ШЭМ БАНГАЛ

ActionScript

Основы

СИМВОЛ

friendsof

DESIGNER TO DESIGNER™

Foundation ActionScript

Sham Bhangal

friendsof

DESIGNER TO DESIGNER™

ActionScript
Основы

Шэм Бангал

Санкт-Петербург
2002

Шэм Бангал

ActionScript. Основы

Перевод И. Асеева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Михайлов*
Редактор	*Р. Павлов*
Художник	*С. Борин*
Корректура	*С. Беляева*
Верстка	*А. Дорошенко*

Бангал Ш.
ActionScript. Основы. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 480 с., ил.
ISBN 5-93286-031-6

К настоящему времени Flash 5 стал стандартом разработки динамических приложений в Сети, а язык ActionScript из интересной возможности превратился в дисциплину, обязательную при освоении Flash. Внесенные во Flash 5 усовершенствования сделали ActionScript более удобным для дизайнеров за счет множества выпадающих меню и встроенных функций; стандартных, принятых в JavaScript обозначений с точкой, делающих команды легко запоминаемыми; новых способов включения звукового сопровождения и более развитых средств тестирования и отладки. С обеспечиваемой ActionScript интерактивностью веб-дизайн внезапно вышел на новый уровень.

Книга «ActionScript. Основы» написана опытным веб-дизайнером для своих коллег – веб-дизайнеров, уже владеющих основами Flash, но не имеющих опыта в программировании. Понимая, насколько сложно освоить премудрости программирования человеку творческому, далекому от этого занятия, автор последовательно, шаг за шагом знакомит читателя с основными понятиями программирования и их связью с дизайном, после чего основное внимание уделяется применению ActionScript в реальных веб-проектах, обсуждаются планирование больших ActionScript-проектов и динамическая анимация для интерактивных игр. Изучив материал, вы сможете превратить простую линейную анимацию в презентацию с богатыми интерактивными возможностями.

ISBN 5-93286-031-6
ISBN 1-903450-32-2 (англ)

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
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 14.05.2002. Формат 70x100$^{1}/_{16}$.
Печать офсетная. Объем 30 печ. л. Тираж 3000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Введение 9

Добро пожаловать 9
Загружаемые файлы 12
Что вам необходимо знать 12
Movie Explorer 12
Поддержка: мы готовы помочь вам 15

1. Начинаем 17

События и как с ними обращаться 17
Комментарии 27
Команды, аргументы и свойства 29
Инструкции 31
Синтаксис 32
Резюме 45

2. Строим планы 46

Планирование 47
Построение сценариев ActionScript 52
Веб-сайт stun:design 62
Резюме 66

3. Действия временной диаграммы 67

Пути временных диаграмм 68
Добавление основных действий на временную диаграмму 74
Добавление меток на временную диаграмму 82
Веб-сайт stun:design 96
Резюме 100

4. Основы интерактивности 101

Интерактивность и клипы 109
Создание интерактивности при помощи событий кнопок 119
События кнопок и навигация 121
Резюме 134

5. Трюки с навигацией 135

Невидимые кнопки 135
Структура сайта 140
Интерфейс stun:design 144
Перетаскиваемые окна 147
Резюме 159

6. Переменные 161

Переменные 162
Ввод и вывод 166
Именование переменной 173
Хранение чисел 175
Логические значения 178
Объекты, методы и свойства в ActionScript 181
Резюме 187

7. Циклы и принятие решений 188

Принятие решений 189
Действие if 192
Действие else if 198
Действие else 202
Циклы 205
Массивы 219
Игра в виселицу 225
Веб-сайт stun:design 236
Резюме 241

8. Звук 242

Объект sound 242
ActionScript и стриминг 259
Веб-сайт stun:design 260
Резюме 273

9. Интерактивность в развитии 274

И снова об экземплярах 275
Классы 276
Определение объектов 282
Интерактивность и свойства объектов 283
Галерея stun:design 297
Веб-сайт stun:design 305
Создание структуры 306
Резюме 320

10. Модульный ActionScript 321

Модульный ActionScript 322
Управляющие клипы 333
Инерционное поведение 343
Эффект роения 347
Настраиваемые клипы 353
Веб-сайт stun:design 356
Сокрытие страниц 363
Резюме 372

11. Спрайты 373

Что такое спрайт? 373
Управление 375
Движение 378
Столкновения 378
Планирование игры stun:zapper 383
Мир игры (глобальный уровень) 385
Игрок: объект ship (корабль) и его интерфейсы 390
Роящийся пришелец: объект alien1 и его интерфейсы 393
Пуля: объект bullet и его интерфейсы 398
Веб-сайт stun:design 401
Страница 1: Заставка 401
Страница 2: Введение 406
Страница 3: Основное содержимое 413
Графика окна и кнопки 417
Клип содержимого 423
Управляющий клип окна 424
Интеграция 428
Две последние кнопки 430
Последнее напутствие 433

Приложение A. Словарь 435

Приложение B. События 440

Приложение C. Свойства 445

Приложение D. Приоритеты операций 452

Приложение E. Совместимость Flash 5 458

Приложение F. Отладка 467

Алфавитный указатель 473

Об авторе

В начале карьеры **Шэм Бангал** работал инженером и специализировался в области промышленных компьютерных систем отображения и управления. Свой досуг он посвящал веб-дизайну, который постепенно занимал все больше и больше времени и вытеснил, в конце концов, инженерную деятельность.

Сейчас Шэм занимается тем, что пишет книги для издательства friends of ED, и это оставляет ему все меньше времени на веб-дизайн... Забавно, как жизнь ходит по кругу!

Шэм живет со своей подругой Карен в Манчестере в Великобритании.

Благодарю Андерса и Джейн за их вклад в проект stun:design. Надеюсь, я не слишком изнурил их работой.

Я изучал веб-дизайн около года, спрятавшись на принадлежащей Роджеру и Энни Вильямс маленькой ферме в глубине графства Сомерсет. Эта книга посвящается вам обоим. Благодарю за то, что вы приютили меня вместе с моей странной привычкой сочетать дневную работу с сетевыми забавами. Прошу прощения за трехсотваттную аудиосистему, которая помогала по ночам бодрствовать мне (и наверное, всей деревне).

Особенная благодарность Роджеру за пиво и жареные трупы животных в жаркие солнечные дни. Для мужчины, у которого подружка вегетарианка, мясо подобно глотку воды в пустыне.

Введение

Добро пожаловать

Интерактивность, которую предоставляет ActionScript для Flash 5, устанавливает новый стандарт в веб-дизайне. ActionScript – это самое значительное усовершенствование Flash 5 по сравнению с Flash 4. Он превратился из простого языка подготовки сценариев в полноценную объектно-ориентированную среду программирования, с улучшенным интерфейсом для ввода и редактирования текста, с новыми командами, обогатившими набор действий, и многими новыми операциями, применяемыми к фильмам. Долгожданные обозначения с точкой позволяют легче запоминать команды, и написание программ стало более удобным по сравнению с Flash 4.

В общем, ActionScript стал теперь значительно более мощным и профессиональным инструментом, который позволяет включать в ваши веб-сайты умопомрачительные эффекты, оставляя размер файлов в разумных пределах, и дает возможность повысить уровень интерактивности ваших разработок.

Сочетание всех этих и многих других усовершествований предоставляет новые средства и возможности управления, преимуществами которых пользуются лучшие Flash-дизайнеры при создании знакомых вам впечатляющих эффектов. Они не просто следуют правилам, они переопределяют их и учат Flash действовать совершенно по-новому. Это то, что сможете делать вы, прочитав эту книгу.

Мы стремимся дать вам твердые основы, которыми нужно овладеть, чтобы держаться в русле событий. Вы начнете с изучения того, как планировать ActionScript-проект, изучите действия, обеспечивающие простое управление временной диаграммой, совершите полный тур по кодам, допускающим повторное использование, узнаете, как работать со звуком, и создадите свою первую интерактивную игру. ActionScript может показаться какой-то совершенно новой игрой, но эта книга докажет вам, что ее правила не так уж сильно отличаются от старых и что конечный результат оказывается более эффектным, чем все то, что вы до сих пор видели!

Одна оговорка – обучение ActionScript вовсе не запрещает вам оставаться дизайнером. Эта книга открывает перед вами новый путь и ве-

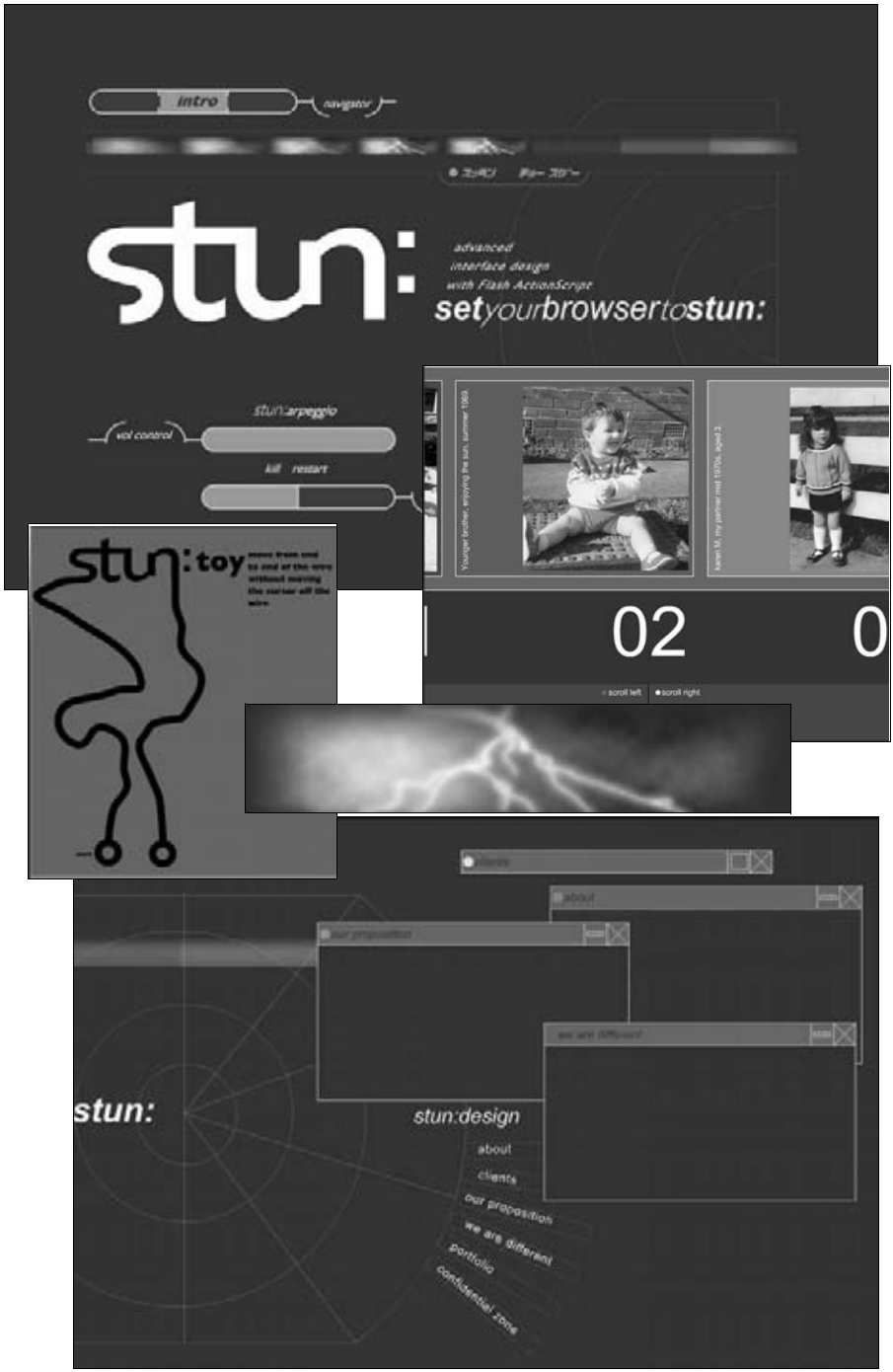
дет вас от линейной анимации к созданию современных динамических Flash веб-сайтов, ни на минуту не упуская из виду, что для веб-дизайнера важнее всего, какие новые средства для реализации его собственных замыслов может дать ActionScript. Мы не хотим сделать из вас программистов, рассуждающих в терминах нулей и единиц (будто больше нет ничего на свете). Идеи остаются самым дорогим товаром, и мы думаем, что облегчить путь от воображения к реальности точно так же важно, как и научить вас всем этим скобкам и жутким символам, в которых легко запутаться, словно в паутине.

