Объектно-ориентированная разработка на ActionScript 2.0

Основы ActionScript 2.0

СИМВОЛ®

O'REILLY®

Колин Мук

Essential ActionScript 2.0

Colin Moock

O'REILLY®

ActionScript 2.0
Основы

Колин Мук

Санкт-Петербург — Москва
2006

Колин Мук

ActionScript 2.0. Основы

Перевод Н. Шатохиной

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*М. Антипин*
Редактор	*В. Овчинников*
Корректор	*О. Макарова*
Верстка	*О. Макарова*

Мук К.
ActionScript 2.0. Основы. – Пер. с англ. – СПб.: Символ-Плюс, 2006. – 576 с., ил.
ISBN 5-93286-087-1

Руководство посвящено описанию ActionScript 2.0 – последней версии, реализованной во Flash MX 2004 и Flash MX Professional 2004, и адресовано тем, кто работает во Flash и делает первые шаги в программировании, а также тем, кто уже знаком с ООП по таким языкам, как Java или C++, и хочет применить свои знания во Flash.

Рассмотрены основные принципы ООП, его синтаксис и применение в ActionScript 2.0; типы данных, классы, объекты, методы, свойства, наследование, композиция, интерфейсы, пути к классам, пакеты и обработка исключений. Описаны лучшие методики создания объектно-ориентированных проектов, структурирования приложений и обмена кодом с другими разработчиками, овладев которыми, вы научитесь создавать стабильные, масштабируемые и расширяемые приложения. Показано, как применять во Flash паттерны проектирования Observer, Singleton и Model-View-Controller, а также модель делегирования событий, при этом особое внимание уделяется их реализации на ActionScript 2.0. В приложении А содержится справочник по языку (типам данных, классам, объектам, глобальным свойствам и глобальным функциям). Этот материал поможет предупредить возникновение ошибок несоответствия типов при объявлении типов данных.

ISBN 5-93286-087-1
ISBN 0-596-00652-7 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@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 10.02.2006. Формат 70x100 1/16 . Печать офсетная.
Объем 36 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Посвящается Грэю, чудо-ребенку

Оглавление

Предисловие

Летом 2000 года сразу после окончания колледжа я пришла в компанию Macromedia на должность программиста обеспечения в команду разработчиков Flash. В первые дни моего пребывания в компании команда неутомимо трудилась над выпуском Flash 5, и все были слишком заняты, чтобы загружать меня работой, – разве что помогли мне постичь особенности корпоративной жизни Macromedia. Как же мало я понимала, изучая сложную C++-архитектуру средств разработки Flash, что ActionScript тоже начинал свой путь в индустрии веб-разработки. Flash 5 стал поворотной вехой в истории развития средства разработки Flash: он превратил ActionScript из интерфейса взаимодействия типа «указал и щелкнул» в полноценный язык сценариев, базирующийся на стандарте ECMAScript, с настоящим текстовым редактором. Я приступила к работе как раз в тот момент, когда команда Flash вкладывала реальную мощь сценариев в руки Flash-разработчиков. В следующих двух версиях Flash эта работа была продолжена: сначала я принимала участие в создании отладчика ActionScript во Flash MX и затем в разработке компилятора ActionScript 2.0. Последние несколько лет для меня неразрывно связаны с этим языком, и он сыграл немаловажную роль в моем профессиональном росте, так же как и я способствовала его развитию.

Вначале мое отношение к ActionScript было таким же, как и у многих обычных разработчиков, когда они переходят на какой-то язык программирования. Мне нравилась его гибкость, но расстраивала его ограниченность. Я была счастлива воплотить в жизнь такие возможности как, например, отладчик, потому что он помог Flash реализовать мои собственные ожидания от среды программирования. Мне нравилось шаг за шагом закрывать пробелы в возможностях Flash. Во Flash MX мы достигли больших успехов, существенно улучшив редактор кода и обеспечив пользователю возможность отладки ActionScript. Однако ActionScript 1.0 имел одно неприятное ограничение, которое мы не устранили во Flash MX: возможность написать код, использующий методы ООП, существовала, но сделать это было крайне сложно, и этот код плохо сочетался с реалиями Flash, такими как библиотечные символы.

Создав Flash MX 2004 и ActionScript 2.0, мы прошли еще один основной этап эволюции ActionScript. ActionScript 2.0 предлагает более развитый синтаксис для конструкций ООП, которые ActionScript все-

гда поддерживал. ActionScript 2.0 проще в изучении, чем его предшественники, и ближе к другим промышленным языкам программирования, таким как Java и C#. Он предоставляет разработчикам инфраструктуру, необходимую для создания и обслуживания больших, сложных приложений. Кроме того, наша реализация требовала внесения минимальных изменений во Flash Player, это означает, что ActionScript 2.0 можно экспортировать во Flash Player 6, который на момент выхода Flash MX 2004 применялся уже практически повсеместно.

За короткое время существования ActionScript разработчики успели оценить его исключительную мощь. Flash практически не налагает ограничений на доступ разработчиков к иерархии и объектной модели *MovieClip*, позволяя им делать все, что угодно, везде, где угодно. Эта гибкость побудила наших пользователей к творчеству, позволив им освоить ActionScript и экспериментировать с ним. Однако отсутствие структуры в ActionScript 1.0 усложняет масштабируемость приложений, в результате чего проекты становятся громоздкими и командам трудно обслуживать и организовывать их. Можно было запросто написать плохой код, не говоря уже о том, чтобы разместить код там, где его практически невозможно было найти, если ты плохо знаком с проектом. Создатели ActionScript 2.0 постарались учесть эти просчеты, поддерживая понятную структуру, которой могут придерживаться все разработчики. Более того, компилятор ActionScript 2.0 обеспечивает разработчикам обратную связь по ошибкам, которые в противном случае не были бы обнаружены вплоть до времени выполнения. ActionScript по-прежнему предоставляет широкий и уникальный контроль над графическими элементами. Мы стремимся доказать, что ActionScript является мощным развивающимся языком, не задевая интересов бывалых пользователей языков сценариев.

ActionScript 2.0 также был основой для некоторых замечательных элементов Flash MX 2004.

Все нижеприведенные элементы написаны на ActionScript 2.0:

- Второе поколение компонентов (т. е. компоненты v2)
- Новый инструмент Screens (Экраны), включающий Slides (Слайды) и Forms (Формы) (доступные только во Flash MX Professional 2004)
- Усовершенствованные возможности интеграции данных
- Поддержка многоязыковых ресурсов, предлагаемая панелью Strings (Строки)

Создание важных крупномасштабных функциональных возможностей с помощью ActionScript 2.0 предоставило ценную тестовую и контрольную информацию при работе над компилятором и подсказало многие конструкторские решения. Что еще важнее, эти функциональные возможности обеспечили разработчикам Flash полные рабочие примеры ActionScript 2.0 в действии (см. папку *Macromedia\Flash MX 2004\en\First Run\Classes*). Похожим образом преимущества Ac-

tionScript 2.0 отчетливо проявляются в этих функциях, состоящих из классов, организованных в иерархию класса *mx.**. Кроме того, определить принадлежность кода к компоненту теперь стало просто как никогда, поскольку ActionScript 2.0 сделал возможным устранять ненадежные пережитки прошлого ActionScript, такие как указание транслятору `#initclip` (директива компилятора).

ActionScript начинался с нескольких команд сценариев, вставляемых по щелчку мыши. Пять лет спустя он превратился в полнофункциональный объектно-ориентированный язык, позволяющий создавать большие и сложные приложения. Его синтаксис прост, строен, легок для восприятия и понятен даже новичку. Принимая участие в создании двух версий средств разработки Flash, я шаг за шагом все глубже и глубже узнавала ActionScript и теперь горжусь тем, что в его успехе есть и моя заслуга. Предыдущая книга Колина Мука, «ActionScript for Flash MX: The Definitive Guide», была очень ценной для меня, хотя я уже работала над новой версией ActionScript. Это единственная книга, которую вы найдете на столе каждого специалиста команды разработки Flash. Выхода новой книги, «Essential ActionScript2.0», уже с нетерпением ожидают многие наши сотрудники. И не зря. В ней Мук в который раз продемонстрировал свой глубокий и доступный стиль при рассмотрении сложных тем, излагая не только синтаксис ActionScript 2.0, но также теорию и принципы ООП. Он всесторонне исследовал взаимоотношения между ActionScript 2.0, его предшественниками и другими языками и подробно проиллюстрировал их отличия. Он досконально знает Flash и ActionScript, и это очевидно. Данная книга без сомнения необходима всем, кто желает освоить язык ActionScript 2.0.

