Министерство образования и науки Российской Федерации

Федеральное агентство по образованию

Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Д.Г. Николаев, Д.Г. Штенников

Web-программирование. Клиентский ActionScript

Учебное пособие

Санкт-Петербург

2005

УДК 681.3

Николаев Д.Г., Штенников Д.Г. Web-программирование. Клиентский ActionScript.
Учебное пособие. – СПб., 2005. - 136 с.

Рецензенты: Л.С. Лисицына, к.т.н., доцент, зав. каф. КОТ СПбГУ ИТМО,
А.А. Бобцов, к.т.н., доцент каф. СУиИ СПбГУ ИТМО

Учебное пособие подготовлено на кафедре «Компьютерные образовательные технологии» (КОТ) факультета ИТиП СПбГУИТМО и предназначено для студентов специальностей 230202 – «Информационные технологии в образовании» и 230201 – «Информационные системы и технологии», изучающих курс «Web-программирование», а также для разработчиков Интернет-ресурсов, слушателей курсов ДО для подготовки в рамках обучения по программам городского комитета по образованию, слушателей специализированных курсов. Пособие служит руководством к выполнению практических заданий. Практикум формирует набор практических навыков использования современных информационных технологий в сфере образовательной деятельности.

Печатается по решению УМС факультета ИТиП СПбГУ ИТМО, протокол № 2 от 27.09.2005.

Оглавление

Введение

Дисциплина «Web-программирование» относится к дисциплинам специализации подготовки дипломированных специалистов 230202 – «Информационные технологии в образовании» и 230201 – «Информационные системы и технологии». Данное пособие может быть использовано на курсах повышения квалификации учителей информатики и информационных технологий по программам соответствующего содержания.

Дисциплина базируется на знаниях, приобретенных при изучении дисциплин: «Информатика», «Технология программирования», «Объектно-ориентированное программирование» и «Информационные технологии».

Целью данного пособия является методическая поддержка:

- практических занятий по курсу «Web-программирование»;
- подготовки к аттестующим тестам по лекциям курса «Web-программирование».

Курс практических занятий по «Web-программированию» состоит из ряда практических заданий по программированию на **ActionScript** и разработке интернет-ресурсов.

Тестирование по лекционному курсу «Web-программирование» проводится в системе ДО ИТМО в рамках расписания практических занятий. Темы аттестующих тестов (все тесты – с оценкой):

1. «Основы языка ActionScript»,
2. «Объектно-ориентированное программирование на ActionScript»,
3. «Общие вопросы создания интернет-ресурсов на основе Flash».

Аттестация по курсу «Web-программирование» осуществляется в форме экзамена, допуском к которому является зачет по всем практическим заданиям и положительная оценка по аттестующим тестам.

Введение в ActionScript

Элементы AS

ActionScript – язык программирования, который соединяет промежуток между web-браузером и программой. Подобно всем языкам, ActionScript содержит множество различных элементов: слов пунктуации, структур – то есть те элементы, используя которые, вы сможете написать проект. При этом необходимо соблюдать определенные правида записи таких элементов, в противном случае вы не добьетесь желаемого результата. Данное пособие весьма кракое, поэтому, если у вас возникнет небходимость более подробного изучения программирования на AS, то желательно обратиться к

более подробному изданию, например к книгам, которых на прилавках магазинов расплодилось в немеренном количестве. Чтобы начать понимать как работает ActionScript, посмотрите на следующий сценарий, который содержит многие из основных элементов, составляющих типичный сценарий. Для начала, можно предположить, что этот сценарий присоединен к кнопке:

```
on (release) {
// установить стоимость
mugCost = 5.00;
// установить местный налоговый
taxPercent = .06;
// определить сумму налога
totalTax = mugCost * taxPercent;
// определить общую сумму сделки
totalCost = mugCost + totalTax;
// отобразить сообщение в текстовом поле
myTextBox.text = "The total cost of your transaction
is " + totalTax;
// перводим Муви Клип (Movie Clip – MC) с именем
cashRegister на 50 кадр и запускаем //процесс его
проигрывания
cashRegister.gotoAndPlay (50);
}
```

Хотя на первый взгляд это может напомнить английкий, далее вы поймете, что это язык программирования весьма и весьма простой для понимания, особенно есмли у вас уже имелся опыт работы с любым языком программирования ранее.

Примечание

Существуют дополнительные элементы сценария (например, объекты, функции, циклы, свойства, и методы), примеры работы которых, будут рассмотрены далее в примерах в посмобии.

События

События происходят в течение воспроизведения проекта и вызывают выполнение того или иного сценария. В типовом сценарии это случай, который вызывает определенную последовательность действий, которую также можно назвать сценарием. Напрмер слово *release* показывает, что в случае, когда кнопка, к которой этот сценарий прикреплен, будет нажата и отпущена, то в таком случае будет выполнен сенарий. Каждый сценарий вызван событием.