Задачи и философия этой книги

Нельзя не признать, что на 500 страницах невозможно научить всему, что следует знать об ActionScript, но мы собираемся дать вам твердые основы, обеспечивающие глубокое знание ключевых понятий ActionScript. Мы намерены с самого начала книги использовать такие приемы, как обозначения с точкой, которые часто называют «продвинутыми», но которые предлагают простой путь к овладению неизмеримой мощью Flash 5.

Мы не верим тем книгам, где основные принципы кодирования демонстрируются на примитивных примерах, не обладающих коммерческой ценностью и не применимых в реальной жизни. Обучение на теоретических примерах – это не для нас. Мы решили, что всю силу ActionScript, которую вам предстоит применять в своих разработках, вы сможете лучше оценить, если связать изучаемое с **реальным** веб-сайтом. В процессе проработки этой книги мы будем строить **stun:design**, профессиональный веб-сайт, который полностью создан на ActionScript и существует как коммерческий проект. Это поможет закрепить полученные знания путем применения их в ситуации из реальной жизни, где сроки поджимают, клиенты жалуются и все идет не так, как надо.

Делая акцент скорее на практику, нежели на теорию, в каждой главе мы следуем основной модели и вводим новую тему, проиллюстрированную примерами, которые постепенно усложняются и наконец складываются вместе в работающий веб-сайт, полный трюков и хитростей ActionScript. Зайдите на *stundesign.com* – и вы увидите, что будете уметь, прочитав эту книгу.


intro
navigator
stun:
advanced
interface design
with Flash ActionScript
setyourbrowsertostun:
stun:arpeggio
vol control
kill restart
stun:toy
02
0
scroll left
scroll right
stun:
stun:design
about
clients
our proposition
we are different
portfolio
confidential zone

Загружаемые файлы

Мы также подготовили полностью законченные FLA-файлы для каждого упражнения. Их можно в готовом виде, главу за главой, скачать с *friendsofed.com*. Пользуйтесь ими в собственных разработках – или не пользуйтесь, как вам угодно, но они определенно могут пригодиться для того, чтобы сравнить сделанное вами с законченным продуктом Шэма, когда вдруг окажется, что вы совсем перестали понимать происходящее.

Что вам необходимо знать

Вы взялись за эту книгу, так что мы думаем, что вам знакомы основы Flash. Мы предполагаем, что вы уже создавали обычные фильмы, монтировали кадры на временной диаграмме, но при этом слышали, что Flash 5 дает гораздо больше возможностей для творчества.

Вы, может быть, даже прочли книгу «Foundation Flash 5» издательства «friends of ED», которая дала вам основательные познания во Flash и пробудила желание продолжить путешествие.

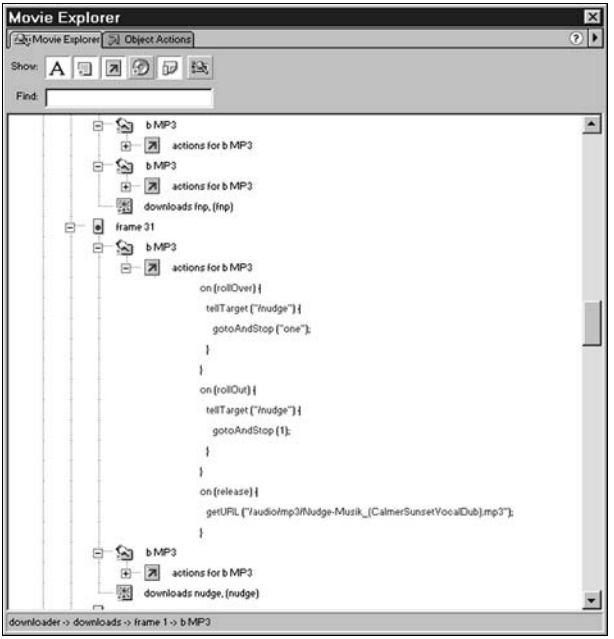
ActionScript для дизайнера – как электродрель, заменившая старый коловорот, который был, конечно, полезен, но слабоват. Как и всякий электроинструмент, получающий энергию по проводам, эта электродрель несет с собой опасность наделать много бед, если с ней неправильно обращаться. Чтобы этого не случилось, прочтите, пожалуйста, следующие краткие инструкции по технике безопасности. Мы собираемся пройтись по небольшому списку общих установок Flash 5, которые необходимо сделать перед началом работы, чтобы приступить к созданию сценариев прямо с первой главы.

Работая с Flash, вы научитесь превращать объекты в символы (при помощи клавиши <F8> или Insert ▸ Convert to Symbol) – хороший прием, позволяющий сократить время загрузки. Если вы вытягиваете два экземпляра символа из библиотеки, ActionScript столкнется с проблемой, как различать их. Поэтому следует снабжать объекты **именами**, чтобы ActionScript знал, где находится каждый экземпляр, и мог ими управлять. Не беспокойтесь, мы покажем, как это делается, когда придет время. Вы умеете давать имена клипам (movieclips), а в этой книге мы встретимся с множеством символов клипов!

Movie Explorer

По мере изучения этой книги ваши FLA будут все сложнее, и может случиться так, что создав фильм или фрагмент сценария, вы затем потеряете его в безграничном пространстве Flash. Эту проблему решит Movie Explorer, с помощью которого легко отыскивать символы, фильмы

и сценарии. В любой момент можно вызвать Movie Explorer при помощи Window ▸ Movie Explorer.



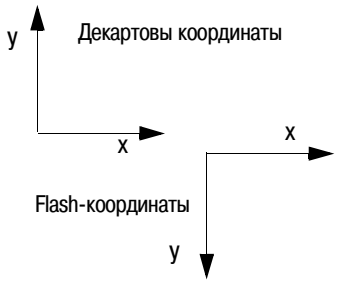
Так выглядит окно Movie Explorer – вашего проводника по просторам Flash

Поиграв немного с Movie Explorer, вы обнаружите, что с его помощью можно сделать несколько вещей. Если распечатать его содержимое, это поможет понять, что происходит, когда мы начнем создавать во Flash необычные конструкции, вроде вкладывания одного фильма в другой. Вы сможете найти все объекты на временной диаграмме, нажав кнопку с двумя квадратиками  .

Действительно полезно и поле Find:, с помощью которого можно найти все строки ActionScript, управляющие данным объектом. Если вы заметили, что объект ведет себя странно и вопреки ожиданиям, то это позволит разобраться с его поведением. Можно пройти от библиотечного шаблона ко всем экземплярам соответствующего библиотечного символа и проследить, что меняется при изменении библиотечной версии этого символа. Войдите в тексты сценариев ActionScript, нажав на кнопку со стрелкой  , и Movie Explorer позволит вам распечатать все коды разом, причем это единственное место во Flash, где можно это сделать. Возможно, вы пока не видите смысла в некоторых из этих функций, но все равно подружитесь с Movie Explorer – в трудную минуту он вас выручит.

Рабочая область

Если вы включите линейки при помощи View ▸ Rulers, то заметите, что цифровые деления на линейках начинаются с верхнего левого угла, и их значения растут слева направо (вдоль оси X) и от левого верхнего угла вниз (вдоль оси Y). Такие направления осей приняты на печатных прессах и у дизайнеров, но они не совпадают с направлениями, принятыми в геометрии Декарта и у математиков, которые обычно направляют ось Y вверх, так что начало координат находится в левом нижнем углу.



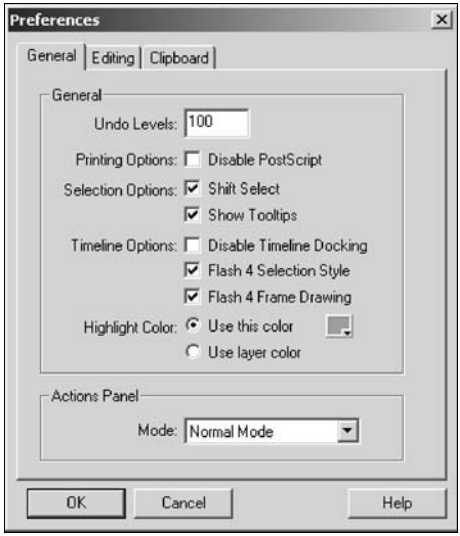
Особенно это может смутить тех, кто имеет математическую подготовку – приходится увеличивать координату Y, чтобы опустить объект. Будьте внимательны!