– Ребекка Сан (Rebecca Sun),
ведущий разработчик программного обеспечения,
команда разработки Flash компании Macromedia,
март 2004

Введение

В сентябре 2003 года компания Macromedia выпустила в свет Flash MX 2004 и вместе с ним ActionScript 2.0 – существенно расширенную версию языка программирования во Flash.

Для создания Flash-приложений ActionScript 2.0 предлагает формальный синтаксис и методологию *объектно-ориентированного программирования* (ООП). По сравнению с традиционными методиками разработки на базе временной диаграммы основанные на ООП методы разработки ActionScript 2.0 обычно делают приложения:

- Более дружественными к разработчику
- Более стабильными и свободными от ошибок
- Более удобными для повторного использования в проектах
- Более удобными для поддержки, внесения изменений и расширения
- Более пригодными для тестирования
- Более пригодными для совместной разработки двумя или более разработчиками

Это просто исключительные качества. Настолько исключительные, что они превращают эту книгу в своего рода библию. Она призвана привлечь вас в ряды ярых сторонников ActionScript 2.0.

Книга зовет

Она призывает вас избрать ООП инструментом своей ежедневной работы с Flash. Она призывает вас собрать урожай преимуществ ООП – одной из самых важных революций в истории программирования. Она призывает вас постичь ActionScript 2.0 до самых глубин. И она ни перед чем не остановится, чтобы ее призывы достигли цели.

Вот ее план...

В главах части I «Язык ActionScript 2.0» представлены основные идеи, синтаксис и применение ООП. Даже если вы никогда ранее не сталкивались с объектно-ориентированным программированием, прочитав часть I, вы поймете, что это такое и как его использовать. В главе 1 представлен краткий обзор ActionScript 2.0. Глава 2 излагает основы ООП и помогает вам принять решение о том, что подходит для вашего проекта. В главах 3–10 подробно рассмотрены классы, объекты, мето-

ды, свойства, наследование, композиция, интерфейсы, пакеты и огромное количество других понятий ООП. Тем, кто уже знаком с ООП, например, работали с Java или другими объектно-ориентированными языками, эта книга поможет упорядочить имеющийся опыт. Здесь ООП во Flash сравнивается с тем, что уже вам известно. По ходу дела книга вводит ООП в обычную для вас практику через упражнения, демонстрирующие реальное применение ООП во Flash.

В части II «Разработка приложений» вы научитесь создавать целые приложения с помощью ActionScript 2.0. В главе 11 приводятся лучшие примеры планирования и разработки объектно-ориентированного проекта. Из глав 12–13 вы узнаете, как компоненты пользовательского интерфейса и клипы вписываются в хорошо структурированное приложение Flash. В главе 14 показано, как распределить совместно используемый код между разработчиками. Все это поможет создавать более масштабируемые, гибкие, стабильные приложения. И все это входит в план книги.

И наконец, часть III «Примеры паттернов проектирования на ActionScript 2.0» посвящена исследованию различных подходов к разным ситуациям при программировании. Вы увидите, как применять испытанные и широко распространенные стратегии ООП – известные как *паттерны проектирования* (*design patterns*) – во Flash. Паттерны проектирования в части III охватывают два ключевых момента Flash-разработки: передачу событий и управление пользовательским интерфейсом. После введения в шаблоны проектирования в главе 15 в главах 16–19 мы рассмотрим четыре основных паттерна. Поработав с паттернами, представленными в части III, вы поверите в свои силы и перейдете к рассмотрению разнообразных паттернов, представленных в сети или в другой литературе. И получите навык для перехода к другим общепринятым технологиям ООП. Как видите, книга понимает, что не будет с вами вечно. Она должна научить вас находить собственные решения.

Не имеет значения, известно ли вам, что такое «класс», «наследование», «метод», «прототип» или «свойство». Если вы понятия не имеете о том, что такое ООП или почему оно так широко распространено, эта книга рада приветствовать вас. А тех, кто уже имеет опыт ООП-разработок, эта книга хочет сделать лучше. Она хочет снабдить вас исчерпывающим справочным материалом и примерами, необходимыми для максимального повышения производительности в ActionScript 2.0.

Эта книга твердо стремится сделать вас опытным объектно-ориентированным разработчиком. И я уверен, что эта задача будет выполнена.

Чего нет в этой книге

Данная книга посвящена основам ActionScript 2.0 и ООП и поэтому не охватывает все темы, которые могут быть связаны с ActionScript. В частности, здесь нет пространных описаний сопутствующих технологий,

таких как Flash Remoting или Flash Communication Server, как нет здесь и похожего на словарь справочника по языку, какой был в «ActionScript for Flash MX: The Definitive Guide» (O'Reilly).[1] Здесь описываются «родные» функции, свойства, классы и объекты Flash Player, поэтому, прочитав эту книгу, вы научитесь применять эти классы и объекты и встраивать их в свои собственные структуры. Встроенная библиотека классов Flash Player во Flash Player 7 была лишь дополнена, поэтому «ActionScript for Flash MX: The Definitive Guide» остается справочником, актуальным даже для разработчиков, программирующих на ActionScript 2.0. Она представляет собой замечательное дополнение к данной книге.

Здесь не рассматривается инструмент Screens (включая Slides и Forms), поддерживаемый только во Flash MX Professional 2004 и применяемый для визуальной разработки пользовательских интерфейсов (в традициях MS Visual Basic) и для создания слайдовых презентаций (в традициях MS PowerPoint). Несмотря на это, постигнув основы, изложенные в этой книге, вы, без сомнения, будете готовы рассмотреть Screens самостоятельно.

Эта книга также не является руководством по основам программирования, таким как условные операторы (операторы *if*), циклы, переменные, массивы и функции. Основы программирования во Flash рассмотрены в книге «ActionScript for Flash MX: The Definitive Guide».

И наконец, эта книга не обучает использованию среды разработки Flash, за исключением того, что касается разработки приложений на ActionScript 2.0. Для получения справки по среде разработки, например о создании графических объектов или анимации, стоит обратиться к документации или любой хорошей книге, например книге Роберта Хокмана (Robert Hoekman) «Flash Out of the Box», O'Reilly, 2004.

Кому следует и кому не следует читать эту книгу

Вам следует прочитать эту книгу, если вы:

- Программируете на ActionScript 1.0 или JavaScript лучше, чем новичок, но не достигли профессионального уровня, знаете, что такое переменные, циклы, условные операторы, функции, массивы и другие основные понятия программирования.
- Опытный программист на ActionScript 1.0 или ActionScript 2.0, желающий получить достоверную информацию о лучших практических методах ООП в ActionScript 2.0, включая подробный синтаксис и информацию о вариантах использования, индивидуальные особенности языка и примеры структур приложений.

[1] К. Мук «ActionScript для Flash MX. Подробное руководство». – Пер. с англ. – СПб.: Символ-Плюс, 2004.

- Flash-дизайнер, немного занимающийся программированием и интересующийся разработкой приложений.
- Программист, переходящий к разработке в среде Flash с другого языка программирования, например Java, C++, Perl, JavaScript или ASP. (Будьте готовы изучать среду разработки Flash по другим упомянутым ранее источникам. Вам также следует прочитать главу 13 «Клипы» и книгу «ActionScript for Flash MX: The Definitive Guide», которую можно найти в сети по адресу *http://moock.org/asdg/samples*.)

Вам не следует читать эту книгу, если вы являетесь дизайнером/мультипликатором во Flash и имеете небольшой или вообще не имеете опыта программирования. (Лучше начните изучение ActionScript с книги «ActionScript for Flash MX: The Definitive Guide».)

ActionScript 2.0 и ActionScript 1.0

Более подробно ActionScript 2.0 представлен в главе 1, здесь я лишь кратко сориентирую разработчиков, пишущих на ActionScript 1.0.

