошибку.

Чтобы такой контроль был возможен, необходимо «сообщить» Flash о том, какие данные будут храниться в той или иной переменной. Для этого до первого использования переменной нужно объявить ее с помощью ключевого слова `var` и задать тип хранимых в ней данных, указав его после двоеточия (:), следующего за именем переменной. Таким образом, рассмотренный выше пример присваивания переменной значения 1 следует записывать следующим образом:

```
var myVariable:Number = 1;
```

Существует набор встроенных типов данных, используемых в ActionScript. Некоторые из них представлены в табл. 2.1:

Таблица 2.1. Типы переменных

Тип данных	Пример	Описание
`Number`	`4.5`	Любое числовое значение, в том числе с плавающей точкой (десятичная дробь)
`Int`	`-5`	Любое целое число
`Unint`	`1`	Целое число без знака (неотрицательное)
`String`	`"привет"`	Текстовое значение, т. е. последовательность символов
`Boolean`	`True`	Принимает одно из двух значений: true или false (истина или ложь)
`Array`	`[2, 9, 17]`	Несколько значений, хранящихся в одной переменной
`Object`	`myObject`	Структура, лежащая в основе любой сущности ActionScript, а также созданный пользователем контейнер, который можно использовать для хранения нескольких значений (подобно `Array`)

Существуют также десятки дополнительных типов данных, указывающих на класс, используемый для присваивания значения переменной. (Как было сказано в главе 1, классы можно представлять себе как внешние сценарии, которые возвращают определенные данные в ваш сценарий и при создании приложений выступают в качестве необходимых составных частей единого целого.) В следующем примере для создания клипа на этапе исполнения используется класс `MovieClip` (встроенный класс Flash):

```
var myMC:MovieClip = new MovieClip();
```

Мы не будем перечислять здесь все возможные типы данных; на страницах книги мы еще не раз обратимся к этому вопросу – и очень скоро вы привыкнете к их использованию так, будто имели с ними дело с самого рождения.

В предыдущих версиях ActionScript объявление переменной и указание ее типа были опциональными. Однако в ActionScript 3.0 эти действия являются обязательными. Поначалу такое требование может показаться нелепым и излишним, но вы быстро к нему привыкнете и по достоинству оцените сопутствующие такому подходу преимущества – эффективное предотвращение ошибок.

В дальнейшем вы узнаете о том, что переменные могут относиться как ко всей области видимости (области, которой принадлежит объект, – это может быть, к примеру, основная временная диаграмма Flash или определенный класс), так и к отдельным конструкциям кода – то есть быть локальными. Ниже мы продемонстрируем это на конкретных примерах.

Условные операторы

При выполнении сценария часто необходимо совершить выбор – выполнить одну либо другую операцию в зависимости от определенных условий. В этом случае обычно используется *условный оператор*. Говоря простым языком, выполняется проверка того, соблюдается ли некоторое условие. Если условие выполнено, проверка возвращает значение `true` – и выполняется соответствующий фрагмент кода. В противном случае не происходит ничего либо же выполняется другая операция.

if

В качестве условного оператора чаще всего используется выражение `if`. В состав этого выражения входит ключевое слово `if`, за которым следует заключенное в круглые скобки условие для проверки, а затем – фигурные скобки, в которых содержится код, выполняемый в том случае, когда проверяемое условие возвращает значение `true`. В приведенном ниже примере первые три строки задают некоторый набор фактов. С помощью выражений `if` осуществляется их проверка. (Подразумевается, что в последующих примерах в данном разделе используется этот же набор фактов.)

```
var a:Number = 1;
var b:String = "привет!";
var c:Boolean = false;

if (a == 1) {
  trace("вариант a");
}
```

Чтобы установить, является ли условие в круглых скобках истинным или ложным, часто используются *операторы сравнения* и *логические операторы*. Операторы сравнения сравнивают два значения с помощью знаков «равно» (`==`), «меньше» (`<`), «больше или равно» (`>=`) и ряда других.

Логические операторы служат для логического комбинирования выражений. К ним относятся логическое И (`&&`), логическое ИЛИ (`||`) и НЕ (`!`). С их помощью вы задаете вопросы вроде «являются ли истинными это *и* это?», «является ли истинным это *или* это?» или же «является ли это *не* истинным?».