Установки Flash

Мы собираемся начать программировать прямо с первой главы (не пугайтесь, это гораздо проще, чем вы думаете!), поэтому важно согласовать наши Preferences (общие установки) до начала работы, чтобы Flash у всех нас работал одинаково. Вызовите окно Preferences во Flash, выбрав Edit ▸ Preferences.

Все установки, которые необходимо сделать, находятся на вкладке General. В группе Selection Options выберите Shift Select. Так вы определите способ множественного выбора объектов во Flash. Во многих графических программах (таких как Adobe PhotoShop) для выбора нескольких объектов или добавления объекта к выделенному набору требуется нажимать клавишу <Shift>. Если выделить что-нибудь при ненажатой клавише <Shift>, все предыдущие выделения теряются. Чтобы заставить Flash работать именно таким образом, следует установить флажок Shift Select. Если вы его сбросите, Flash позволит осуществлять множественный выбор простым щелчком мышью. Затем выделение можно будет сбросить, щелкнув в пустой области. Такой способ действий может вызвать затруднения у тех, кто привык к методу Shift Select, стандартному для большинства графических и 3D-программ.

Также следует обратить внимание на группу Timeline Options. Flash 5 предлагает слегка отличающийся от Flash 4 способ выбора кадров. Методы Flash 5 – это шаг вперед в анимации, основанной на временной диаграмме, но они могут вызывать затруднения при желании выделить отдельные кадры для редактирования ActionScript, так что поправим это. Установите флажки Flash 4 Selection Style и Flash 4 Frame Drawing. Flash 4 Selection Style делает рисование кадров в стиле Flash 5 излишним.



Пока вы находитесь в диалоговом окне Preferences, убедитесь, что установлен флажок Show Tooltips, так как во всплывающих подсказках часто содержится ценная информация. Окно Preferences теперь должно выглядеть примерно так.

Поддержка: мы готовы помочь вам

Если у вас есть вопросы по этой книге или относительно издательства friends of ED, заходите на наш веб-сайт, где можно найти много e-mail-адресов или просто воспользоваться адресом *feedback@frindsofed.com*.

На этом веб-сайте вы найдете также массу других интересных вещей: интервью со знаменитыми дизайнерами, отрывки из других наших книг, дискуссии, в которых можно принять участие, а можно просто посидеть и посмотреть, о чем говорят между собой другие дизайнеры. Так что если у вас есть замечания или вопросы, пишите нам – именно для этого мы там и находимся. Мы любим читать ваши отклики.

Об оформлении этой книги

Мы как только могли старались сделать эту книгу ясной и легкой для чтения, а поэтому использовали всего несколько стилей:

- Когда вы впервые встретите важное слово, оно будет выделено **полужирным** шрифтом, а впоследствии будет набираться обычным шрифтом.
- Особым шрифтом выделяются технические термины, фразы, появляющиеся на экране, и `коды`.
- Команды меню мы будем записывать в таком виде: Menu ▸ Sub-menu ▸ Sub-menu.
- Когда речь пойдет о том, что действительно важно, эта информация будет выделена так:

Примечание

Это очень важный материал – не пропустите его!

- Прорабатываемые упражнения выглядят примерно так:
  1. Откройте Flash.
  2. Создайте новый файл фильма и сохраните его как TestMovie.fla.
  3. И так далее.

PC и Mac

Чтобы книга была возможно более легкой для чтения, мы по умолчанию используем команды PC, так что всякий раз, когда речь пойдет о командах с мышью, вам не придется читать что-нибудь вроде: «щелкните правой кнопкой мыши на PC или левой кнопкой при нажатой клавише <Ctrl> на Mac».

Обе инструкции мы пишем только там, где имеется отличие от стандартного соответствия команд Mac командам на PC и соответствующая команда необходима. Когда мы просто пишем: «щелкните мышью», мы подразумеваем щелчок *левой* кнопкой мыши на PC и просто *щелчок* на Mac. Обычное соответствие команд таково:

PC	Mac
Щелчок правой кнопкой	Щелчок левой кнопкой при нажатой клавише <Ctrl>
Щелчок левой кнопкой при нажатой клавише <Ctrl>	Щелчок левой кнопкой при нажатой клавише <Apple>
Нажатие клавиши <Z> при нажатой клавише <Ctrl> (Отмена)	Нажатие клавиши <Z> при нажатой клавише <Apple>
Нажатие клавиши <Enter> (Ввод) при нажатой клавише <Ctrl>	Нажатие клавиши <Enter> при нажатой клавише <Apple>

Теперь мы готовы приняться за дело. Начнем!

- *Переменные как контейнеры для данных*
- *Различные типы переменных – числовые, строковые и логические*
- *Выражения, порождающие значения, хранимые в переменных*
- *Создание полей ввода-вывода, обеспечивающих большую интерактивность, чем кнопки*
- *Как объекты могут хранить наборы значений и не только...*

6

Переменные

До сих пор наш ActionScript следил только за тем, что происходит в настоящий момент – где находится указатель мыши или какой кадр является текущим, – но не запоминал, что было в прошлом, и не думал о том, что может случиться в будущем. Он не мог делать этого, поскольку не располагал памятью – не мог запомнить, что происходило ранее, и потому ему не на что было опереться при принятии решений, предсказании будущего или при создании новой информации. Как только в нашем примере из предыдущей главы клип заканчивался, Flash ничего уже о нем не помнил.

С точки зрения Flash, отсутствие памяти равносильно отсутствию места для хранения **данных**. Сохранение данных позволит Flash получать информацию относительно событий прошлого. Еще важнее, возможно, то, что если у Flash будет где хранить данные, он сможет запрашивать у пользователя ввод таковых, а затем, по их получении, действовать, на них основываясь.

Нам необходимо место для хранения данных, к которым мы имели бы беспрепятственный доступ и которые, вероятно, захотели бы время от времени заменять. Место, в котором мы храним данные, – это **переменная**. А вот ее по-канцелярски звучащее определение:

Переменная – это именованное место в памяти, предназначенное для хранения данных, которым свойственно часто обновляться.

В этом разделе мы увидим, что на деле означает это определение и как при помощи переменных Flash отслеживает происходящее. Мы уви-

дим также, как Flash способен применять доступные ему сведения для принятия решений и предвидения и как он осуществляет вычисления. Не исключено, что поначалу эта глава может вызвать у вас воспоминания об уроках математики, однако переменные не обязаны содержать квадрат скорости света, с тем же успехом они могут заключать в себе высоту, на которую должен подпрыгнуть герой вашего Flash-мультфильма. Это не *настоящая* математика – это просто эффективное средство заставить Flash делать те забавные штуки, которых вы от него ждете.

В главе 7 мы рассмотрим циклы и процесс принятия решений, в котором большей частью и применяются переменные. Поэтому эти две темы удобно рассматривать вместе. Использование переменных на нашем сайте stun:design я оставил на конец следующей главы. Я могу порадовать вас тем, что тогда вы сможете разбить на склоне пика ActionScript базовый лагерь номер два, из которого сквозь клубящиеся облака уже будет открываться вид на блистающую снегами вершину. К этому моменту вы уже изучите основные конструкции ActionScript, на которые опирается все остальное, и сможете перейти к изучению материала следующих глав, посвященных звуковым эффектам, спрайтам, объектам и всему прочему, что поможет вам многого добиться.

Переменные

Переменная – это именованный «контейнер», предназначенный для хранения данных, которые могут изменяться. Когда бы вам ни потребовались эти данные, нужно только сообщить Flash имя переменной, а уж он сам найдет ее и выдаст ее содержимое. Это понятие может показаться не вполне ясным, так что рассмотрим его в непрограммистских терминах.

У вас в кошельке, я надеюсь, есть деньги. Их количество растет или уменьшается в зависимости от ваших трат. Важно отметить, что обычно кошелек сам по себе не представляет ценности: деньги, лежащие в нем, – вот что определяет его цену. В супермаркете пустой кошелек не примут в качестве оплаты покупок.

Посмотрим на дело с другой стороны. Ваш кошелек – это вещь, содержимое которой вы рассматриваете как принадлежащие вам деньги. Слыша слово «кошелек», вы всегда знаете, что внутри обозначаемого этим словом предмета находятся деньги, а не носки или замороженная пицца. Вам также известно, что количество этих денег недолго остается неизменным.

Для хранения денег используются, однако, и другие контейнеры. Так, на кухне может стоять жестянка с деньгами на кофе, чай и шоколадные бисквиты. У вас, быть может, есть глиняная свинья с прорезью или бутылка, куда вы в течение года ссыпаете мелочь. (Я, например,

каждое Рождество трачу содержимое такой маленькой бутылочки с деньгами на покупку большой бутылки чего-нибудь еще.) У вас есть несколько различных контейнеров, поскольку вам необходимо хранить деньги для каждой цели отдельно.

Итак, у меня есть три разных места для хранения денег, в каждом из которых содержится своя постоянно меняющаяся сумма:

- Кошелек с деньгами – на повседневные расходы
- Жестянка с деньгами – на чай и бисквиты
- Бутылочка с деньгами – на бутылку

Эти места будут моими «переменными», в каждой из которых хранится своя сумма, способная измениться в любой момент. Моя первая переменная – это *кошелек,* и сумма, которая в нем содержится, предназначена для оплаты ежедневных покупок. У меня есть также переменная *жестянка* с «чайными» деньгами и переменная *бутылочка* с «пивными» деньгами.

Важным аспектом моего высокопрофессионального управления финансами является то, чтобы деньги, предназначенные для определенной цели, хранились там, где им полагается. Вообразите себе трагедию, которая случится, если я перепутаю свою переменную *жестянка* с переменной *бутылочка*, и в результате к концу года у меня окажется достаточно денег, чтобы напоить чаем весь Китай, но лишь два с половиной доллара на стакан пива в новогоднюю ночь.

По счастью, маловероятно, что я спутаю такие осязаемые предметы, как жестянка, бутылка и кошелек, но при программировании мы лишены этой роскоши. Чтобы избежать возможной трагедии, я буду вынужден наклеить особую метку на каждый из контейнеров, иначе для меня неразличимых, то есть присвоить имя каждой из переменных, чтобы наверняка знать, что каждая из них содержит.

Типы и значения

В реальном мире мы окружены множеством различных хранилищ, предназначенных для того, чтобы держать в них разного рода вещи:

- Кошелек создан для хранения денег
- Холодильник предназначен для хранения еды
- Платяной шкаф спроектирован для хранения одежды

В ActionScript тоже имеются значения разного вида, которые нам, возможно, захочется приберечь для дальнейшего употребления. Фактически существуют три типа значений:

- Строки, или последовательности символов, такие как `Шэм` или `Привет!`, или `a1b2c3`. Строковые значения могут иметь любую длину, от одной буквы до целого предложения, и они особенно полезны для сохранения ввода, поступающего от пользователей.