Действия (Actions)

Действия формируют основу сценария и, как полагают, являются любой строкой кода, которая заставляет программу совершать действия внутри MC, установливать, создавать, изменять{заменять}, загружать, или удалять.
Далее несколько примеров действий типового сценария:

```
mugCost = 5.00;
cashRegister.gotoAndPlay (50);
```

Если, действия включают болшее чем две строки внутри сценария, то они должны быть аключены в фигурнве скобоки ({}) и строчки должны отделяьбся друг от друга точками с запятой (;).

Операторы

Операторы включают множество символов (=, <,>, +,-, *, **, и т.д.) и используются, чтобы произвести действие с переменными внутри проекта. Например:

```
taxPercent = .06;
```

присваивает значение .06 переменной, названной taxPercent

```
amountA <amountB
```

спрашивает, если amountA - меньше чем amountB

```
value1 * 500
```

умножает value1 на 500

Ключевые слова

Это слова, зарезервированные для различных целей в пределах синтаксиса ActionScript. Такие слова, не могут использоваться как имя переменной, функции, или имени метки. Например, слово *on* - ключевое слово и может только использоваться в сценарии, чтобы обозначить событие, которое вызывает сценарий. Попытка использовать ключевые слова в ваших сценариях для иных целей, чем они предназначались авторами языка, приведет к ошибкам и неработоспособности проекта в целом.

Данные

Динамический сценарий в течение своего выполнения почти всегда создает, использует, или обновляет различные данные. Переменные - самые общие части динамических данных, которые могут найтись в сценариях и представляют данные, которым дали уникальные названия. Как только переменная была создана и получила значение, к ее значению можно обратиться внутри сценария просто, используя ето название.
В нашем типовом сценарии, приведенном выше, переменной mugCost присвоено значение 5.00. Позже в сценарии, название етойже переменной используется, чтобы обратиться к значению, которое она содержит. Переменные также нельзя называть с использованием кирилицы.

Изогнутые фигурные скобки

То, что падает между открытием и закрытием изогнутых фигурных скобок показывает действие или последовательность действий, которые должны быть исполнить одно за другим после вызова этого сценария. Например:

```
on(release)
{
// установить стоимость света
mugCost = 5.00;
// установить местный налог
taxPercent = .06;
}
```

Точки с запятой

Появляясь в конце большинства строк сценариев (как правило в пределах изогнутых фигурных скобок), точки с запятой используются, чтобы разделить различные действия, которые, возможно, должны быть выполнены как результат одного события. Следующий пример обозначает шесть отдельных действий, отделенных точками с запятой:

```
mugCost = 5.00;
taxPercent = .06;
totalTax = mugCost * taxPercent;
totalCost = mugCost + totalTax;
myTextBox = "общая стоимость вашей сделки - " +
totalTax;
cashRegister.gotoAndPlay (50);
```

Точечный синтаксис (dot)

Точки (.) используются в пределах сценариев несколькими способами: Каждый должен обозначить путь к определенному MC или его графику времени. Например, _root.usa.indiana.bloomington направляет на movie clip, находящийся на главном (_root) графике времени (т.е. на сцене) с именем "usa", которыq содержит movie clip, названный "indiana", который содержит movie clip, названный "bloomington". Т.е. как в стихотворении про дом который построил Джек, AS поддерживает высокую степень вложенности объектов одного в другой и обратиться к ним легче всего используя дот синтаксис. Не стоит забывать, что ранее в версиях Flash до 5-го использовался slash (слеш) синтаксис, который возможно прменять и на сегодняшний день. Особенно хотелось бы упомянуть свойство MC _droptarget , скоторым вы столнетесь позднее. В версии Flash MX, это всойство возвращяет путь с использоыванием именно слеш синтаксиса, в более поздних версиях адрес возвращается уже с использованием более продвинутого дот синтаксиса.

Поскольку ActionScript – объектно-ориентированный язык, большое количество задач решается путем изменения характеристик объекта. При изменении свойств или при вызове метода, точки используются, чтобы отделить название объекта от свойства или метода. Например, зная, что movie clip – объект, ему можно панели свойств (она же инспектор свойств, она же панель Properties) задать имя экземпляра (Instance name), например wheel, для последуещего обращения к данному MC и установить свойство вращения (_rotation) для него:

```
wheel._rotation = 90;
```

Обратите внимание, точка отделяет название объекта от устанавливаемого свойства. Для тогоже MC точка используется для отделения имени MC от метода запуска проигрыывания внутри MC, например:

```
wheel.play()
```

Здесь, точка отделяет название объекта от вызванного метода play().

Круглые скобки

Круглые скобки используются различными способами в ActionScript. Главным образом, сценарии использует их, чтобы установить определенное количество параметров, которые будет использованы в том или ином действии в процессе его выполнения. Помотрите на последнюю строку типового сценария, который говорит экземпляру MC с именем cashRegister перейти на 50 кадр и начать проигрывание gotoAndPlay (50):

```
cashRegister.gotoAndPlay(50);
```

Если значение в пределах круглых скобок будет изменено с 50 на 20, действие все еще исполняет ту же самую задачу (переместит график времени к указанному номеру кадра{фрейма}); только сделает это согласно новому значению.