ActionScript 1.0 и ActionScript 2.0 имеют аналогичный основной синтаксис. Такие основные понятия, как условные инструкции, циклы, операторы и другие не объектно-ориентированные аспекты ActionScript 1.0, могут применяться в ActionScript 2.0 и по-прежнему остаются официальной частью языка. Кроме того, создание объектов, доступ к свойствам и вызов методов в ActionScript 1.0 и ActionScript 2.0 имеют аналогичный синтаксис. Таким образом, вообще говоря, ActionScript 2.0 привычен для разработчиков на ActionScript 1.0. Основное отличие между двумя версиями этого языка программирования заключается в объектно-ориентированном синтаксисе и поддержке средств объектно-ориентированной разработки.

Объектно-ориентированный синтаксис в ActionScript 1.0 непрозрачен и практически не имеет поддержки со стороны средств разработки (например, нет сообщений компилятора, нет структуры файла класса, нет контроля типов, слаба связь между кодом и ресурсами фильма и т. д.). В ActionScript 1.0 объектно-ориентированное программирование было трудным и малопонятным занятием, а в ActionScript 2.0 это дело совершенно естественное. В ActionScript 2.0 средства ООП реализованы в более традиционном ключе, что позволяет применять в нем навыки, полученные при работе с другими объектно-ориентированными языками, и наоборот, переносить навыки работы с ActionScript 2.0 в другие языки.

Те, кто программирует на ActionScript 1.0 и уже применял методы ООП, получат удовольствие от работы с ActionScript 2.0. Тем же, кто работал с ActionScript 1.0 и не применял ООП, не придется учить ООП в ActionScript 1.0, перед тем как обучаться ему в ActionScript 2.0. Вам представилась идеальная возможность изучить и освоить эту важную методику. ООП позволяет повысить производительность, упростить

управление проектами и улучшить качество кода и возможность его повторного использования.

В этой книге не уделяется много времени вопросу обновления кода при переходе с ActionScript 1.0 на ActionScript 2.0, но после ее прочтения у вас не должно быть проблем с этим. Главная задача состоит в том, чтобы сформировать у читателя твердое фундаментальное понимание ActionScript 2.0, и я не хотел слишком отклоняться от этого курса, рассуждая об устаревшем коде ActionScript 1.0. Это означает, что вам придется следить за многочисленными примечаниями, которые выглядят вот так:

В таких примечаниях проводится прямое сравнение методов ActionScript 1.0 с аналогичными методами ActionScript 2.0, так что вы можете увидеть различия между старым и новым усовершенствованным вариантом реализации той или иной задачи.

Наконец, давайте определимся с тем, что я имею в виду под «сравнением программирования на ActionScript 2.0 и ActionScript 1.0». Если вы просто создаете код во временной диаграмме и не используете классы ActionScript 2.0, статические типы данных или другие возможности ООП, то действительно, довольно сложно понять, на чем он написан – на «ActionScript 2.0» или на «ActionScript 1.0». Без особенностей ООП код ActionScript 2.0 совершенно неотличим от кода ActionScript 1.0. Поэтому, когда я говорю «мы научимся программировать на ActionScript 2.0», то я, без сомнения, подразумеваю, что вы займетесь созданием содержательного ООП-приложения, в котором разработаете один или более классов. Например, рассмотрим интерактивную форму, которая просто отправляет сообщения электронной почты. Ее можно было бы полностью реализовать во временной диаграмме Flash, применяя лишь переменные и функции. Если это все, что вы хотите делать в своих приложениях, тогда, откровенно говоря, эта книга вам не нужна. Но она (книга) способна расширить ваш кругозор и превратить вас в высококвалифицированного объектно-ориентированного разработчика, которому под силу справиться и с намного большими проектами. Итак, когда я говорю «программирование на ActionScript 2.0», я имею в виду разработку объектно-ориентированных приложений на ActionScript 2.0. При этом ударение делается на ООП, а не на ActionScript 2.0, поскольку, по существу, ActionScript 2.0 – лишь средство. Вы можете задать вопрос: «Так эта книга о синтаксисе ActionScript 2.0, объектно-ориентированной методологии или объектно-ориентированном программировании?» Ответ: «Обо всем сразу».

Об особенностях ActionScript 2.0 и ActionScript 1.0 применительно к Flash Player 6 и Flash Player 7 рассказано в главе 1.

Расшифровка версий Flash

Представляя семейство продуктов Studio MX, включающее Flash MX, компания Macromedia отказалась от стандартной системы нумерации

версий для своей среды разработки Flash. В названия продуктов, выходящих после Flash MX, Macromedia включала год их выпуска (продукты, выходящие после сентября, маркировались следующим годом). В 2004 году Macromedia разделила Flash на две версии: Flash MX 2004 и Flash MX Professional 2004, как показано в табл. П.1. Для профессиональной версии характерны следующие основные особенности:

- Инструмент Screens (разработка содержимого на базе форм и слайдов)
- Дополнительные средства создания и редактирования видеоинформации
- Средства управления проектами и ресурсами
- Внешний редактор сценариев
- Привязка данных (связывание компонентов с источниками данных, поставляемыми посредством веб-сервисов, XML или наборов записей)
- Усовершенствованные компоненты (однако компоненты Flash MX Professional 2004 замечательно работают во Flash MX 2004)
- Средства разработки для мобильных устройств

Методики, излагаемые в этой книге, могут применяться как во Flash MX 2004, так и во Flash MX Professional 2004, хотя я всегда отмечаю те редкие случаи, когда между двумя версиями есть разница с точки зрения разработки на ActionScript 2.0. В отличие от среды разработки Flash, версии Flash Player по-прежнему нумеруются, и на момент выхода книги последней является Flash Player 7. В табл. П.1 приведены названия версий Flash, применяемые в этой книге.

Таблица П.1. Названия версий Flash, применяемые в этой книге

Название	Значение
Flash MX	Версия среды разработки Flash, выпущенная одновременно с Flash Player 6.
Flash MX 2004	Стандартная версия среды разработки Flash, выпущенная одновременно с Flash Player 7. В общем смысле выражение «Flash MX 2004» применяется и к стандартному изданию (Flash MX 2004), и к версии для профессионалов (Flash MX Professional 2004) этого программного обеспечения. При обсуждении особенностей, характерных для профессиональной версии, эти ограничения в данной книге обозначены явно.
Flash MX Professional 2004	Версия среды разработки Flash для профессионалов, выпущенная одновременно с Flash Player 7. Профессиональное издание включает некоторые возможности, отсутствующие в стандартной версии (см. предыдущую графу таблицы). Для работы с этой книгой и использования ActionScript 2.0 профессиональная версия не является обязательной.

Таблица П.1 (продолжение)

Название	Значение
Flash Player 7	Flash Player, версия 7. Flash Player – это плагин (подключаемый модуль) для основных веб-броузеров Windows и Macintosh. На момент выхода данной книги для Linux был доступен Flash Player 6, а не Flash Player 7. Плагин существует в двух версиях, как элемент управления ActiveX и как версия для Netscape, но я обобщенно называю их «Flash Player 7».
Flash Player *x*.0.*y*.0	Flash Player, в частности версия, определяемая главным номером версии *x* и номером основной сборки *y*, как для Flash Player 7.0.19.0. Второй номер версии и второй номер сборки выпускаемых версий всегда 0.
Standalone Player (автономный проигрыватель)	Версия Flash Player, которая запускается непосредственно как исполняемый файл локальной системы, а не как плагин веб-броузера или элемент управления ActiveX.
Projector (проектор)	Самостоятельный исполняемый файл, который включает и файл *.swf*, и Standalone Player. Проекторы могут создаваться как для операционной системы Macintosh, так и для Windows, с помощью функции File→Publish меню Flash.

Файлы примеров и ресурсы

Официальный веб-сайт этой книги:

http://moock.org/eas2

Файлы примеров для этой книги можно скачать по адресу:

http://moock.org/eas2/examples

Примеры Flash-кода также можно найти в сетевом хранилище кода (Code Depot) книги «ActionScript for Flash MX: The Definitive Guide»:

http://moock.org/asdg/codedepot

Обширный список сетевых ресурсов, касающихся Flash, приведен по адресу:

http://moock.org/moockmarks

Большая коллекция ссылок на сотни ресурсов по ActionScript 2.0:

http://www.actionscripthero.com/adventures

Принятые обозначения

Для представления различных синтаксических компонентов ActionScript в этой книге приняты следующие обозначения:

Пункты меню

Переходы по пунктам меню отображаются с помощью символа →, например File→Open.