- Числа, такие как 27 или –15, или 3.142. Как мы увидим из дальнейшего обсуждения, помещение в переменные чисел может пригодиться для отслеживания положения элементов в фильмах и для управления поведением ActionScript-программ.
- Логические значения, которые бывают только двух видов: «истинно» или «ложно». Более пристально мы их рассмотрим в конце этой главы.

Некоторые языки программирования, имитируя положение дел в реальном мире, заставляют вас для хранения данных разных типов создавать разные переменные, способные хранить данные только одного типа. ActionScript, напротив, не требует, чтобы вы выражались столь определенно. ActionScript – это **нетипизированный язык**, то есть язык, в котором при создании переменной не нужно указывать тип данных, которые она будет содержать. Более того, переменные в ActionScript могут содержать в разные моменты времени данные разных типов. При необходимости Flash с ними разберется.

Выражения и литералы

Рассмотрим вкратце несколько примеров, из которых вам станет понятнее, насколько полезными могут быть переменные. Добравшись до упражнений, мы обнаружим, что Flash очень хорошо помогает нам в работе с переменными, но часто бывает приятно заранее знать, чего следует ожидать.

Проще всего сохранить некоторое значение в некоей переменной при помощи знака равенства (=). Проще всего указать, что именно вы хотите сохранить, используя **литерал (буквальное значение)**.

Литералы

Допустим, у нас имеется переменная ActionScript с именем myname. Если бы я захотел сохранить в этой переменной собственное имя, я мог бы поступить так:

```
myname = "Sham";
```

Здесь "Sham" – это строковый литерал, или буквальное значение, которое сохраняется в myname. Это *буквально* то, что содержится между двумя кавычками.

Подобным образом у нас могла бы быть еще одна переменная, скажем, myage. Чтобы сохранить значение в этой переменной, мы можем применить **числовой литерал** таким образом:

```
myage = 33;
```

Примечание

Любое встретившееся в кодах ActionScript число можно назвать числовым литералом, но простое и привычное слово «число» в большинстве случаев не вызовет недоразумений.

После того как написаны две эти строчки кода, переменные myname и myage могут использоваться в программах как непосредственные заменители содержащихся в них значений. Пока это может показаться не слишком важным, но скоро вы увидите, какой гибкости выражения можно таким образом достичь.

Прежде чем двигаться дальше, вспомним, что переменная определяется своим именем, – то, что в ней хранится некоторое значение, никак не препятствует тому, чтобы сохранить в ней другое значение (или даже значение другого *muna*). Переменная x может сначала содержать числовое значение 10, но завтра я помещу в нее строковое значение, написав просто x="sham".

Выражения

Следующий способ присвоения переменным значений использует **выражения**. Как вы увидите из упражнений, это как раз то место, где программирование начинает напоминать математику, поскольку выражения часто являются последовательностью переменных и литералов, связанных вместе математическими символами. К примеру, можно написать так:

```
result = myage - 10
```

Если myage все еще содержит значение 33, то в результате выполнения данной строки ActionScript в переменную result будет помещено значение 23 (то есть 33 - 10).

Именно при использовании выражений нужно внимательно следить за типами данных, содержащихся в переменных. Плата за удобство языка без типов состоит в *нашей собственной* ответственности за учет типов данных. Если мы не уследим за правильными сочетаниями типов в выражениях, мы рискуем получить неправильный или бессмысленный результат.

Вспомните школьные уроки алгебры. Вы могли быть уверены, что, написав равенство $a+b=c$, вы ничем не рискуете, даже если вам ничего не известно о точных значениях a и b. Что бы там ни было, в любом случае a и b – это числа, а вы знаете, что сложение чисел всегда дает число. Просто и понятно.

Программирование несколько сложнее алгебры. У нас есть переменные, содержащие не только числа, но также и другие значения – буквы и слова. Мы должны следить за типами данных и за тем, чтобы данные разных типов не смешивались.

Если переменная a=2, а переменная b=3, мы можем сложить a и b, получив значение c, имеющее смысл, т. к. 2+3=5.

Если переменная a="mad", а переменная b="house", мы можем сложить a и b, получив осмысленное значение для c, т. к. "mad"+"house"="madhouse".

Если переменная a="mad", а переменная b=3, и мы сложим их, значение c вряд ли будет иметь смысл, поскольку "mad"+3="mad3".

Чтобы переменные верно служили нам, мы должны обращаться с типами изолированно и не смешивать их между собой. Как правило, если вы вводите числа, следует ожидать числовой результат. Если вы вводите текст, ожидайте текстового результата. Если вдруг это оказалось не так, важно знать почему. Это может быть одна из тех неуловимых ошибок, которые так трудно обнаружить.

Ввод и вывод

До сих пор мы просто говорили о переменных и о тех значениях, которые они могут содержать. Пришло время для настоящего примера их применения, после которого вы начнете понимать, какую пользу они могут принести вашим веб-сайтам. Мы начнем с того, что будем сохранять в переменных строки, так как в этом случае выгоды использования переменных кажутся более наглядными.

Сохранение строк тесно связано с вводом и выводом во Flash по той простой причине, что пользователи, вообще говоря, предпочитают общаться с компьютером посредством текста, а не голых чисел. Самые сложные взаимодействия с компьютером осуществляются с использованием какого-либо рода текста – от заполнения бланка заказа в интернет-магазине до ввода ключевых слов в поисковую машину. Оба эти действия встречаются чаще, чем, скажем, ввод чисел в калькулятор, и строковые переменные сияют здесь во всей своей красе.

Создание поля ввода

1. Если мы собираемся использовать переменную для хранения введенной пользователем строки, мы должны дать пользователю возможность ввести ее. Откройте новый фильм и вызовите панель Text Options. В выпадающем меню по умолчанию содержится Static Text. Выберите в нем Input Text (рис. 6.1).

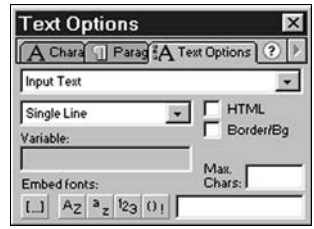


Рис. 6.1. Панель Text Options

Выберите на панели инструментов инструмент Text и щелкните один раз в рабочей области, чтобы появилось поле ввода текста (рис. 6.2).

Рис. 6.2. Поле ввода текста

Вы заметите, что появившееся поле в двух отношениях отличается от обычного текстового поля:

- Статическое текстовое поле помечено кружком в правом верхнем углу. В правом нижнем углу появившегося поля находится квадратик, сообщающий нам, что мы имеем дело с полем ввода.
- Статическое текстовое поле, возникнув, имеет ширину в один символ и растет по мере того, как добавляется текст. Наше текстовое поле, напротив, сразу предоставляет пространство для ввода нескольких символов. Это оттого, что текст в это окно не будет вводиться, пока вы не опубликуете свой файл. До этого момента Flash должен сделать приблизительное предположение об объеме вводимого текста. Принятая по умолчанию ширина поля ввода представляется Flash самой разумной. Вы можете сделать поле шире (или даже предоставить место для ввода нескольких строк текста), потянув за маленький квадратик в правом нижнем углу текстового поля.

2. Давайте еще раз взглянем на панель Text Options. Поскольку мы определили это текстовое поле как поле ввода, Flash понимает, что нам понадобится сохранить введенный текст для последующего использования. Он сделает первый шаг в создании нужной нам переменной. Поле Variable изменит свой вид, и в нем появится текст `Textfield1` – это предлагаемое по умолчанию имя переменной, связанной с только что созданным нами текстовым полем. Нам надо дать переменной, в которой будет храниться наш ввод, какое-нибудь более значащее имя. Это наше первое поле ввода, поэтому мы назовем переменную `in1` (рис. 6.3).

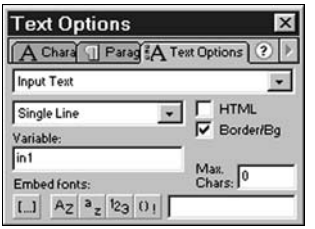


Рис. 6.3. Имя переменной указано в поле Variable панели Text Options

3. Мы создаем область для ввода текста, поэтому, когда она появится на веб-сайте, пользователь должен знать, где ему нужно щелкнуть, чтобы ввести текст. На панели Text Options установите флажок Border/Bg, чтобы Flash нарисовал рамку по границам области ввода (на рис. 6.3 флажок уже установлен). Теперь пользователь будет знать, куда ему следует вводить текст.

4. Теперь протестируйте фильм. В рабочей области вы увидите пустой прямоугольник. Щелкните на нем, и появится курсор для ввода текста. Наберите в окошке любой текст (рис. 6.4). Если введенный вами текст не умещается в окне, оно будет прокручиваться, чтобы принять весь текст в процессе ввода. Можно использовать здесь обычные клавиши копирования и вставки, <Backspace> и большинство других клавиш, хотя нельзя завершать ввод нажатием <Enter>.

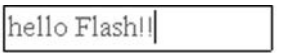


Рис. 6.4. Текст, набранный в поле ввода

Все это прекрасно, но мы не видим, что Flash делает с введенным текстом (и делает ли что-либо). Я вас уверяю, что все-таки он кое-что *делает* – он сохраняет введенный текст в переменной `in1`. Теперь мы начнем использовать переменные по назначению: пусть Flash извлечет содержимое `in1` и отобразит его в **поле вывода**.

Создание поля вывода

1. Щелкните в пустой части рабочей области инструментом Arrow, чтобы отменить выделение только что созданного поля ввода. Снова вызовите панель Text Options, в верхнем выпадающем меню выберите Dynamic Text и убедитесь, что опции Border/Bg и Selectable не отмечены (рис. 6.5).

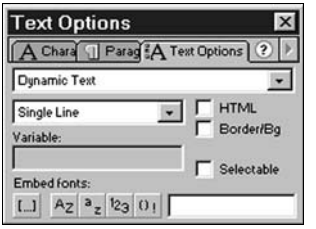


Рис. 6.5. Настройки панели Text Options перед созданием окна вывода

   Мы отменили эти опции, потому что создаваемым нами полем будет управлять Flash, а не пользователь. Оно будет служить в качестве области отображения, так что пользователю незачем видеть его границы или иметь возможность выделять его.

2. Выбрав инструмент Text, вставьте новое текстовое поле точно под предыдущим. Когда вы это сделаете, поле Variable на панели Text Options активизируется. Измените имя переменной на `in1`. Это имя указывает Flash, где искать текст, который ему следует отобразить. Разумеется, это та самая переменная, в которой мы сохраняли ввод.