К примеру, в следующем выражении вычисление условия даст значение `false`, поскольку истинным является только одно условие, а не **оба**. В результате на панели `Output` ничего не отобразится.

```
if (a == 1 && b == "пока!") {
  trace("варианты a и b");
}
```

Примечание

При проверке условий в данном примере используется двойной знак равенства. Так обозначается оператор сравнения, задающий вопрос: «Это равно тому?». Мы останавливаемся на этом отличии неспроста: дело в том, что случайное использование простого знака равенства вместо двойного может привести к совершенно неожиданным результатам, поскольку одиночный знак равенства играет роль оператора присваивания, задающего расположенному слева от него объекту указанное в правой части выражения значение. Так как при этом происходит присваивание, проверка всегда будет истинной.[1]

В следующем примере проверка показывает, что выражение истинно, поскольку *одно* (первое) из перечисленных условий истинно. В результате будет выведен текст «вариант a или b».

```
if (a == 1 || b == "пока!") {
  trace("вариант a или b");
}
```

Наконец, в следующем фрагменте кода проверка также показывает истинность выражения, поскольку оператор НЕ совершенно верно определяет ложность переменной `c`. (Помните, что в основе механизма `if` лежит именно проверка на истинность.)

```
if (!c) {
  trace("не вариант c");
}
```

Логический оператор НЕ может входить также как составная часть в оператор сравнения. Рядом со знаком равенства он означает «не равно». Таким образом, следующее выражение ложно, поскольку a на самом деле равно 1, – и никакого сообщения выведено не будет.

```
if (a != 1) {
  trace("a не равно 1");
}
```

Выражение `if` можно использовать более эффективно, добавив в него безусловную альтернативу. Это альтернативный фрагмент кода, в котором нет собственной проверки условий, – для его выполнения нужно лишь, чтобы проверка первоначальных условий показала их ложность. Таким образом, если в рассмотренный ранее пример добавить дополнительный фрагмент кода, как показано ниже, будет выведено сообщение, заданное в последней команде `trace`.

```
if (a != 1) {
  trace("a не равно 1");
```

[1] Не совсем точное утверждение: в частном случае проверочного условия `a == 0` ошибка (одиночный знак равенства вместо двойного) превратит условие в выражение присваивания `a = 0`, которое будет преобразовано в `false`, а не в `true`. – *Примеч. науч. ред.*

```
} else {
  trace("a равно 1");
}
```

Эту структуру можно сделать еще более гибкой, добавив в нее условную альтернативу (то есть дополнительную проверку условия). В приведенном ниже примере будет выведено сообщение, определенное во второй команде `trace`.

```
if (a == 2) {
  trace("a не равно 1");
} else if (a == 1) {
  trace("a равно 1");
}
```

Условное выражение `if` обязано включать в себя одно ключевое слово `if` и может включать одно ключевое слово `else` и произвольное количество дополнительных проверок `else if`. При этом в любом случае будет выполнена только одна ветвь всей условной структуры. Рассмотрим следующий пример, в котором теоретически возможно выполнение всех трех методов `trace`: первых двух – потому, что они истинны, а последнего – потому, что это безусловная альтернатива.

```
if (a == 1) {
  trace("вариант a");
} else if (b == "привет!") {
  trace("вариант b");
} else {
  trace("прочее");
}
```

Однако на панель `Output` в данном случае будет выведен только текст «вариант a», поскольку истинность первого условия приведет к выполнению блока в первых фигурных скобках – и на этом условный оператор `if` завершит свою работу, невзирая на то, что остальные условия тоже истинны. Чтобы независимо выполнить несколько операций, потребуется несколько условных операторов. К примеру, в следующем примере будет выполнена первая команда `trace` каждого оператора `if`.

```
if (a == 1) {
  trace("вариант a");
}
if (b == "привет!") {
  trace("вариант b");
} else {
  trace("прочее");
}
```

switch

В зависимости от стоящих перед вами задач вы можете выстраивать условное выражение `if` любой сложности. Однако длинная и запутан-