Кавычки

Кавычки используются, чтобы обозначить текстовые данные (текстовые или строковые переменные) внутри сценария. они обеспечивают единственное средство для сценария, чтобы различить команды или вещественные перемнные и строковые значения. Например, Derek (без кавычек) показывает имя переменной. "Derek", показывает фактическое слово Derek.

Комментарии

Коментарии - строки в сценарии, которым предшествуют две метки косой черты (//), или открытие /* и закрытие */. При выполнении сценария, программа игнорирует строки, содержащие комментарии. Вместо этого, они являются средством описаний и примечания внутри сценария о том, что делается в той или иной строке и в том или ином блоке программы. Используя комментарии, вы можете вспомнить какие мысли у вас витали в голове спустя месяцы после того как сценарий был написан и вы все еще

желаете получать четкое представление о том, что вы делели или хотели сделать.

Сдвиг/Интервал

Сдвиг строчек кода не является необходимым элдементом программы, более того, он совершается автоматически строкой ниже открытия фигурных скобок и убирается также автоматически при закрытии оных, но он является хорошим тоном при написании сценария для того чтобы и вам и другим было понятно где какой структцрный блок программы находится. Например:

```
{
mugCost = 5.00;
}
Выполнит то же самое как:
{
mugCost = 5.00;
}
```

Выравнивая код, вы облегчаете возможность читать. Хорошее правило состоит в том, чтобы выровнять что-нибудь в пределах изогнутых фигурных скобок, чтобы указать, что код в пределах тех фигурных скобок представляет блок кода, или кусок кода, который должен быть выполнен единовременно. (Программная особенность Flash MX заботится о большинстве из этого для вас). Вы можете вложить блоки кода в пределах другого блока кода, которое станет более ясным.

Главным образом, незаполненное пространство игнорируется в пределах сценария. Помотрите на следующую строку сценария:

```
totalCost = mugCost + totalTax;
```

Это выполнится тем же самым способом как:

```
totalCost=mugCost+totalTax;
```

В то время как некоторые программисты чувствуют, что дополнительное незаполненное пространство делает их код более легким для чтения, другие полагают, что это замедляет их. Главным образом, этот выбор за вами. Однако, существует несколько несколько исключений, достаточно важных чтобы знвть об этом: Имена переменной не могут содержать пробела, и вы не можете поместить пробел между именм объекта и связанным свойством/методом. Например:

```
myObject.propertyName ;
```

Это – не:

```
myObject. PropertyName;
```

Создание первого скрипта

Основы написания самого простого проекта важны при освоении любого языка программирования, поэтому первый пример будет посвящен созданию одного из самых простых решений на основе ActionScript.

В этом уроке, вы:

- Узнаете об элементах сценария
- Напишите первый сценарий
- Проверите сценарий на работоспособность и внесете необходимые коррективы

1. Откройте Windows Notepad или Apple Simple Text для создания нового текстового файла, и запишите следующее:

```
&electricBill=60.
```

2. Назовите текстовый файл Electric_Bill.txt и сохраните его в ту же папку, что и файлы с уроком, например Lesson01/ Complete.
3. Откройте electricbill1.fla в папке Lesson01/Assets.
4. Разверните (или откройте) Property inspector (Инспектор свойств, панель свойств, панель Properties), выберите текстовое поле в верхнем левом углу сцены (Scene), которая находится сразу под текстом "*Amount you owe is:*". В Property inspector, задайте этому текстовому полю имя экземпляра (имя копии, имя объекта, имя образца, Instance name Name) – owed.
5. Не закрывая Property inspector, выберите текстовое поле (text field), находящееся под текстом "*Amount you would like to pay is:*". В Property inspector дайте текстовому полю instance name paid.
6. Пока Property inspector еще открыт, выберите текстовое поле внизу сцены, select the text field at the bottom of the scene, под кнопкой Switch. В Property inspector, дайте текстовому полю instance name - message.
7. Выберите movie clip (ролик, мувик, муви клип), изображающий лампочку, и задайте ему имя light.
8. Откройте панель Actions (панель действий), переведите ее в экспертную моду (Expert Mode) для того, чтобы вводить сценарии по своему усмотрению, а не мучиться со стандартными панелями ввода сценариев, выберите первый кадр (он же первый ключевой кадр) в слое (layer) Actions (действия) и введите следующий скрипт:

```
loadVariablesNum ("Electric_Bill.txt", 0);
```

9. Добавьте ключевой кадр (keyframe, КК) в 10 кадр слоя Actions. Затем, в панели Actions добавьте следующий скрипт в этот кк:

```
owed.text = electricBill;
stop ();
```

10. С открытой панелью Actions выберите кнопку в виде выключателя и введите следующий скрипт:

```
on (press) {
amountPaid = Number(paid.text);
amountOwed = Number(owed.text);
}
```