3. Теперь протестируйте фильм. Перед вами будет все тот же пустой прямоугольник, *пока вы не начнете вводить текст*. А как только вы начнете это делать, динамическое текстовое поле будет повторять ваш ввод. Обратите внимание, однако, что оно имеет фиксированную длину и не будет, в отличие от поля ввода, прокручивать текст, чтобы отобразить все, что вы вводите.

Примечание

Скорость обновления содержимого обоих текстовых полей определяется скоростью смены кадров, свойственной данному фильму. При принятой по умолчанию скорости 12 fps (12 кадров в секунду) обновление поля ввода и его копии в окне вывода будет выглядеть мгновенным. Если вы измените (при помощи *Modify* ▸ *Movie*) скорость смены кадров на 1 fps, то, еще раз протестировав фильм, увидите заметное запаздывание отклика. Когда закончите экспериментировать, вернитесь к скорости 12 fps.

Сейчас Flash отображает копию текста, набираемого в поле ввода, в реальном времени, как только мы нажимаем на клавишу. Если мы ошибемся – позору не оберешься. Нужно найти способ заставить Flash подождать до тех пор, пока мы не наберем текст целиком, не отредактируем его, не насладимся плодами собственного

творчества и не решим, что пора его представить миру. Легче всего мы сможем достичь желаемого, создав кнопку enter.

4. Выберите нижнее из наших текстовых полей (поле вывода). В панели Text Options перемените значение в поле Variable с `in1` на `in2`.
5. Вставьте кнопку справа от поля ввода (рис. 6.6). Вы уже знаете, как создаются кнопки, так что сделайте это самостоятельно. Каким бы способом вы ни создали кнопку, убедитесь, что, помещая на ней текст enter, вы обозначили его как статический (другими словами, на панели Text Options он значится уже не как Dynamic, а как Static).

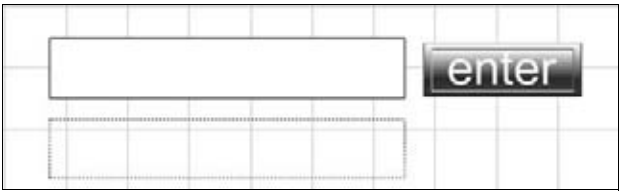


Рис. 6.6. Кнопка enter рядом с полем ввода текста в рабочей области

Примечание

Если вам не хочется самостоятельно создавать очередную кнопку, можете воспользоваться полуфабрикатом кнопки, взяв его из *Window ▸ Common Library ▸ Buttons*. Там содержится, может быть, не совсем то, что нужно, но все эти кнопки состоят из нескольких слоев, так что их действительно легко модифицировать. Если вам нужна именно моя кнопка, можете взять ее из загружаемого архива к этой главе, открыв его как библиотеку.

Теперь пора добавить к этой кнопке немножко ActionScript.

6. Выделите кнопку и вызовите окно Object Actions. Выберите действие On Mouse Event из списка Basic Actions. В списке событий оставьте отмеченным событие Release, но отметьте также событие Key Press. Немедленно активизируется текстовое поле рядом с меткой соответствующего флажка. Нажмите клавишу ввода <Enter>, назначаемую таким образом «клавишей события» (рис. 6.7).

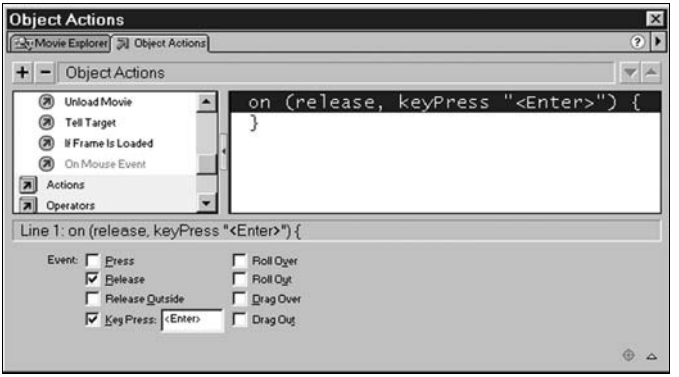


Рис. 6.7. Теперь с кнопкой связан и щелчок мышью, и нажатие клавиши

Следующим шагом мы должны заставить Flash отображать в окне вывода (переменная in2) в точности то, что мы набрали в поле ввода (переменная in1). Прежде это происходило автоматически, поскольку в обоих окнах отображалось содержимое одной и той же переменной (in1). Теперь мы должны сделать значение in2 равным значению in1.

7. При выделенной строке `on (release, keyPress "<Enter>"){` щелкните дважды по строке set variable в списке Actions (рис. 6.8).

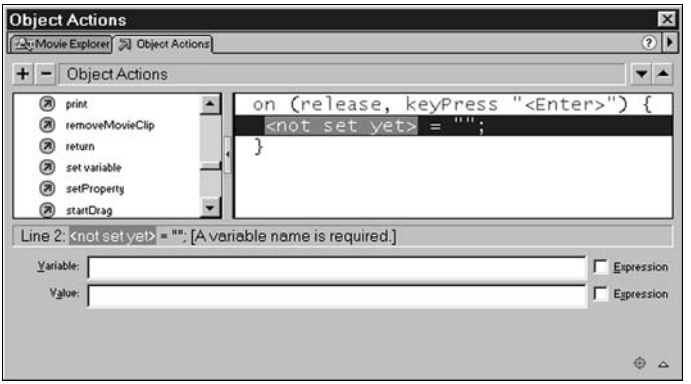


Рис. 6.8. Установка переменной

Ух! Откуда этот красный цвет? Впрочем, мы уже видели его мельком прежде. Тогда, как и сейчас, Flash сигнализировал нам цветом, что нужно ввести дополнительную информацию, но если в прошлый раз Flash предлагал значения по умолчанию (как, например, первый кадр в качестве аргумента для действия gotoAndPlay()), то в этом случае он не может оказать нам такой помощи. Присвоение значений переменным – гораздо более сложная операция.

Для начала мы должны указать имя переменной, значение которой хотим установить с помощью этого действия. Flash никоим образом не может предположить, какая именно переменная нам понадобилась. Так что вместо того чтобы гадать, он поступает самым очевидным образом и предупреждает нас, что необходимо ввести имя:

```
Line 2: <not set yet> = ""; [Требуется имя переменной.]
```

8. В поле Variable наберите in2. Теперь Flash чувствует себя лучше, и строка в правом окне превращается в

```
in2 = "";
```

Красный цвет исчез, потому что с точки зрения Flash все вполне законно – мы помещаем в переменную in2 пустую строку. Но хотя Flash удовлетворен, эта команда все равно не делает того, что нам нужно.

9. Наберите в поле Value `in1`. Flash преобразует нашу строку в

```
in2 = "in1";
```

Это по-прежнему неверно. Хотя в первой части равенства находится имя переменной, содержащей значение, которое мы хотим сохранить в `in2`, кавычки означают, что Flash поместит в переменную `in2` строковый литерал `"in1"`, а не значение, которое хранится в переменной `in1`.

10. Справа от поля Value имеется флажок Expression (рис. 6.9). Пусть он будет установлен – это сообщит Flash, что текст в поле Value является именем переменной. Тогда вы увидите, что Flash наконец справился с задачей и выдал нужное нам выражение:

```
in2 = in1;
```

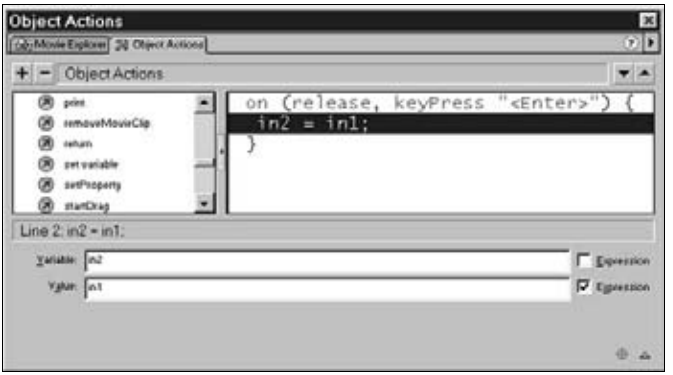


Рис. 6.9. Теперь, нажав Enter, мы приравняем in2 к in1

11. Протестируйте фильм. На этот раз, пока вы вводите текст, в нижнем поле ввод не повторяется. Нужно нажать на кнопку enter, чтобы в поле вывода появился введенный текст (рис. 6.10). Сохраните этот FLA, мы собираемся неоднократно использовать его впоследствии в качестве простого Flash-интерфейса.

Рис. 6.10. Результат работы кнопки enter

Применение строковых выражений

Итак, чтобы соединить вместе первые строительные блоки ввода и вывода текста во Flash, мы создали простые текстовые окна для получения и отображения значений основных переменных. Теперь, когда вы знаете, как это делается, коммуникационный поток между вами и

пользователем (как и уровень возможных взаимодействий) сделал большой шаг от элементарных щелчков мышью до обмена фразами и предложениями.

Имея в руках действующие средства ввода и вывода, мы теперь хотим научиться использовать строковые выражения, чтобы придать Flash «человеческое лицо», а общению с пользователем – большую доверительность. Мы будем создавать строковые выражения, комбинирующие значение переменной `in1` с некоторым другим текстом, чтобы показать больше, чем мог бы ожидать пользователь.

Работа со строками

1. Загрузите FLA, который вы только что приготовили, или же используйте готовый файл chapter06_01.fla.
2. Снова выделите кнопку enter и откройте окно Object Actions. Выделите строчку `in2 = in1;`, чтобы в нижней части окна открылись поля Variable и Value. Отредактируйте поле Value таким образом, чтобы оно соответствовало рис. 6.11.

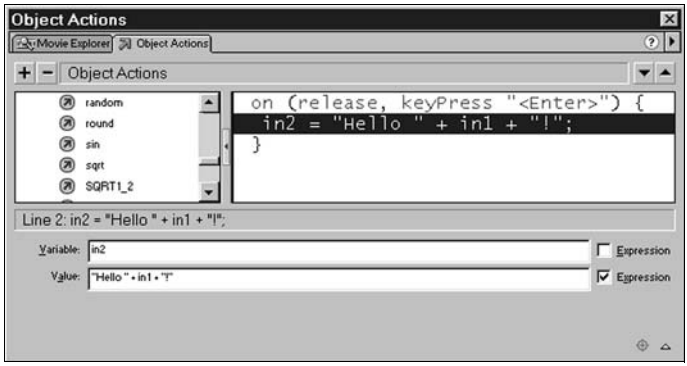


Рис. 6.11. Вместо in2 = in1 теперь написано in2="Hello"+ in1+"!"