`Моноширинный`

Представляет фрагменты кода, имена экземпляров клипа, метки кадров, имена свойств, имена переменных и идентификаторы связывания символов.

Курсив

Представляет имена функций, методов, классов, пакетов, слоев, URL, имена файлов и расширения файлов, например *.swf*. Кроме курсивного начертания, имена методов и функций в тексте сопровождаются круглыми скобками, например *duplicateMovieClip( )*.

`Моноширинный полужирный`

Представляет текст, который при выполнении пошаговой процедуры необходимо ввести дословно. Кроме того, в примерах кода **`Моноширинный полужирный шрифт`** служит для визуального выделения, например подчеркивает важную строку кода в большом примере.

`Моноширинный курсив`

Указывает на код, который необходимо заменить соответствующим значением (например, *здесь расположено ваше имя*). *`Моноширинное курсивное`* начертание также служит для визуального выделения имен переменных, свойств, методов и функций, помещаемых в комментарии в примерах кода.

Это подсказка. Она содержит полезную информацию по рассматриваемой теме, зачастую обращает внимание на важные идеи или лучшие практические методы.

Это предупреждение. Помогает избежать досадных ошибок, обойти трудности или предупреждает о надвигающейся опасности. Внять ему или игнорировать его – ваше дело.

Это примечание об ActionScript 1.0. Здесь сравниваются и противопоставляются ActionScript 1.0 и ActionScript 2.0, что помогает перейти к ActionScript 2.0 и понять важные различия между этими двумя версиями языка программирования.

Примеры кода

Эта книга призвана помочь вам в работе. В общем код, приведенный в данной книге, можно помещать в программы и документацию. Не надо запрашивать у нас разрешения, если только вы не воспроизводите значительные части кода. Например, если в вашей программе присутствует несколько фрагментов кода из этой книги, то разрешение не требуется. Если вы будете отвечать на вопросы, цитируя данную книгу и ссылаясь на примеры кода, разрешение не требуется. А вот для включения существенного количества примеров кода из этой книги в документацию своего продукта разрешение необходимо.

Мы признательны за указание авторства, что обычно включает название, автора, издателя и ISBN, например «Essential ActionScript 2.0 by Colin Moock. Copyright 2004 O'Reilly Media, Inc., 0-596-00652-7», но не требуем этого.

Если, используя примеры кода, вы чувствуете, что выходите за рамки законности или данных выше разрешений, не стесняйтесь обращаться к нам по адресу *permissions@oreilly.com*.

Мы хотели бы знать ваше мнение

Мы тщательно проверили информацию, излагаемую в данной книге, но что-то могло измениться (могли даже вкрасться какие-то ошибки!). Пожалуйста, сообщайте нам о любых обнаруженных ошибках, а также присылайте свои предложения для будущих изданий по адресу:

O'Reilly Media, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (в США или Канаде)

(707) 829-0515 (международный/местный)

(707) 829-0104 (fax)

Для этой книги мы создали веб-страницу, на которой приводим список опечаток, примеры или любую дополнительную информацию. Вы можете найти эту страницу по адресу:

http://www.oreilly.com/catalog/0596006527

Чтобы задать технические вопросы по этой книге или высказаться о ней, пришлите электронное письмо по адресу:

bookquestions@oreilly.com

Более подробная информация о наших книгах, конференциях, программному обеспечению, Центрах ресурсов (Resource Centers) и O'Reilly Network представлена на нашем веб-сайте:

http://www.oreilly.com

Благодарности

Иногда вам предоставляется возможность поблагодарить, но вы осознаете, что не можете выразить глубину своей признательности в полной мере. Вы можете высказать все, что хотите, но в конце концов должны просто поверить, что человек знает, насколько велика ваша благодарность. Я верю, что Ребекка Сан, ведущий разработчик ActionScript 2.0 компании Macromedia, знает это.

Мой ближайший партнер – Дерек Клейтон (Derek Clayton). В течение многих лет мы работали с Дереком над Unity, нашей коммерческой

инфраструктурой для создания многопользовательских приложений (смотрите *http://moock.org/unity*). Дерек был моим наставником с тех пор, как я впервые столкнулся с оператором *if*, а дружим мы еще дольше. Я учусь у него чему-то новому практически каждый день. Эта книга наполнена мудростью, которой он делился со мной все эти годы.

Брюс Эпштейн (Bruce Epstein), мой редактор. Ну что можно сказать? Просто он – лучший. Ни одна гипербола не может преувеличить ни его достоинств, ни справедливости, поэтому я промолчу.

Также я хотел бы поблагодарить всех редакторов, отдел подготовки к печати, дизайнеров, художников, сотрудников служб продаж и маркетинга издательства O'Reilly, включая Гленн Бисигнани (Glenn Bisignani), Клер Клотье (Claire Cloutier), Колина Гормана (Colleen Gorman), Тима О'Рейли (Tim O'Reilly), Роба Романо (Rob Romano), Сару Шерман (Sarah Sherman), Эллен Траутман (Ellen Troutman) и Элли Волькхаузен (Ellie Volckhausen). Также мои благодарности редактору Норме Эмори (Norma Emory) за помощь в подготовке рукописи к печати.

Также я благодарен членам команды разработки Flash компании Macromedia, которые являются постоянным источником вдохновения, знаний и дружбы с момента появления «Flash». Я уверен, что все, кто интересуется компьютерами, в долгу перед всей Flash-командой за то, что они постоянно находят новые формы взаимодействия с помощью компьютера. Прежде всего по гроб жизни я обязан Гари Гроссману (Gary Grossman) за его бесконечную поддержку и доброту, ему мои бесконечные «спасибо» и длительные рукопожатия. Особенно хочется отметить членов Flash-команды, бывших и теперешних, с которыми мне выпала честь вместе работать: Найджела Пег (Nigel Pegg), Майкла Вилльямса (Michael Williams), Эрика Нортона (Erica Norton), Валида Анбара (Waleed Anbar), Денеба Мекета (Deneb Meketa), Метта Вобенсмита (Matt Wobensmith), Майка Чемберса (Mike Chambers), Криса Силджена (Chris Thilgen), Джиллиса Дрю (Gilles Drieu), Найвиша Раджбхандари (Nivesh Rajbhandari), Тею Ота (Tei Ota), Троя Эванса (Troy Evans), Лусьена Бииба (Lucian Beebe), Джона Дауделла (John Dowdell), Бентли Вулфа (Bentley Wolfe), Джеффа Мотт (Jeff Mott), Тайника Уро (Tinic Uro), Роберта Татсуми (Robert Tatsumi), Майкла Ричардса (Michael Richards), Шерон Селдон (Sharon Seldon), Джоди Женг (Jody Zhang), Джима Корбетта (Jim Corbett), Карен Кук (Karen Cook), Джонатана Гей (Jonathan Gay), Пита Сантанджели (Pete Santangeli), Шона Кранзберга (Sean Kranzberg), Майкла Морриса (Michael Morris), Кевина Линча (Kevin Lynch), Бена Чан (Ben Chun), Эрика Виттмана (Eric Wittman), Джереми Кларка (Jeremy Clark) и Джениса Пирса (Janice Pearce).

Мне исключительно повезло иметь действительно замечательных технических редакторов и корректоров при работе над этой книгой. Мудрый взгляд Ребекки Сан охватил весь текст. Гари Гроссман просмотрел ключевые разделы, включая главу 10. Эти проницательные читатели

были моими проводниками при написании книги: Алистар Мак-Лоид (Alistair McLoed), Чафик Казоун (Chafic Kazoun), Джон Виллиамс (Jon Williams), Маркус Дикинсон (Marcus Dickinson), Оуэн Ван Дайджк (Owen Van Dijk), Питер Холл (Peter Hall), Ральф Бокелберг (Ralf Bokelberg), Роберт Пеннер (Robert Penner) и Сэм Нефф (Sam Neff). Особая благодарность Марку Джонкману (Mark Jonkman) и Нику Ридеру (Nick Reader) за их постоянную кропотливую работу по проверке рукописи.

Слова любви моей жене Венди (Wendy), которая поддерживает меня. Моей семье и друзьям. И деревьям за ответы на все вопросы, величие любой мечты и за бумагу, на которой эта книга напечатана.