11. Пока панель Actions еще отрыта, добавьте следующие строки скрипта
    за скриптом, написанным в предыдущем шаге. Добавьте его внутри фигурных скобок ({}) сразу после строчки, содержащей `amountOwed = Number(owed.text);`:

```
if (amountPaid < amountOwed) {
difference = amountOwed - amountPaid;
light.gotoAndStop ("Off");
message.text = "You underpaid your bill by " +
difference + " dollars.";
} else if (amountPaid > amountOwed) {
difference = amountOwed - amountPaid;
light.gotoAndStop ("Hot");
message.text = "You overpaid your bill by " +
difference + " dollars.";
} else {
light.gotoAndStop ("On");
message.text = "You have paid your bill in full.";
}
```

12. При открытой панели Actions и пока еще выбрана кнопка – выключатель света, введите следующий скрипт:

```
on (release) {
light.gotoAndStop ("Off");
message.text = "";
}
```

13. Сохраните этот файл как electricBill2.fla.
14. Протестируйте файл Control > Test Movie.

Тестирование созданного проекта

1. Если вы уже закрыли ваш первый проект под названием electricBill2.fla, то вам придется его вновь открыть.
2. Выберите в меню Flash Control > Test Movie.
3. Введите различные значения в текстовое поле amountPiad, находящееся рядом с надписью "Amount you would like to pay:", после чего нажимайте кнопку включения электричества, чтобы отследить, как будет зажигаться лампочка.

Введите значение платежа менее 60, если вы введете значение, например 35, то после отработки скрипта возникнет сообщение о недоплате: "You underpaid your bill by 25 dollars," и лампочка будет находиться в выключенном состоянии (MC в метке кадра off).

Введите значение большее 60. Если вы введете величину равную 98, то вам будет выведено сообщение о переплате: "You have overpaid your bill by -38 dollars," и movie clip с именем light изобразит, что лампочка зажглась слишком ярко (frame labeled Hot). Сообщение выдает известие о переплате в -38 долларов, но на самом деле такого быть не должно.

Ведите значение равное 60. Если будет введено значение равное 60, то сообщение выдаст известие о том, что аппетиты энергетической компании удовлетворены: "You have paid your bill in full," и лампочка зажжется (movie clip с именем light перейдет в метку кадра on).

Удалите содержимое из текстового поля и снова нажмите на кнопку включения света, и, как ни странно, на экран будет выведено сообщение "You have paid your bill in full", говорящее о том, что все заплачено, а лампочка зажжется, это также является некорректным и требует устранения.

Ведите вместо цифр набор букв и снова, как и в предыдущем примере, загорится надпись о полной уплате и лампочка зажжется.

Таким образом, из процесса тестирования, вы узнали о том, что в программе существуют три ошибки (бага, bug).

4. Закройте окно предварительного просмотра и вернитесь в среду создания приложений. Выберите кнопку включения электричества и откройте панель Actions, после чего измените девятую строчку вашего кода (Line 9) с:

```
difference = amountOwed - amountPaid;
```

на

```
difference = amountPaid - amountOwed;
```

5. Когда кнопка включения электричества еще открыта, а панель Actions еще не закрыта, введите следующие дополнения в условие still `if`:

Добавления:

```
if (isNaN (amountPaid)) {
```

```
message.text = "What you entered is not a proper
dollar amount. Please try
again.";
}
```

Изменения в коде:

```
} else if (amountPaid < amountOwed) {
difference = amountOwed - amountPaid;
light.gotoAndStop ("Off");
message.text = "You underpaid your bill by " +
difference + " dollars.";
}
```

6. В меню выберите Control > Test Movie, введите различные значения оплаты за электричество и после этого нажмите кнопку включения света.
7. Закройте окно предварительного просмотра и вернитесь в среду создания приложений. Сохраните этот файл под именем electricBill3.fla.

Использование событий

Использование обработчиков событий

Наверное, всем известно, что любое действие вызывает некую реакцию. В мире, реальных вещей, события случаются вокруг нас обычно, как результат определенного действия. Мы даем команды, мы используем людей, все с единственной целью: вызвать ответ.
В программной среде, ответы, которые исходят из вывода, остановки, перемещения, ввода, паузы, и так далее вызваны обработчиками события. Благодаря этим обработчикам событий программисты получают возможность создания интерактивного проекта (проект будет откликаться на те или иные действия).

Функции обработчиков событий

Обработчики события управляют деятельностью вашего проекта, управляя, вызовом сценария. Они обеспечивают выполнение таким образом, чтобы это выполнилось только, по определенному событию (действию пользователя) и ни в каком ином случае. Каждый сценарий в проекте вызван пользователем, например нажатием кнопки мыши или нажатием клавиши на клавиатуре и т.д. Существует вариант, когда событие пользователем вызывается неявно – это случай когда график времени вашего проекта, достигает некоторого ключевого кадра.