Примечание

Вам не нужно самим добавлять в конце выражения точку с запятой, Flash сделает это за вас. Если текст выделен красным цветом, и вы видите сообщение «trailing garbage found», это, вероятно, из-за лишней точки с запятой.

Когда ActionScript встречает такое выражение, он берет строковый литерал «Hello», добавляет к нему содержимое переменной `in1`, приписывает в конце строки строковый литерал «!», а результат помещает в переменную `in2`.

3. Протестируйте фильм и введите ваше имя. Flash поприветствует вас, как старый друг (рис. 6.12).

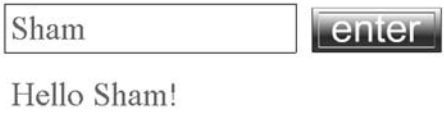


Рис. 6.12. Flash отвечает приветствием Шэму

Я не хочу преуменьшать наши успехи, но должен указать здесь на одну проблему. Еще не удалось научить Flash грамматическим правилам, и он не может проверить, имеет ли смысл то, что он вам говорит. Что бы вы ни набрали в окне ввода, Flash вернет это как часть ответа (рис. 6.13).

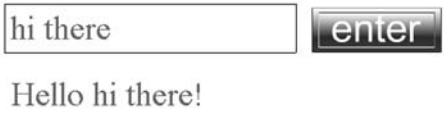


Рис. 6.13. Flash отвечает приветствием на любой вводимый текст

Здесь я вряд ли могу вам в чем-нибудь помочь. Я продемонстрировал, как можно при помощи строковых выражений придать общению с пользователем капельку теплоты и разумности, но вы теперь знаете и то, как легко срываются покровы иллюзии. Если вы попробуете применить подобный прием на своем веб-сайте, будьте осторожны. Кстати, это еще один аргумент в пользу того, что компьютеры пока еще не умнее нас!

Именование переменной

Теперь, когда вы составили себе первое и далеко не полное представление о перспективах усовершенствования веб-сайтов с помощью переменных, и прежде чем мы примемся за вопросы, связанные с хранением числовых значений, пришло время поосновательней поговорить об именах, которые мы даем переменным. Как вы увидите, это важно по двум причинам.

Во Flash можно давать переменным любые имена, *если только нет вероятности, что Flash спутает их с чем-нибудь еще*. Во Flash встроено множество имен и символов, имеющих важное значение для его правильного функционирования, поэтому нам нельзя использовать их для произвольных целей. Это так называемые **зарезервированные имена**. Среди них:

- **Команды**, или любые другие элементы языка ActionScript, которые могут быть приняты Flash за часть действия, включая символы `{ }`, `;` и `()`. Зарезервированными и недопустимыми в качестве имен переменных являются слова: `break`, `continue`, `delete`, `else`, `for`,

`function`, `if`, `in`, `new`, `on`, `return`, `this`, `typeof`, `var`, `void`, `while` и `with`. Очевидно, что нельзя начинать имя переменной с пары символов //, поскольку тогда оно превращается в комментарий. Как общее правило следует иметь в виду, что если в окне Actions имя переменной отображается черными символами, вероятно, все в порядке. Если оно отображается синим цветом или же выделяется красным фоном, когда вы выбираете строку, содержащую это имя, – значит, Flash принимает его за что-то другое.

- **Операторы**, такие как +, - и /. В противном случае Flash может попытаться интерпретировать имя переменной как, скажем, сумму двух переменных.
- **Пробелы**. Используйте вместо них подчеркивания _.
- Имена, начинающиеся с цифры, такие как `6y`. Если вам хочется, то напишите `y6`.

С учетом вышеизложенного, допустимыми именами будут:

- `cow`
- `menu6`
- `label7c`
- `off`

А недопустимыми:

- `on` Flash посчитает его частью действия `OnEvent`
- `label 7` из-за пробела внутри имени
- `6cmenu` поскольку имя начинается с цифры
- `my-toys` из-за наличия оператора минус (-) Flash подумает, что речь идет о разности двух переменных

Если вы будете следовать этим правилам, не тревожа Flash понапрасну, то и вам будет спокойнее. Наверняка вам не хочется тратить время, гадая, данные какого типа содержит переменная или откуда она вообще взялась, так что действуйте здесь – по мере сил – рационально. Старайтесь, чтобы имена переменных что-нибудь говорили об их содержимом. К примеру, если у меня имеются три переменные для хранения трех упоминавшихся прежде сумм денег, я мог бы назвать их `dayMoney`, `teaMoney` и `beerMoney`. Такой подход существенно облегчит отладку программ. С ростом опыта вы заметите, что иногда оказывается необходимым использование одноименных переменных в разных местах. Если только вам вполне ясно, зачем это понадобилось, все будет хорошо.

Необходимо также знать о некоторых странностях Flash в отношении к регистру. Flash различает строчные и заглавные буквы в названиях действий, так что `gotoandplay(1)` не сработает – Flash хочет, чтобы вы писали `gotoAndPlay(1)`. И в то же самое время он безразличен

к регистру в именах переменных, то есть big и BIG будут им рассматриваться как одна и та же переменная. Не пожалейте времени на то, чтобы это запомнить!

Наконец, путь – это часть имени переменной, так что если у вас на временной диаграмме экземпляра клипа darkcastle имеется переменная igor, полное имя переменной будет _root.darkcastle.igor.

Одновременно и без каких-либо проблем вы можете завести переменную с тем же именем igor на диаграмме экземпляра клипа transylvania. Ее полное имя будет _root.transylvania.igor.

Переменные могут быть **локализованы** на определенной временной диаграмме и существовать только на ней, так что одинаковые имена переменных на разных путях различаются, могут содержать отличные друг от друга значения и даже значения разных типов. Будьте уверены, Flash разберется в том, что это два разных igor, но вам надо постараться самим не забыть о разнице!

Хранение чисел

В нашей жизни числа используются не только для того, чтобы мучить нас на уроках математики и указывать цены в супермаркетах. Номер на кресле в кинотеатре означает не *стоимость* стула, но его *положение*. ISBN этой книги не представляет собой ее цену или положение, но является *ссылкой* или *указателем* на книгу.

В гораздо большей степени, чем строки, числа допускают над собой операции: их можно вычитать, складывать, умножать, делить друг на друга и так далее. Сохраняя числа в переменных, вы имеете в своем распоряжении все эти возможности.

Вернемся к примеру с местами в кинотеатре. Если мы сохраним в переменной seatstaken число купленных билетов, тогда при продаже следующего билета число купленных билетов всегда можно будет записать как seatstaken+1, вне зависимости от того, каково значение этой переменной в данный момент.

Взгляните на следующую строку:

```
seatstaken = seatstaken +1;
```

Она гласит: «Добавьте единицу к текущему значению seatstaken и сохраните результат снова в seatstaken». Всякий раз, когда ActionScript встретит такую строчку, значение переменной будет увеличено на единицу. Помните – переменные обеспечивают ActionScript памятью.

Попробуем реализовать кое-что из вышеописанного на основе недавно нами созданного и сохраненного FLA (если у вас что-то не получилось или вы не сохранили его, воспользуйтесь chapter06_01.fla из загружаемого архива). В следующем упражнении мы собираемся предложить

пользователю ввести какие-то числа, а затем совершить с ними некоторые вычисления.

В предыдущем примере значения, которые принимала переменная in1, были произвольными строками, то есть они могли содержать *не только* цифры от 0 до 9 и десятичную точку, с помощью которых обычно отображаются числа. Прежде чем начать использовать in1 в числовых выражениях, мы должны предпринять некоторые шаги, чтобы воспрепятствовать пользователю вводить символы, способные помешать нашей работе.

Выполнение числовых операций

1. Выделите поле ввода и вызовите панель Text Options. В ее нижней части вы видите несколько кнопок с изображениями символов. Если ни одна из кнопок не выбрана, Flash будет принимать все символы, которые пользователь может набрать. Если же какая-нибудь из них выбрана, Flash становится разборчивей. Выберите кнопку, помеченную цифрами 123, как это показано на рис. 6.14. Теперь мы не будем допускать в наше окно ввода ничего кроме цифр. Чтобы получать также и десятичные дроби, мы должны допустить ввод десятичной точки. На этот случай справа от нижних кнопок панели имеется поле ввода, в котором можно указывать дополнительные допустимые символы. Щелкните по нему и введите точку.

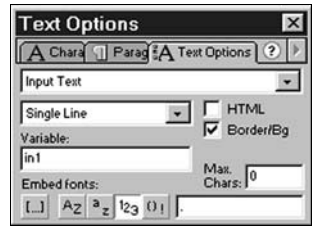


Рис. 6.14. Настройка поля ввода на прием числовой информации

2. Протестируйте фильм. Вы обнаружите, что теперь невозможно ввести в поле ввода ничего, кроме десятичных чисел, а это шаг в правильном направлении (рис. 6.15).

Рис. 6.15. Возможен ввод только цифр

3. В рабочей области выделите кнопку enter и вызовите окно Actions. Поскольку теперь мы уверены, что без опасений можем трактовать значение in1 как число, то попробуем присвоить in2 результат арифметической операции с in1:

```
in2 = 2*in1;
```

Примечание

В языках программирования звездочка * часто используется для обозначения умножения вместо традиционного крестика ×, который легко спутать с буквой «x».

Заменим вторую строку нашего сценария этой новой командой, поместив в поле Value `2*in1` вместо `in1` (рис. 6.16).

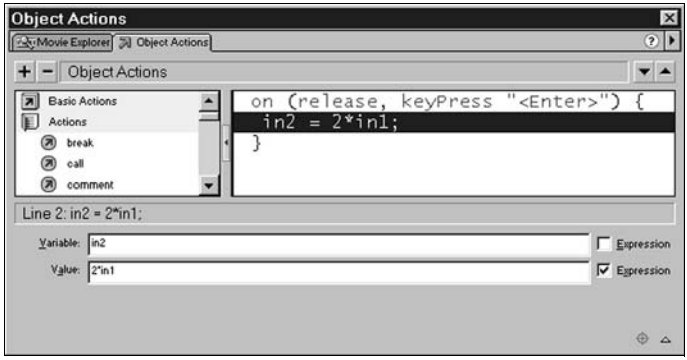


Рис. 6.16. Новая команда – удвоение введенного числа

Протестируем фильм. Наш вывод теперь представляет собой результат вычислений над введенным числом – другими словами, значение числового выражения (рис. 6.17).