Колин Мук (Colin Moock)
Торонто, Канада
март 2004

I

Язык программирования ActionScript 2.0

В части I вы познакомитесь с основами объектно-ориентированных понятий, синтаксиса и применения ActionScript 2.0. Даже если вы никогда ранее не сталкивались с ООП, эта часть книги обеспечит вам его понимание. Здесь рассматриваются классы, объекты, методы, свойства, наследование, формирование, интерфейсы, пакеты и масса других основных понятий ООП. Материал этой части книги также поможет вам понять, насколько ООП подходит для ваших проектов и как лучше структурировать классы и их методы.

- Глава 1 «Краткий обзор ActionScript 2.0»
- Глава 2 «Объектно-ориентированный ActionScript»
- Глава 3 «Типы данных и контроль типов»
- Глава 4 «Классы»
- Глава 5 «Создание класса на ActionScript 2.0»
- Глава 6 «Наследование»
- Глава 7 «Создание подкласса в ActionScript 2.0»
- Глава 8 «Интерфейсы»
- Глава 9 «Пакеты»
- Глава 10 «Исключения»

1

Краткий обзор ActionScript 2.0

В данной книге мы изучим ActionScript 2.0 и объектно-ориентированное программирование во Flash. Материал объемный, но прежде чем углубиться в детали, рассмотрим кратко основные свойства ActionScript 2.0 и новые возможности Flash Player 7. Тем, у кого есть базовые знания по ActionScript 1.0, этот обзор обеспечит общее представление об изменениях в языке. Те же, кто еще не знаком с Flash или ActionScript, возможно, захотят сразу перейти к главе 2.

Возможности ActionScript 2.0

Реализованный во Flash MX 2004 и Flash MX Professional 2004 язык программирования ActionScript 2.0 представляет собой основательно переработанный ActionScript из Flash 5 и Flash MX (задним числом названного *ActionScript 1.0*). В ActionScript 2.0 добавлено относительно немного новых функциональных возможностей, но в нем радикально улучшена объектно-ориентированная разработка в среде Flash за счет формализации объектно-ориентированного синтаксиса и методологии.

Хотя ActionScript 1.0 мог бы использоваться как объектно-ориентированный язык, ему не хватает традиционной формальной лексики для создания классов и объектов. В ActionScript 2.0 введена дополнительная синтаксическая поддержка традиционных объектно-ориентированных возможностей. Например, для создания классов в ActionScript 2.0 есть ключевое слово *class*, а для реализации наследования – ключевое слово *extends*. Их не было в ActionScript 1.0 (но в нем можно создавать прототипы объектов, которые могли бы использоваться как классы). Традиционный объектно-ориентированный синтаксис ActionScript 2.0 делает этот язык привычным для программистов, имеющих опыт работы с другими объектно-ориентированными языками, такими как Java и C++.

Большая часть нового ООП-синтаксиса ActionScript 2.0 базируется на предложенном стандарте ECMAScript 4. Его спецификация размещена по адресу *http://www.mozilla.org/js/language/es4*.

Вот некоторые из ключевых возможностей, введенных в ActionScript 2.0. Ничего страшного, если некоторые из них вы видите впервые; они подробно рассмотрены в этой книге:

- Ключевое слово *class*, предназначенное для создания формальных классов. Рассматривается в главе 4.
- Ключевое слово *extends*, позволяющее реализовать наследование. В ActionScript 1.0 наследование обычно осуществлялось с помощью свойства prototype, но также могло быть реализовано посредством свойства `__proto__`. Наследованию посвящена глава 6.
- Ключевое слово *interface*, посредством которого можно создавать интерфейсы в стиле Java (т. е. абстрактные типы данных). Классы обеспечивают реализации интерфейсов с помощью ключевого слова implements. ActionScript 1.0 не поддерживал интерфейсов. Интерфейсы рассматриваются в главе 8.
- Официальным расширением для файлов классов является *.as*. Раньше классы можно было определять в коде временной диаграммы или во внешних *.as*-файлах. Теперь ActionScript 2.0 требует, чтобы классы описывались во внешних файлах. Файлы классов можно редактировать в редакторе сценариев Flash MX Professional 2004 или во внешнем текстовом редакторе.
- Формальный синтаксис описания методов в теле класса, применяемый для создания методов экземпляров класса и методов класса. В ActionScript 1.0 методы добавлялись в класс через свойство `prototype` конструктора класса. См. главу 4.
- Формальный синтаксис методов, возвращающих значение переменной и устанавливающих значение переменной (getter/setter), который заменяет метод *Object.addProperty( )* ActionScript 1.0. См. главу 4.
- Формальный синтаксис описания свойств в теле класса, служащий для создания свойств экземпляров класса и свойств класса. В ActionScript 1.0 свойства экземпляров можно было добавлять несколькими способами: с помощью свойства `prototype` конструктора класса, в функции-конструкторе или создавать непосредственно в каждом объекте. Более того, в ActionScript 1.0 свойства класса определялись непосредственно в функции-конструкторе класса. Смотрите главу 4.
- Ключевые слова *private* и *public*, применяемые для защиты определенных методов и свойств класса от доступа извне.
- Статический контроль типов переменных, свойств, параметров и возвращаемых значений, применяемый для объявления типа данных

каждого из этих элементов. Это предупреждает случайные ошибки, обусловленные применением в той или иной ситуации неверного типа данных. Об ошибках несоответствия типов подробно рассказывается в главе 3.

- Приведение типов, применяемое для указания компилятору интерпретировать объект так, как если бы он был экземпляром другого типа, что иногда требуется в случае статического контроля типов. Подробно приведение типов рассматривается в главе 3.
- Пути классов, применяемые для определения местоположения одного или более хранилищ классов. Они обеспечивают возможность многократно использовать классы в разных проектах и упрощают работу с исходными файлами. См. главу 9.
- Обработка исключений, в том числе операторы *throw* и *try/catch/finally*, служащие для генерирования и обработки ошибок программы. См. главу 10.
- Простое связывание символов в библиотеке и классов ActionScript 2.0 посредством свойства символа Linkage. Это упрощает реализацию наследования *MovieClip* по сравнению с ActionScript 1.0, где требовалось применение директивы `#initclip` и метода *Object.registerClass( )*. См. главу 13.

Новые возможности Flash Player 7

Кроме расширения языка ActionScript 2.0, Flash Player 7 представляет несколько важных новых классов и возможностей. Они доступны только для фильмов в формате Flash Player 7, воспроизводимых во Flash Player 7 или более поздних версиях. (Информация по экспортированию форматов приведена в разделе «Задание версии ActionScript и версии проигрывателя фильма» этой главы.) Хотя эти возможности не являются непосредственным предметом рассмотрения данной книги, мы коснемся некоторых из них во время изучения ActionScript 2.0.

К новым ключевым возможностям Flash Player 7 относятся:

- Новые возможности сортировки массивов
- Классы *ContextMenu* и *ContextMenuItem* для настройки контекстного меню Flash Player, появляющегося по правому щелчку мыши (Windows) или по Ctrl-щелчку мыши (Macintosh) на фильме Flash
- Файлы с описанием междоменной политики безопасности для обеспечения возможности загрузки данных и содержимого из внешнего домена
- Поддержка тегов формата ID3 v2 для загружаемых MP3-файлов
- Поддержка скроллинга в текстовых полях (только для Windows)
- Улучшенные методы управления глубиной вложения *MovieClip*
- Класс *MovieClipLoader* для загрузки клипов и изображений

- Класс *PrintJob* для улучшения контроля печати по сравнению с тем, что было раньше
- Поддержка изображений в текстовых полях, включая текст, обтекающий изображения
- Улучшенные метрики текста (более точные измерения текста в текстовом поле по сравнению с Flash Player 6)
- Поддержка каскадных таблиц стилей (Cascading stylesheet – CSS) для текстовых полей, что позволяет форматировать текст фильма с помощью стандартной таблицы стилей CSS
- Улучшенная производительность ActionScript на этапе исполнения
- Строгая чувствительность к регистру

Предмет рассмотрения данной книги – базовый язык программирования ActionScript 2.0. Таким образом, не все из приведенных выше возможностей Flash Player рассматриваются здесь подробно. Более детальную информацию о новых возможностях Flash Player 7 можно получить во встроенной справочной системе Flash (Help (Справка) → ActionScript Reference Guide (Справочное руководство по ActionScript) → What's New in Flash MX 2004 ActionScript (Что нового в ActionScript Flash MX 2004)).