В ActionScript, обработчики события (за исключением событий фрейма - кадра) обычно представляют первые строки в любом сценарии. Например:

```
When this happens (eventHandler) {
do this;
do this;
}
```

События фрейма происходят, в момент времени когда график времени достигает фрейма, который содержит сценарий. При размещении сценария на фрейме, вы не должны идентифицировать событие фрейма, чтобы вызвать сценарий, потому что достаточно графика времени, достигающего фрейма, чтобы заставить сценарий выполняться. Таким образом, если бы сценарий был помещен в фрейм, вышеупомянутый сценарий выглядел бы следующим образом:

```
do this;
do this;
```

Компьютерные программы позволяют пользователям выполнять задачи, позволяя им перетаскивать элементы на экране, изменять размеры окон, и

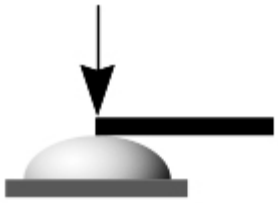
создавать артистические шедевры, используя "виртуальные" художественные инструментальные средства - все режимы взаимодействия определено таким образом, как программное обеспечение, было запрограммировано, чтобы иметь дело с различными событиями (нажатие клавиши мыши, движения мыши, ввод с клавиатуры, и так далее).

Использование событий от нажатия левой клавишей мыши

События от нажатия мыши управляют выполнением сценариев, когда мышь взаимодействует с кнопкой или movie clip'ом. Поскольку манипулятор мышь выполнен таким образом, чтобы стать продолжением руки пользователя, то и те или иные события, инициированные мышью, могут быть подобны нажатию руки на тот или иной предмет.
Если вы знакомы с Flash 5, и с выполнением событий от нажатия мыши, то вы знаете, что можете использовать их только с кнопками. Во Flash MX, однако, вы можете прикрепить события от нажатия мыши к movie clip'у.

Вступление в контакт, нажатие: on (press)

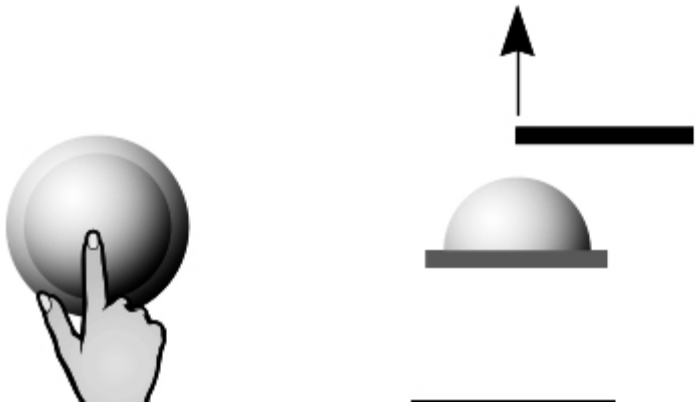
Когда вы касаетесь или нажимаете на некий предмет, вы ожидаете реакцию: человек, например, отвечает. Обработчик события on (press) создан таким образом, чтобы подражать касанию, захвату, так же как и результату любого из этих действий. Вы можете использовать этот обработчик события, чтобы вызвать сценарий, когда курсор мыши находится над кнопкой или экземпляром объекта movie clip, и кнопка мыши нажата.



Отпускание on (release)

Когда вы выпускаете предмет из рук или прекращаете вступать в контакт с ним, вы обычно заканчиваете, взаимодействуя с этим предметом. Данный обработчик события - подражает отпусканию предмета и является самым стандартным способом для пользователя заставлять movie clip выполнить необходимый сценарий. Дело в том, что еще с первых версий пакета Flash обработчик событий on(release) вставлялся по умолчанию, в случае если вы хотели записать действия на кнопку. Поэтому традиционно именно это событие часто используется на кнопках или экземпляра объектах movie clip,

чтобы вызвать действия. Вы можете использовать это, чтобы вызвать сценарий, когда кнопка мыши находится над объектом, нажата и отпущена.



Перемещение, моментальный снимок: on (releaseOutside)

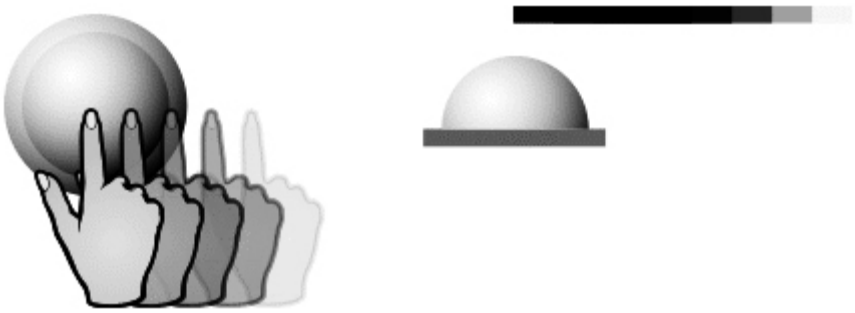
Вы можете использовать этот обработчик события, чтобы вызвать сценарий, когда ваш пользователь нажал кнопку проект или movie clip, и переместил мышь, чтобы подражать перемещению или моментальному снимку.