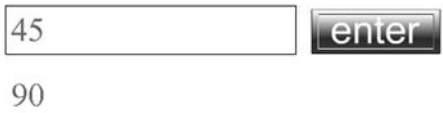


Рис. 6.17. Результат умножения на 2

Если хотите, можете попробовать изменить числовое выражение так, чтобы оно выдавало следующую информацию:

`in2=in1*in1` – для квадрата введенного числа

`in2=1/in1` – для обратного к введенному числу

Дальнейшие применения числовых выражений

Числовые выражения во Flash способны на гораздо большее, чем суммирование счетов. Например, можно использовать результаты вычислений для управления положением или ориентацией экземпляра, что позволяет создавать весьма сложные движения. Представьте себе выражение, указывающее кораблю пришельцев, куда ему отойти, когда Бларда в него стреляет!

Мы можем также применять числовые выражения для переходов от кадра к кадру, основываясь на *динамических* значениях, а не только на фиксированных, которые мы до сих пор использовали, что позволяет управлять протеканием фильма на гораздо более высоком уровне.

Например, «подбив» инопланетянина, мы можем перейти к изображению взрыва (рис. 6.18).

Рис. 6.18. Использование числовых выражений для перехода от кадра к кадру

Кое-что из обсуждавшихся нами предметов может показаться далеким от порхающих по экрану инопланетных захватчиков или эффектных и занятных динамических сайтов – но мы изучаем механику, которая заставляет вращаться мир ActionScript. В следующем разделе мы продолжим эту работу.

Логические значения

Покупая пакет печенья, я поступаю так потому, что оно выглядит вкусным, а я не сижу на диете. Это очень просто и ясно устроенное решение – если выполняются два условия, я покупаю печенье. Обычно, однако, мы не склонны поступать столь прямолинейно – мы основываемся, хотя бы отчасти, на сравнении, на нашем настроении и предрасположенностях. Иногда мы основываем свои решения на обстоятельствах, как будто бы не имеющих никакого отношения к делу: «Я не хочу с ним разговаривать, потому что мне не нравится цвет его рубашки». И наши решения имеют более сложную форму, чем просто «буду» или «не буду».

Компьютеры, однако, привержены к более однозначным решениям, что очень успокаивает, если я доверяю им отслеживать состояние моего банковского счета и операции по нему. Мне бы не хотелось, чтобы остаток на моем счете уменьшился вдвое только оттого, что у компьютера сегодня черный день или оттого, что я явился в банк в сиреневой рубашке. Компьютеры основывают свои решения на суждениях «да» и «нет» (или «истинно» – «ложно»). Каждое их решение в конкретный момент может иметь ровно одно из двух этих значений, хотя они могут принимать поочередно множество решений «да» или «нет», в результате чего образуются более сложные логические построения.

Предположим, к примеру, что нам хочется знать о некотором числе, больше оно 100 или меньше. Это может быть цена акции на бирже – если цена меньше, мы задумываемся, не прикупить ли еще, а если больше, то не пора ли продавать. В зависимости от того, по какую сторону от 100 находится цена, мы будем совершать одну из двух последовательностей действий. В другом случае это могла бы быть характеристика ситуации в игре про вторжение марсиан, когда злые пришельцы опускаются по экрану, пытаясь приземлиться. Когда расстояние от какого-либо из захватчиков до верха экрана превышает 100, мы знаем, что он достиг цели, и наш игрок проиграл.

Возьмем за основу первый пример и допустим, что цена интересующих нас акций хранится в переменной `price`. Теперь взгляните на следующую строку кода:

```
buy = price < 100;
```

Раньше, когда мы видели подобное, выражение по правую сторону от знака равенства оценивалось, а результат сохранялся в переменной, имя которой находилось слева. Но *что есть* значение выражения справа? Вспомнив заглавие этого раздела, вы не удивитесь, узнав, что это **логическое значение**, то есть что `buy` получит значение `true`, если `price` меньше 100, и `false`, если `price` имеет любое другое значение. `True` и `false` – это единственные возможные логические значения, и только они могут быть значениями выражения в правой части равенства.

Примечание

Заметьте, что `true` и `false` здесь – это не строки и не имена переменных. Это просто единственно возможные логические значения.

До тех пор пока в следующей главе мы не начнем рассматривать условные операторы, которые действительно позволят нам совершать различные действия в зависимости от логических значений, мы ограничены в возможностях их использования. Тем не менее, вы можете убедиться в том, что я говорил правду, если попробуете составить с ними чуть более сложные выражения.

Экспериментируем с логическими значениями

1. Еще раз модифицируем наш фильм. Загрузите chapter06_01.fla (файл, с которым вы работали в начале этой главы) и убедитесь, что в нем отсутствуют те незначительные вариации, которые вы внесли в него после того, как впервые сохранили. Выделите поле ввода и в панели Text Options измените установки (рис. 6.19) таким образом, чтобы мы снова могли вводить только числа, то

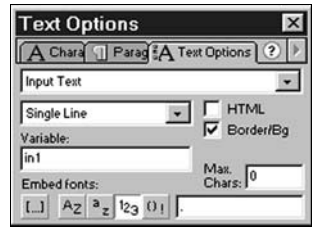


Рис. 6.19. Аналог рис. 6.14

есть выберите кнопку 123 и поставьте точку в правом нижнем поле ввода.

2. Выделите теперь кнопку enter и в окне Actions выберите строку:

```
in2 = in1;
```

И измените значение в поле Value на:

```
in2 = (in1<100);
```

Рис. 6.20. Присваиваем in2 логическое значение

3. Запустите фильм. Flash увидит число, которое вы введете, и сравнит его с 100. Если оно окажется меньше, то утверждение `(in1<100)` истинно и вы получите `true`. Если оно больше или равно 100, тогда `(in1<100)` ложно и вы получите `false` (рис. 6.21).

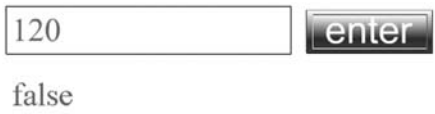


Рис. 6.21. Отображенное логическое значение

Если понадобится, для создания более сложных проверок вы сможете использовать **логические операторы**. Этим мы как раз и займемся. Рассмотрим наш прежний пример с ценой на акции. Если вы готовы покупать при цене меньше 100, вы, возможно, захотите отказаться от покупки, если цена опустится ниже 80, поскольку такое резкое падение может свидетельствовать о неблагоприятном состоянии рынка. Условие покупки акций можно записать словами так:

«Цена меньше 100 и больше 80».

Приводя это выражение к обозначениям, более близким для ActionScript, получим:

```
(in1<100) AND (in1>80)
```

Для обозначения AND Flash использует пару амперсандов (&&), которые можно найти в левой панели в книжке Operators. В действительности Flash допускает и более понятное обозначение AND, которое, однако, сохранено только для обеспечения совместимости с Flash 4 – это обозначение устарело, и его следует избегать.

4. Находясь на второй строке сценария кнопки, поместите текстовый курсор в поле Value в конец строки. Щелкните дважды на операторе && в списке Operators. Теперь у вас получится:

   ```
   in2 = (in1<100) &&
   ```

   И Flash выдаст сообщение о синтаксической ошибке, поскольку за оператором должен следовать еще один терм. Допишите (in1>80) после оператора &&, чтобы получилось (рис. 6.22):

   ```
   in2 = (in1<100) && (in1>80);
   ```

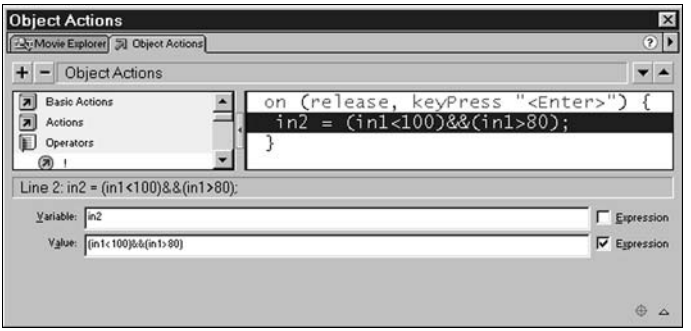


Рис. 6.22. Логический оператор &&

5. Запустите фильм. Убедитесь, что вы получаете результат true, только если цена лежит в промежутке между 80 и 100.

Действуя подобным образом, вы можете строить очень сложные условия, включающие множество других операторов и термов, и в следующей главе мы будем использовать их в полную силу.

Объекты, методы и свойства в ActionScript

Прежде чем мы продолжим, я должен познакомить вас еще кое с чем – или, лучше сказать, кое-что вам напомнить. Возможно, от вашего внимания не ускользнуло, что сохранение значений в переменных имеет

сходство с тем, что мы делали, определяя x-координату движущегося шарика:

```
ball._x = _xmouse;
```

Здесь, разумеется, нет полного совпадения. Те «именованные экземпляры» (как мы их тогда называли), например ball, были на самом деле переменными, содержавшими **объекты**, и хотя мы займемся ими вплотную только в главе 9, поговорить о них мы можем уже сейчас.

Объекты

Если представлять себе строки, числа и логические значения как плитки молочного, черного и белого шоколада в мире переменных, то объекты представятся коробками шоколадных конфет, какие мы дарим на Рождество. Объекты дают возможность хранить целые наборы значений и еще кое-что впридачу. Они представляют собой зерно, из которого произросли многие из средств Flash 5, и мы уже отчасти познакомились с некоторыми возможностями, которые открывает нам их использование.

На настоящий момент из всех объектов вы лучше всего знакомы с клипами. Вы знаете, что это вполне определенный элемент, который применяется для вполне определенных целей, и что он обладает некоторым множеством **свойств** – числом кадров, положением и так далее, – относящихся к его внешнему виду и поведению. Эти свойства являются частью любого клипа, который вам когда-либо встретится.

Это отнюдь не единственное полезное качество объекта – в рукаве у него есть еще пара тузов. Объект «клип» у нас имел команду «stop», которую мы могли применять, например, так:

```
ball.stop();
```

Вы увидите еще множество разноименных команд и действий, но корректный термин, пригодный для их обозначения, – это **методы**, и их использование обычно приводит к тому, что с объектом что-то происходит. Начинающего никто не упрекнет, если он называет их просто командами, но употребление правильного термина неизменно производит хорошее впечатление. Умение применять методы во Flash жизненно важно для тех, кто хочет взобраться на вершины мастерства ActionScript.