Компоненты версии 2 во Flash MX 2004

Во Flash MX были введены *компоненты* (*components*) – готовые к использованию специальные графические элементы интерфейса и модули кода, реализующие обычную функциональность. Встроенные компоненты Flash обеспечивают относительную легкость создания Flash-приложений. Flash MX 2004 представляет новые *компоненты v2*, созданные в ActionScript 2.0 с нуля и встроенные в версию 2 Архитектуры компонентов Macromedia, которая предоставляет намного более богатый набор возможностей, чем ее предшественник. Новая архитектура требует перехода на новые пути разработки и использования компонентов (см. главу 12). Официально компоненты v2 требуют версии Flash Player 6.0.79.0 или выше, однако тестирование показало, что многие компоненты v2 работают и в более ранних версиях Flash Player 6 (особенно Flash Player 6.0.40.0 и выше). Если предполагается, что компонент v2 будет работать в более ранней, чем Flash Player 6.0.79.0, версии, то необходимо тщательно протестировать приложение.

В одном приложении, созданном во Flash MX 2004 или Flash MX Professional 2004, могут присутствовать и компоненты v2, и компоненты v1 Flash MX при условии, что v1-компоненты будут обновлены для поддержки ActionScript 2.0 и фильм экспортируется в формат ActionScript 2.0.

Не путайте компоненты v1 и v2 с версией ActionScript, на которой они написаны. Да, компоненты v2 написаны на ActionScript 2.0 и Action-

Script 1.0-версий компонентов v2 не существует. Однако, хотя компоненты v1 изначально написаны на ActionScript 1.0, существуют их обновленные версии для компиляции под ActionScript 2.0.

Обновленные версии компонентов v1 для ActionScript 2.0 доступны в разделе Flash Exchange (Обмен Flash) (*http://www.macromedia.com/exchange/flash*) в категории User Interface (Пользовательский интерфейс) под заголовком «Flash MX Components for Flash MX 2004 (Компоненты Flash MX для Flash MX 2004)».

Если в одном фильме одновременно присутствуют компоненты v1 и v2, то могут возникнуть некоторые ошибки компиляции и времени выполнения в зависимости от применяемых компонентов.

Не смешивайте ООП-методы ActionScript 1.0 с кодом ActionScript 2.0. Если собираетесь применять классы, наследование и другие возможности ООП, убедитесь, что ваш код обновлен до ActionScript 2.0.

Новые основные возможности компонентов v2 включают:

- Новую модель слушателей событий для обработки событий компонентов, которая позволяет многим внешним объектам получать сообщения о событиях одного компонента.
- Поддержку таблиц стилей CSS, что упрощает изменение атрибутов текста всех компонентов.
- Управление фокусом для поддержки возможности перемещения по элементам пользовательского интерфейса по нажатию клавиши табуляции.
- Управление глубиной для организации визуального наложения компонентов на экране.
- Улучшенную поддержку возможностей облегчения доступа (т. е. лучшую поддержку программ чтения с экрана).
- Улучшенную поддержку тем (т. е. графической подстановки).
- Инкапсуляцию ресурсов компонентов в единый файл, что упрощает организацию и совместное использование компонентов.

Компоненты v2 обычно занимают больше места, чем их v1-аналоги. Это особенно заметно, когда используются один или два компонента, поскольку архитектура v2 оптимизирована для приложений, содержащих, по крайней мере, три или четыре типа компонентов. Поэтому если можно обойтись одним или двумя компонентами и нет необходимости в управлении фокусом или поддержке облегчения доступа, то применение компонентов v1 обеспечивает более быстрое скачивание результирующего ролика.

Помните, что применяемая по умолчанию тема («halo» – ореол), или скин, компонентов v2 не поддерживает специальных цветов для полос

прокрутки и кнопок. То есть во Flash MX 2004 и Flash MX Professional 2004 свойства стиля `scrollTrackColor` и `buttonColor` с применяемым по умолчанию скином компонентов v2 не работают. Чтобы задать цвета кнопок и полос прокрутки компонентов v2, необходимо применить к документу новую тему (новый скин). Загляните в меню Help → Using Components (Применение компонентов) → About Themes (О темах) → Applying a Theme to a Document (Применение темы к документу).

В табл. 1.1. показан полный набор компонентов Flash MX 2004 и Flash MX Professional 2004. Профессиональные компоненты, недоступные во Flash MX 2004, будут работать в этой версии среды разработки. То есть документ *.fla*, содержащий компоненты, специфичные для версии Professional, во Flash MX 2004 будет нормально открываться и работать вполне правильно. Лицензионное соглашение End User License Agreement для Flash MX 2004 явно не запрещает использования профессиональных компонентов в стандартной версии программного обеспечения.

Таблица 1.1. Компоненты v1 и v2

Компонент	Flash MX	Flash MX 2004	Flash Pro	Примечания
Accordion	[a]		v2	
Alert	[b, c]		v2	
Button	v1	v2	v2	
CheckBox	v1	v2	v2	
ComboBox	v1	v2	v2	
Компоненты для работы с данными			v2	Включают DataHolder, DataSet, RDBMSResolver, WebServiceConnector, XMLConnector и XUpdateResolver
DataGrid	[b]		v2	
DateChooser	[c, d]		v2	
DateField			v2	
Label	[a]	v2	v2	
List	v1	v2	v2	

[a] Аналогичный компонент доступен в DRK3 (*http://www.macromedia.com/software/drk/productinfo/product_overview/volume3*).

[b] Аналогичный компонент доступен в DRK1 (*http://www.macromedia.com/software/drk/productinfo/product_overview/volume1*).

[c] Аналогичный компонент доступен в наборе Flash UI Component Set 2, в разделе Flash Exchange (*http://www.macromedia.com/exchange/flash*).

[d] Аналогичный компонент доступен в DRK2 (*http://www.macromedia.com/software/drk/productinfo/product_overview/volume2*).

Компонент	Flash MX	Flash MX 2004	Flash Pro	Примечания
Label	[a]	v2	v2	
List	v1	v2	v2	
Loader	[b]	v2	v2	
Media Components	[b]		v2	MediaController, MediaDisplay, MediaPlayback
Menu	[d]		v2	
MenuBar			v2	
NumericStepper		v2	v2	
ProgressBar	[b, c]	v2	v2	
RadioButton	v1	v2	v2	
ScrollPane	v1	v2	v2	
TextArea	v1	v2	v2	
TextInput	v1	v2	v2	
Tree	[c]		v2	
Window	[c]	v2	v2	

В главе 12 мы научимся программировать графические ООП-приложения, которые используют компоненты v2.

ActionScript 1.0 и 2.0 во Flash Player 6 и 7

ActionScript 1.0 основан на стандарте ECMAScript 3 (как JavaScript 1.5), тогда как ActionScript 2.0 использует новый стандарт ECMAScript 4 (как JavaScript 2.0). Как мы узнали из «Введения» (раздел «ActionScript 2.0 и ActionScript 1.0»), это общее наследие обеспечивает двум версиям сильное семейное сходство; синтаксис большинства их возможностей, не относящихся к ООП, таких как циклы, условные ветвления и операторы, совершенно аналогичен.

Хотя ActionScript 2.0 в настоящий момент считается предпочтительной версией ActionScript, синтаксис ActionScript 1.0 по-прежнему полностью поддерживается Flash Player 7 и не переходит в разряд нерекомендуемых. Как мы вскоре убедимся, во Flash MX 2004 и Flash MX Professional 2004 можно писать как на ActionScript 1.0, так и на ActionScript 2.0 (но нельзя применять ActionScript 2.0 во Flash MX). За малыми исключениями, что отмечено в тексте, код на ActionScript 2.0 также обратно совместим с Flash Player 6. Однако ActionScript 2.0 несовместим с более старыми версиями, такими как Flash Player 5 или Flash Player 4.