Управление клавиатуры: on (keyPress)

Вы можете использовать этот событие, чтобы вызвать сценарий, когда пользователь нажимает символ, номер, знак препинания, символ, стрелку, или возврат на один символ, Insert, Home, End, Page Up, или Page Down.

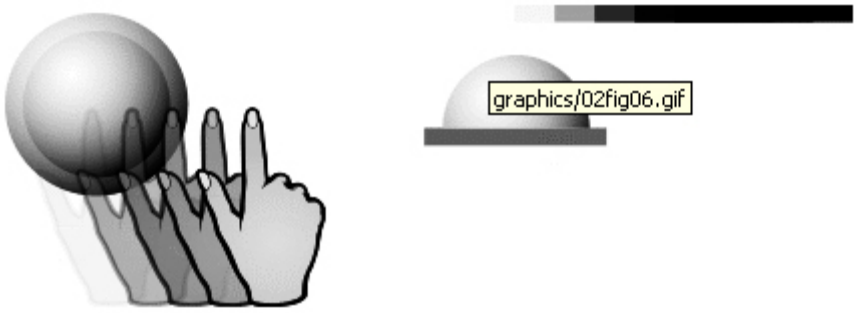
Не касание, наведение: on (rollOver)

Использование этого обработчика события сводится к тому, чтобы подражать способу, которым объекты затрагивают другие объекты, которые излучают теплоту, холод, свет, воздух, и так с лучевым источником, являющимся кнопкой или образцом movie clip. Вы можете также использовать этот обработчик события, чтобы отобразить информацию о том, что кнопка или экземпляр объекта movie clip сделают прежде, чем они нажаты. Вы используете этот обработчик события, чтобы вызвать сценарий, когда пользователь размещает курсор мыши на кнопке или образце movie clip.



Убирание курсора мыши от объекта: on (rollOut)

Вы можете использовать этот обработчик события, чтобы вызвать сценарий, когда пользователь перемещает курсор мыши от кнопки, или экземпляра объекта movie clip таким образом переставая с ним взаимодействовать.



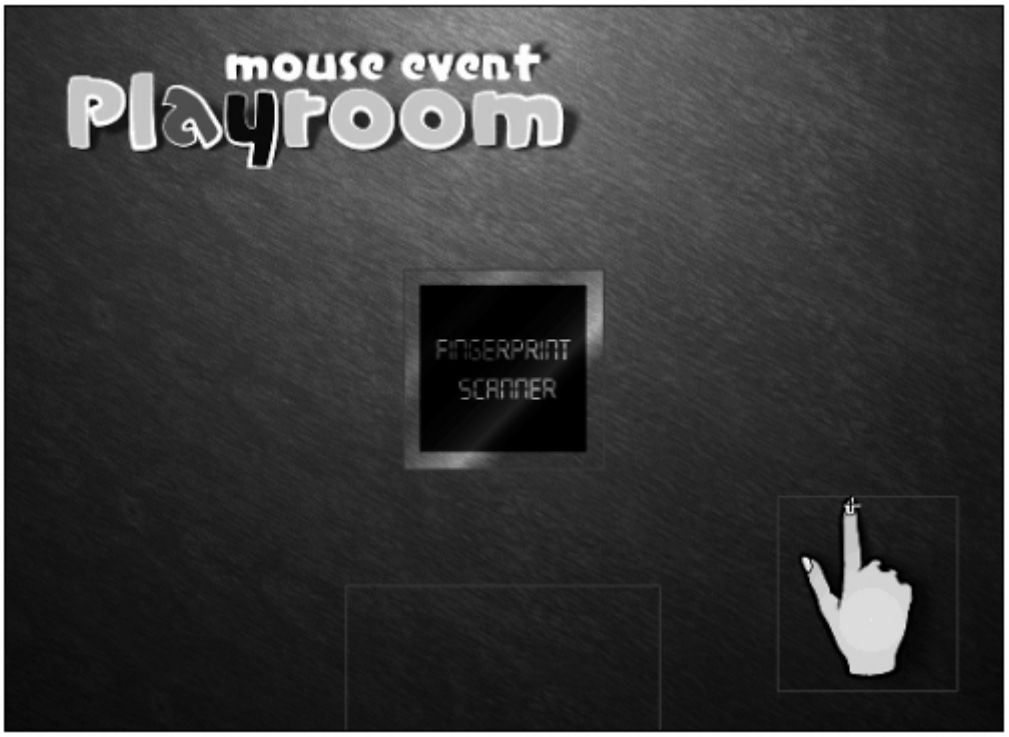
Ударение, протирание: on (dragOver)

Действие протирания стекла влечет за собой движение вдоль и поперек области назад и вперед по поверхности. Этот событие позволяет вам подражать этому типу деятельности в вашем проект, вызывая сценарий каждый раз, когда мышь «проезжается» по той же самой кнопке в проекте или экземпляру объекта movie clip, в то время как кнопка мыши остается нажатой.