Очень легко понять, что из себя представляет объект «клип», поскольку его свойства и методы относятся к его зримым проявлениям. Но не все объекты так наглядны – вообще говоря, они могут быть собранием любого множества свойств и методов, которые могут иметь, а могут и не иметь очевидного визуального отображения. К примеру, Flash 5 использует специальные объекты для обеспечения таких возможностей, как расширенный набор математических операций и управление звуком. Сейчас мы рассмотрим первый из них.

Объект Math

Объект Math доступен в любой части программы ActionScript, но он несколько необычен, поскольку обладает только методами – у него вообще нет свойств! На деле этот объект представляет собой «библиотеку» команд, которыми вы можете пользоваться, когда бы и где бы они вам ни понадобились. Посмотрим, как он работает.

Извлечение квадратного корня числа

1. Вернемся к нашему незабвенному фильму, выделим кнопку enter, откроем окно Actions и очистим поле Value. Откроем в левой панели список Objects. Вы увидите еще множество списков действий, один из которых и есть Math. Выберите в нем sqrt, и, поскольку поле Value уже было активизировано, вы увидите следующее (рис. 6.23).

Рис. 6.23. Выбор метода sqrt

2. В поле Value появилась строка `Math.sqrt (number)`, и от нас требуется заменить `number` на переменную, с которой мы намерены оперировать, то есть на `in1`. Сделав это, запустите фильм. Как вы, вероятно, догадались, Flash теперь выдает нам квадратный корень из введенного числа (рис. 6.24). Если вы посмотрите на другие методы объекта Math, то увидите, что они охватывают тригонометрические функции, логарифмические и некоторые другие. Когда вы продвинетесь в трехмерном программировании в реальном времени, эти функции вам очень пригодятся.

Рис. 6.24. Извлечение корня из числа

Объект String

И наконец, собравшись с духом, я решаюсь открыть вам великую и ужасную тайну: во Flash 5 *все* переменные содержат объекты, даже если они, по-видимому, хранят только простые значения одного из уже знакомых нам типов. Числовые значения, например, являются объектами с единственным свойством – их значением. Как видно из предыдущих упражнений, эта кажущаяся сложность переменных не мешает нам использовать их просто, но в иных случаях, при необходимости, она позволяет взять от них больше.

И далее приходится признать, что все созданные вами строки, так же как и клипы, имеют набор свойств и методов. Вы можете увидеть все эти свойства и методы, раскрыв список Objects ▸ String (рис. 6.25).

Рис. 6.25. Свойства и методы строковых значений

Если вы задержите курсор на одном из элементов списка, Flash при помощи всплывающих подсказок сообщит вам о нем дополнительную информацию. Мы не будем сейчас пытаться охватить весь этот список (хотя по ходу чтения этой книги вы будете встречаться то с одним, то с другим его элементом), но познакомимся с одним свойством строк и двумя методами обращения с ними. Взгляните вот на это:

```
in2 = in1.length;
```

Здесь `in1` содержит строку, и эта строка кода использует свойство `length` строкового объекта, чтобы сохранить в `in2` длину значения `in1`. Как вы, вероятно, можете себе представить, знать это свойство строки очень важно, если понадобится разбить строку, проанализировать ее или совершить с ней что-нибудь еще такое же умное. Посмотрим, как мы могли бы проделать это во Flash.

Определение длины строки

1. Начнем с того, что наберем `in1` в поле Value окна Object Actions нашей единственной кнопки. Перейдем затем к Object ▸ String в инструментальном окне и найдем свойство length. Прежде чем дважды щелкнуть на нем, перейдите в поле Value и поместите курсор сразу после текста `in1`. Теперь дважды щелкните на методе (рис. 6.26).
2. Чтобы протестировать фильм, введите какой-нибудь текст и нажмите кнопку enter. Отметьте, что длина строки вычислена с учетом пробелов, так как (по крайней мере с точки зрения Flash) пробел – это символ, не хуже любого другого (рис. 6.27).

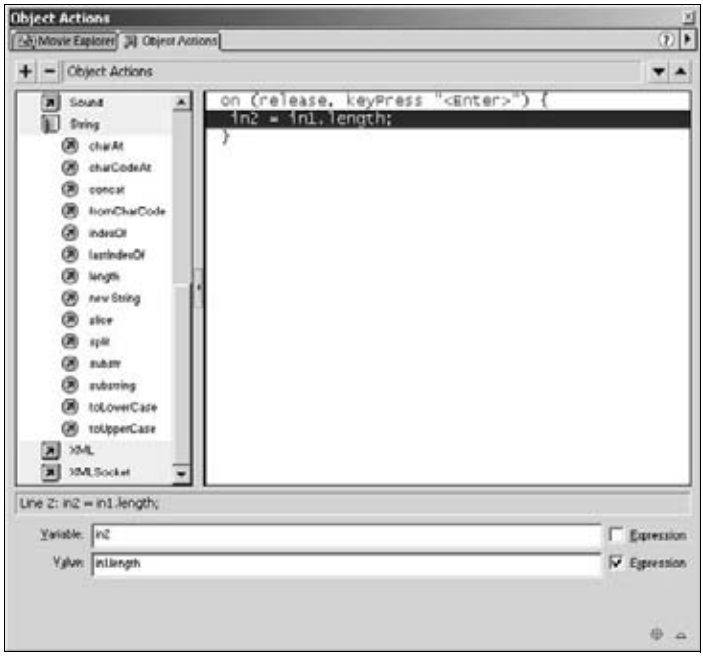


Рис. 6.26. Выбор свойства length

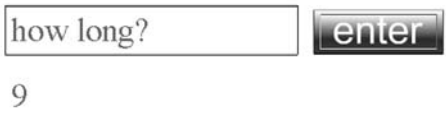


Рис. 6.27. Подсчет длины строки

Вот еще один пример. Часто бывает интересно, содержит ли ввод какой-либо определенный символ. Если вы задаете пользователю вопрос, требующий ответа «да» или «нет» («yes» или «no»), то можете получить в ответ «yes», «y» или даже « y» (с пробелом перед буквой), равно как и все эти варианты, написанные прописными буквами. Чтобы ваши программы работали устойчиво, всегда нужно предполагать, что пользователь что-нибудь сделает неправильно, так что вы должны быть уверены, что ваш код со всей снисходительностью и терпимостью отнесется к самому глупому ответу.

В следующем упражнении мы будем проверять, содержит ли строка латинскую букву «y». Нам не интересно, содержит ли строка какие-либо другие символы, так как необходимой и достаточной информацией, свидетельствующей о положительном ответе на вопрос, мы будем считать наличие в ответе буквы «y».

Проверка содержимого строки

1. Прежде всего добьемся того, чтобы нам нужно было искать только строчную букву «y», а не обе – строчную и прописную. Для этого понадобится промежуточный шаг. Мы создадим новую переменную с именем `temp`.

   В том же окне Object Actions из прошлого примера вставьте новое действие set variable из списка Actions. Поместите его между двумя уже существующими строками кода, пользуясь стрелками «вверх» и «вниз» в правом верхнем углу окна.

2. Введите в поле Variable `temp`. Щелкните в поле Value (убедитесь сначала, что соответствующая опция Expression отмечена флажком). Выберите теперь метод `toLowerCase` из меню Objects ▸ String, щелкните еще раз в самом начале поля Value и введите имя переменной, содержащей строку: `in1`.

3. Теперь пора искать нашу букву «y». Выделите старую строку `in2=in1;` и замените содержимое поля Value:

   ```
   temp.indexOf("y")
   ```

   Эта строка кода ищет «y» в содержимом переменной `temp` и выдает нам ее положение (или указатель) в строке. Окно Object Actions должно выглядеть теперь так, как на рис. 6.28.

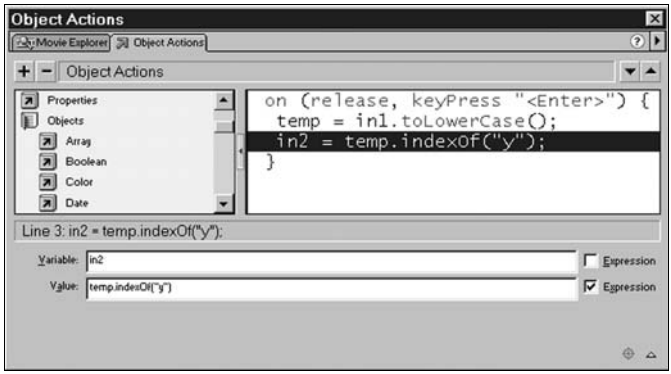


Рис. 6.28. Строка кода для поиска «y» в temp

4. Наименьшее возможное положение «y» – это 0, а вовсе не 1. Если Flash не находит «y», он сдается и возвращает -1. Протестируйте фильм. Попробуйте вводить разный текст. Если вы будете вводить «Y», «y», »yes» или «Yes», отображаемое число будет положительным (или по меньшей мере равно 0).

Как видите, рассматривая строки как объекты, мы получили возможность использовать их методы для совершения с ними различных действий помимо простого сложения. Мы смогли применить к строкам специализированные процедуры, свойственные именно этому типу значений.

Резюме

Вы приобрели большие познания в области переменных и значений. В этой главе вы не создали какого-либо яркого FLA, но зато получили надежную опору для перехода к главе 7, где все вновь изученные предметы позволят вам сделать кое-что эффективное и полезное. Вот чего вы добились, продравшись через эту главу.

В ваших собственных проектах попробуйте работать с теми типами значений, которые годятся для решения ваших проблем. К примеру, мы все хотели бы беседовать с машинами на том языке, на котором мы общаемся друг с другом. Здесь пригодятся строки, так как они предоставляют Flash возможность дружественного общения с пользователем, хотя они и не годятся для сложных вычислений.

В этой главе мы узнали:

- Что из себя представляют переменные и как Flash их использует
- Как применять строковые значения для общения с пользователем
- Как применять числовые значения для вычислений
- Как при помощи логических значений сохранять результаты сравнений

Когда мы будем прорабатывать последующие главы этой книги, вы увидите, что множество эффектов, которых вы сможете добиться при помощи Flash, основаны на переменных. Мы будем использовать строковые значения при создании игры в виселицу и рассмотрим применение логических значений для организации процессов с принятием решений.

Мы мельком упомянули объекты как вместилища наборов значений, способные на большее, чем быть простой суммой своих частей. К объектам мы собираемся вернуться в главе 9, но уже теперь, кое-что о них зная, вы будете замечать их повсюду. Объект **sound**, например, получит детальное освещение в главе 8.

Сделайте небольшой перерыв, а затем переходите прямо к следующей главе, где мы вволю попользуемся переменными!