Программируя на ActionScript 1.0, ActionScript 2.0 можно рассматривать как синтаксическую надстройку к ActionScript 1.0. То есть и ActionScript 2.0, и ActionScript 1.0 компилируются в один и тот же байткод *.swf* (с парой незначительных дополнений для ActionScript 2.0). Во время выполнения Flash Player не делает никакой разницы между ActionScript 1.0 и ActionScript 2.0 (кроме вышеупомянутых незначительных дополнений). Например, когда класс ActionScript 2.0, скажем *Rectangle* (*Прямоугольник*), скомпилирован в файл *.swf*, во время выполнения он существует как объект *Function*, так же как это было бы и с объявлениями функций ActionScript 1.0, выступающих в качестве конструкторов класса. Аналогично во время выполнения экземпляру класса ActionScript 2.0 *Rectangle* (`r`) предоставляется свойство `__proto__`, которое ссылается на `Rectangle.prototype`, опять же делая его похожим для Flash Player на аналог на ActionScript 1.0.

Но по большей части об этих закулисных проблемах компилятора и времени выполнения беспокоиться не надо. Перейдя на ActionScript 2.0 (а я думаю, это надо сделать!), можно полностью забыть о программировании, основанном на прототипах, бытовавшем в ActionScript 1.0. Кстати, большинство методов динамического манипулирования объектами и классами во время выполнения, применяемых в ActionScript 1.0, в ActionScript 2.0 считаются дурным тоном и в сочетании с кодом на ActionScript 2.0 будут приводить к ошибкам компилятора. Но не беспокойтесь, в данной книге особо выделены проблемы ActionScript 1.0 и показано, как решить их с помощью ActionScript 2.0.

Задание версии ActionScript и версии проигрывателя фильма

Flash MX 2004 позволяет экспортировать файлы *.swf* (также называемые фильмами, или роликами) в формат, совместимый с определенными версиями Flash Player. Не путайте это с версией Flash Player, установленной пользователем (что не подвластно контролю программиста, он может лишь проверить версию Player и предложить обновить ее, если надо).

Версию файла *.swf* можно задать в выпадающем меню File (Файл) → Publish Settings (Настройки Публикации) → Flash → Version (Версия). Для обеспечения максимальной совместимости всегда задавайте для своего файла *.swf* самую старую из допустимых для вашего проекта версий Flash Player. Если версия файла *.swf* будет выше, чем версия Flash Player конечного пользователя, существует вероятность, что он не будет воспроизводиться правильно и что большую часть кода выполнить не удастся.

В результате задания версии фильма Flash имеем следующее:

- Фильм будет совместимым (т. е. его можно будет воспроизвести) в заданной версии Flash Player (или в более поздних версиях). В преды-

дущих версиях большая часть кода ActionScript или будет выполняться неправильно, или не будет выполняться вообще.

- Фильм будет воспроизводиться надлежащим образом в самых последних версиях Flash Player, даже если в нем задействованы возможности, изменившиеся с момента выхода заданной версии. Иначе говоря, более новый Flash Player всегда сможет правильно воспроизвести файлы *.swf* старого формата. Например, идентификаторы ActionScript в файле *.swf* формата Flash Player 6 при воспроизведении во Flash Player 7 *не* чувствительны к регистру, даже несмотря на то, что идентификаторы файлов *.swf* формата Flash Player 7 *чувствительны* к регистру. Однако здесь есть одно исключение: в некоторых случаях изменения, с целью обеспечения безопасности внесенные в правила междоменной загрузки данных во Flash Player 7, оказывают влияние на файлы *.swf* формата Flash Player 6. Подробности можно найти по адресу *http://moock.org/asdg/technotes/crossDomainPolicyFiles*.

Экспортируя фильмы в формат Flash Player 6 и Flash Player 7 из Flash MX 2004 или Flash MX Professional 2004, можно дать указание Flash компилировать код как ActionScript 1.0 или как ActionScript 2.0. Конечно, этот выбор необходимо сделать в начале разработки, чтобы не переписывать код в конце. Версия компилятора ActionScript при создании файла *.swf* задается в меню File → Publish Settings → Flash → ActionScript Version.

Далее в книге предполагается, что вы работаете с компилятором ActionScript 2.0.

Если выбрана версия ActionScript 1.0, то:

- Не распознается синтаксис ActionScript 2.0, а возможности ActionScript 2.0, такие как контроль типов (включая синтаксис написания через двоеточие) и обработка ошибок, могут привести или к ошибкам компилятора (для фильмов в формате Flash Player 6), или просто дать сбой (для фильмов в формате Flash Player 7).
- Для переменных допускается «слэш-синтаксис» в стиле Flash 4 (но этого стиля написания кода рекомендуется избегать).
- Введенные в ActionScript 2.0 зарезервированные слова, такие как *class*, *interface* и *public*, могут применяться как идентификаторы (но эта практика усложняет обновление кода, и мы очень советуем отказаться от нее).

Следующие возможности ActionScript 2.0 при воспроизведении не будут работать в файлах *.swf*, экспортированных в формат Flash Player 6, независимо от версии Flash Player, установленного в системе пользователя:

- Обработка исключений (см. главу 10).

- Чувствительность к регистру. Сценарии, экспортированные в файлы *.swf* формата Flash Player 6, не чувствительны к регистру даже во Flash Player 7. Но будьте внимательны! Код на ActionScript 1.0 в файле *.swf* формата Flash Player 7 при воспроизведении во Flash Player 7 чувствителен к регистру (см. табл. 12.)
- Приведение типов (см. раздел «Поддержка динамического приведения типов» в главе 3).

В табл. 1.2. приведены данные по чувствительности к регистру для всех возможных сочетаний версии файла *.swf* и версии Flash Player пользователя. Обратите внимание, что чувствительность к регистру во время выполнения не связана с выбранной версией компилятора ActionScript и зависит только от формата экспортируемого файла *.swf* и версии Flash Player. Иначе говоря, и ActionScript 1.0, и ActionScript 2.0 чувствительны к регистру при экспорте в файлы *.swf* формата Flash Player 7 и воспроизведении во Flash Player 7. В других случаях код нечувствителен к регистру, кроме исключений, приведенных в сносках табл. 1.2. Чувствительность к регистру и ее последствия во Flash Player 6 подробно рассмотрены в книге «ActionScript for Flash MX: The Definitive Guide» (O'Reilly).

Таблица 1.2. Поддержка чувствительности к регистру во время выполнения в зависимости от формата файла и версии Flash Player

Фильм, скомпилированный как ActionScript 1.0 или 2.0	**Воспроизведение во Flash Player 6**	**Воспроизведение во Flash Player 7**
Файл *.swf* в формате Flash Player 6	Нечувствительный к регистру[a]	Нечувствительный к регистру[a]
Файл *.swf* в формате Flash Player 7	Не поддерживается[b]	Чувствительный к регистру

[a] Идентификаторы (т. е. имена переменных и свойств), имена функций, метки кадров и ID экспорта символов в файлах *.swf* формата Flash Player 6 нечувствительны к регистру. Однако зарезервированные слова, такие как «if», чувствительны к регистру даже во Flash Player 6.

[b] Flash Player 6 не может воспроизводить файлы *.swf* формата Flash Player 7.

Изменения, внесенные в ActionScript 1.0 во Flash Player 7

В файле *.swf* формата Flash Player 7, выполняющемся во Flash Player 7, некоторый код на ActionScript 1.0 ведет себя иначе, чем во Flash Player 6. Эти изменения приближают Flash Player 7 к полному соответствию ECMAScript 3. В частности:

- Значение `undefined` (неопределенный) в числовом контексте преобразуется в число `NaN`, а в строковом контексте – в строку «undefined» (во Flash Player 6 `undefined` преобразуется в число 0 и в пустую строку «»).

- Любая непустая строка в булевом контексте преобразуется в булево значение `true` (во Flash Player 6 строка преобразуется в `true`, только если она может быть преобразована в действительное число, отличное от нуля; в противном случае она преобразуется в `false`).
- Идентификаторы (имена функций, имена переменных, имена свойств и т. д.) чувствительны к регистру. Например, идентификаторы `firstName` и `firstname` во Flash Player 7 ссылаются на две совершенно разные переменные. Во Flash Player 6 эти идентификаторы обозначали бы одну и ту же переменную. (Однако, как обычно, метки кадров и ID экспорта символов в библиотеке нечувствительны к регистру.)