Oops: on (dragOut)

Этот событие позволяет вам подражать тому, что может случиться, когда вы нажимаете или касаетесь кое-чего, но если вы должны были коснуться кого-то, затем убираете вашу руку. Вы можете использовать этот обработчик события, чтобы вызвать сценарий, когда пользователь размещает указатель поверх кнопки проект или экземпляра объекта movie clip, нажимает кнопку мыши, и перемещается.

Изучению этих событий будет посвящен второй пример



Использование событий Movie Clip (MC)

Когда экземпляр объекта movie clip с приложенными сценариями появляется на сцене, то сцена может получить новый внешний вид и функциональность в следствии наличия на этом movie clip событий. Эти события позволяют действиям быть вызванными, когда экземпляр объекта загружается или отгружается со сцены, когда мышь перемещается по сцене, и т.д.
Далее описываются различные события movie так же как и реальные аналогии для их использования. Обратите внимание, что вы можете использовать события movie со только сценариями, приложенными к экземплярам объекта movie clip.

Присутствие, загрузка: onClipEvent (load)

Этот обработчик события вызывает ответ, вызывая сценарий, когда экземпляр объекта movie clip загружается на сцену или в уже существующий экземпляр movie. Данный обработчик полезно использовать, чтобы инициализировать movie clip или начальные свойства этого movie.

Отсутствие, убирание со сцены: onClipEvent (unload)

Если экземпляр объекта movie clip может изменить параметры сцены, когда загружается, то, логично, параметры сцены могут измениться при его выгрузке со сцены или изнутри movie. Вы можете использовать этот обработчик события, чтобы вызвать сценарий, когда экземпляр объекта movie clip выходит из сцены.

Мощь, энергия. Дословно каждый кадр вашего приложения: onClipEvent (enterFrame)

Когда объекту сообщают энергию или импульс, это обычно показывает, что он начинает действовать непрерывно. Таким типичным бытовым примером являются часы; без зарядки (завода) они находятся неподвижными и бесполезными. Если вы обеспечиваете власть им, они начинают тикать, и вы способны использовать их, чтобы сказать время. Этот обработчик события используется, чтобы вызывать сценарий непрерывно, в том же самом оценивают ваши игры проект. Есть много мощных приложений для этого обработчика события, поскольку вы будете учиться всюду по этой книге.

Движение, перемещение мыши по экрану: onClipEvent (mouseMove)

Думайте об этом случае как о датчике движения в пределах вашего проекта. Если экземпляр объекта movie clip присутствует в сцене, к которому приложено это событие movie, то, прикрепленный набор действий может быть выполнен каждый раз, когда пользователь перемещает мышь. Это позволяет вам создавать деятельность, связанную с движением, например, способность обнаружить направление движения (направо, налево, вверх и вниз), текущую позицию мыши, и больше.

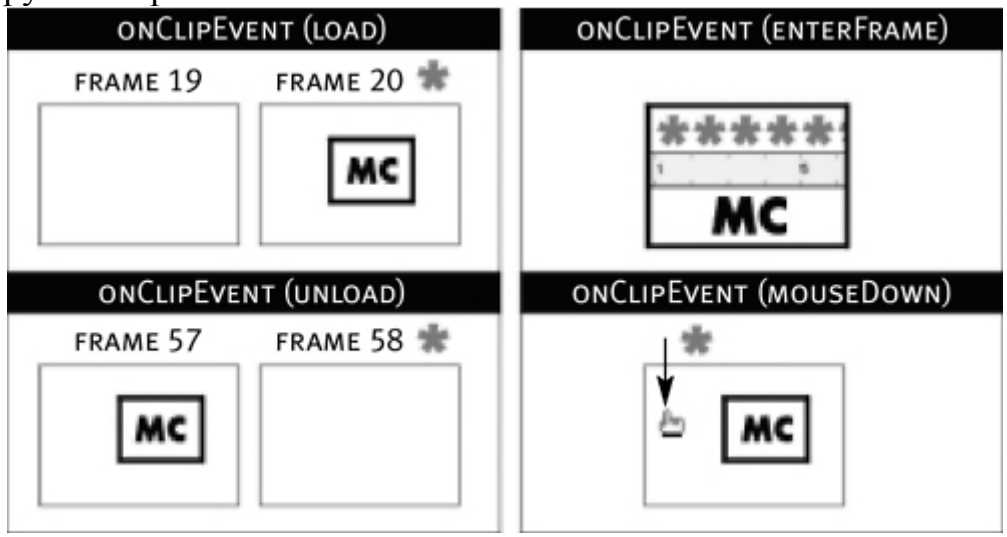
Другие виды взаимодействия с клавиатурой и мышью: onClipEvent (mouseDown), onClipEvent (mouseUp), onClipEvent (keyDown), onClipEvent (keyUp)