С этими изменениями соприкасается только тот, кто обновляет фильм в формате Flash Player 6 в формат Flash Player 7, чтобы получить доступ к возможностям, характерным лишь для Flash Player 7. То есть при обновлении фильма необходимо протестировать и по возможности модифицировать код, чтобы гарантировать, что в формате Flash Player 7 он будет работать так же, как и в формате Flash Player 6. Если возможности Flash Player 7 не нужны, то можно продолжать экспортировать фильм в формат Flash Player 6, и, как правило, он будет выполняться во Flash Player 7 точно так же, как и во Flash Player 6. Последний вариант мы не будем рассматривать подробно.

Компания Macromedia идет на огромные затраты, чтобы обеспечить нормальную работу фильмов, экспортированных в более старые версии формата *.swf*, такие как формат Flash Player 6, в неизменном виде даже при воспроизведении в более новых проигрывателях, таких как Flash Player 7. Однако, публикуя фильм в формате Flash Player 7, необходимо помнить об изменениях, введенных с момента появления предыдущей версии формата *.swf*. То есть изменения, которые необходимо внести в код ActionScript, зависят от версии файла *.swf*, а не от версии Flash Player.

Конечно, любые вновь создаваемые файлы *.swf*, экспортируемые в формат Flash Player 7 (а не только те файлы *.swf*, которые обновляются из формата Flash Player 6), должны подчиняться новым соглашениям, так что забывать о них нельзя никогда. Помните, что эти новые соглашения ставят ActionScript в один ряд с другими языками программирования, такими как JavaScript и Java, что упрощает задачу перенесения кода из ActionScript в другие языки и обратно.

Слэш-синтаксис Flash 4 в ActionScript 2.0 не поддерживается

Во Flash 4 и последующих версиях доступ к переменным может быть осуществлен с помощью так называемого «слэш-синтаксиса». Например, во Flash 4 следующий код является ссылкой на переменную `x` клипа `ball`:

```
/ball:x
```

При попытках применения этого синтаксиса с компилятором ActionScript 2.0, независимо от формата, в который экспортируется фильм – Flash Player 6 или Flash Player 7, генерируется следующая ошибка:

```
Unexpected '/' encountered
```

Займемся ООП

Итак, мы знаем, что может предложить ActionScript 2.0, и можем всерьез начать изучение объектно-ориентированного программирования во Flash. Когда будете готовы приступить к делу, переходите к главе 2!

2

Объектно-ориентированный ActionScript

Как это ни парадоксально, но пользователи Flash, не знающие объектно-ориентированного программирования (ООП), часто хорошо знакомы с многими понятиями ООП, даже не имея представления об их официальных названиях. Эта глава несколько проясняет терминологию и ключевые понятия ООП. Ее также можно рассматривать как краткий обзор ООП в ActionScript для опытных программистов, делающих первые шаги во Flash-разработке.

Процедурное и объектно-ориентированное программирование

Традиционное программирование заключается в группировке различных команд в *процедуры* (*procedures*). Процедуры выполняют определенные задачи, совершенно ничего не зная или не заботясь обо всей программе. Например, процедура может производить вычисления и возвращать результат. В процедурной Flash-программе повторяющиеся задачи хранятся в *функциях* (*functions*), а данные – в *переменных* (*variables*). Выполнение программы заключается в выполнении функций и изменении значений переменных, обычно с целью обработки ввода или генерации вывода. Процедурное программирование удобно для реализации определенных задач, однако по мере увеличения размера или усложнения приложений взаимодействия между процедурами (и программистами, их использующими) становятся все более многочисленными, процедурные программы могут стать громоздкими. Тогда их будет тяжело обслуживать, трудно отлаживать и сложно модернизировать.

Объектно-ориентированное программирование (ООП) – иной подход, призванный решить некоторые из проблем разработки и обслуживания, возникающие обычно с большими процедурными программами. ООП призвано упростить работу со сложными приложениями, разбив

их на самостоятельные модули, взаимодействующие друг с другом. ООП обеспечивает нам возможность переносить абстрактные идеи и осязаемые реалии физического мира в соответствующие части программы («объекты» ООП). Оно также позволяет приложению создавать и управлять множеством экземпляров одного и того же объекта, как это обычно требуется в пользовательских интерфейсах. Например, нам может понадобиться 20 машин в симуляторе гонок, 2 игрока в игре или 4 кнопки-переключателя на форме.

При условии правильного применения ООП обеспечивает программе дополнительный уровень концептуальной организации. Взаимосвязанные функции и переменные группируются в отдельные *классы* (*classes*), каждый из которых является самостоятельной частью программы с собственными целями и задачами. Классы применяются для создания отдельных объектов, выполняющих функции и задающих переменные друг для друга, формируя поведение программы. Организация кода в классы упрощает создание программы, которая посредством реальных компонентов прекрасно отображает реальные задачи. В частях II и III данной книги рассмотрены некоторые обычные ситуации, с которыми вы столкнетесь в ActionScript, и показано, как решать их с помощью ООП. Но перед тем как перейти к изучению прикладных задач, давайте кратко рассмотрим основные концепции ООП.

Ключевые понятия объектно-ориентированного программирования

Объект (*object*) – это самостоятельный программный модуль, содержащий взаимосвязанные функции (называемые его *методами* (*methods*)) и переменные (называемые его *свойствами* (*properties*)). Отдельные объекты создаются из *классов*, которые предоставляют схему методов и свойств объекта. То есть класс – это шаблон, по которому создается объект. Классы могут представлять в программе абстрактные понятия, такие как таймер, или физические сущности, такие как разворачивающееся меню или космический корабль. Один класс может служить для создания любого количества объектов, каждый из которых будет иметь одну и ту же общую структуру, это что-то вроде выпекания множества кексов по одному рецепту. Например, в ООП-симуляторе космического боя одновременно на экране может быть 20 объектов *SpaceShip* (космический корабль), созданных из одного класса *SpaceShip*. Аналогично в игре может быть всего один класс *2dVector*, представляющий математический вектор, и тысячи объектов *2dVector*.

Термин *экземпляр* (*instance*) часто употребляется как синоним *объекта*. Например, фразы «Создайте новый экземпляр *SpaceShip*» и «Создайте новый объект *SpaceShip*» означают одно и то же. Процесс создания нового объекта из класса иногда называется *инстанциированием* (*instantiation*).

Собирая объектно-ориентированную программу, мы:

1. Создаем один или более классов.
2. Создаем объекты (т. е. *экземпляры*) этих классов.
3. Указываем объектам, что делать.

Поведение программы определяется тем, что *делают* объекты.

Программа может использовать не только классы, созданные нами, но и любой класс, встроенный во Flash Player. Например, создавать объекты *Sound* с помощью встроенного класса *Sound* (звук). Отдельный объект *Sound* представляет звук или группу звуков и управляет им(и). Его метод *setVolume( )* может увеличивать или уменьшать громкость звука. Его метод *loadSound( )* может загружать и воспроизводить звуковые файлы в формате MP3. А свойство `duration` может сообщить длительность загружаемого звука в миллисекундах. Вместе встроенные классы и наши специальные классы формируют основные строительные блоки всех ООП-приложений во Flash.

Синтаксис класса

Возьмемся сразу за конкретный пример. Помните, я предложил создать в игре-симуляторе космического боя класс *SpaceShip*. ActionScript, описывающий этот класс, мог бы выглядеть как исходный код, приведенный в примере 2.1 (не переживайте, если многое в этом коде ново для вас; в следующих главах мы подробно изучим его).

Пример 2.1. Класс SpaceShip

```
class SpaceShip {
  // Это открытое свойство speed (скорость).
  public var speed:Number;
 // Это закрытое свойство damage (повреждение).
  private var damage:Number;
  // Это функция-конструктор, инициализирующая
  // каждый экземпляр SpaceShip.
  public function SpaceShip () {
    speed = 100;
    damage = 0;
  }
  // Это открытый метод fireMissile() (запустить ракету).
  public function fireMissile ():Void {
    // Здесь код, который отвечает за запуск ракеты.
  }
  // Это открытый метод thrust() (тяга).
  public function thrust ():Void {
    // Здесь код, который отвечает за движение корабля.
  }
}
```

Обратите внимание, как ловко класс *SpaceShip* группирует взаимосвязанные аспекты программы (как и все классы). Переменные (свойства),