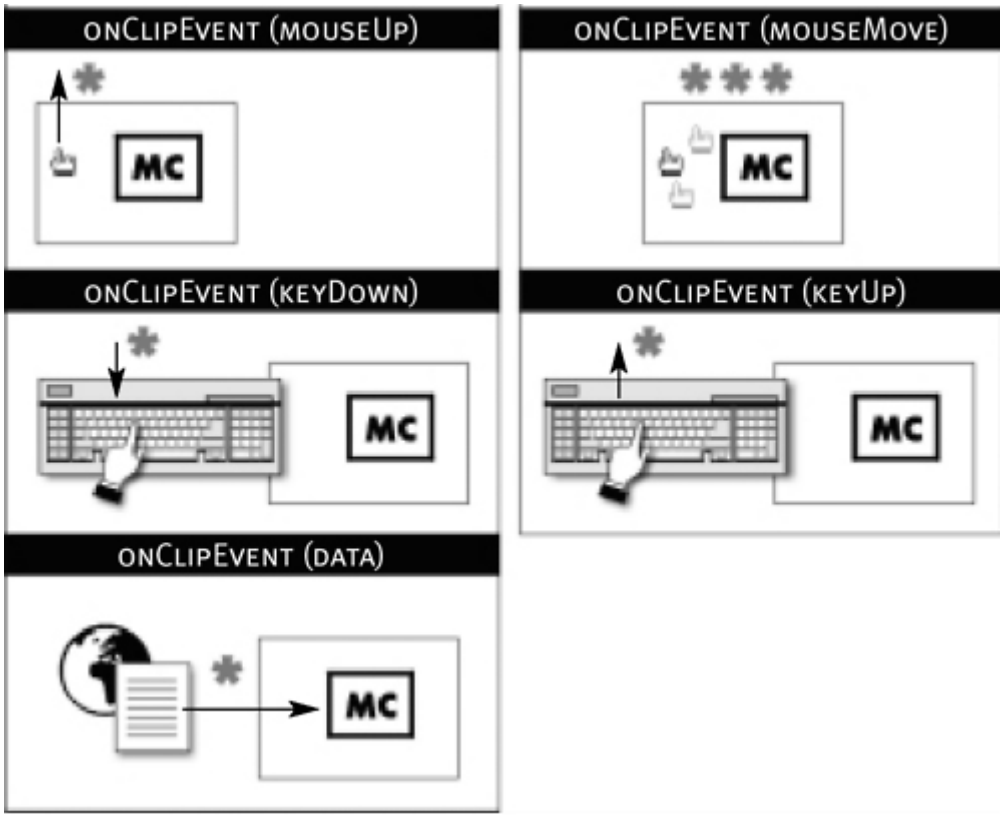
Поскольку мышь и клавиатура предназначены, чтобы взаимодействовать с компьютерами, эти события movie действительно не имеют реальных эквивалентов. Они действительно обеспечивают средство выполнения сценариев, когда определенные клавиши или кнопка мыши нажата (или отпущена). Хотя эти события могут казаться подобными on(press), и on(keyPress) событиям от нажатия мыши, описанным ранее, они фактически немного другие и обладают иной функциональностью. Используя keyUp и keyDown, возможно создавать перехватчик событий такой, как используется в Microsoft Word for Windows, создавая ключевые комбинации клавиш, или горячую клавиатуру, в вашем приложении так, чтобы действие было предпринято, когда последовательность клавиш нажата. Разумеется, все возможно в «разумных пределах» и перехватывать <Alt>+<F4> проект напрямую не сможет. Напротив, keyPress - событие от нажатия мыши, учитывает только единственную клавишу, чтобы инициализировать

действие. События mouseUp и release, mouseDown и press события от отпускания и нажатия мыши различны, потому что, в то время как последний вариант обработчика, работает непосредственно с кнопкой на которую наведен курсор мыши, а mouseDown и mouseUp работают в случае, когда мышь нажата или отпущена в пределах всего проекта.

Получение команд: onClipEvent (data)

В реальной жизни, неполные или недостаточные команды могут вести ко всем видам проблем. То же самое сохраняется в программе. Поскольку программа позволяет вам загружать различные типы данных (переменные, внешние SWFs, JPGs) из внешних источников в экземпляры movie clip, этот обработчик события играет жизненную роль, в которой он вызывает сценарий, приложенный к экземпляру объекта, только после того, как данные были полностью загружены в него из источника данных (файла). Таким образом, он препятствует вам получать тип ошибок, которые следуют из неполных команд или данных. Вы можете использовать этот событие, чтобы лишний раз не выполнять действия, если данные попытались более одного раза загруить в проект.





На рисунке представлено визуальное представление того, как вызваны события на movie clip. Стрелка "вниз" представляет придавливаемую кнопку мыши; стрелка "вверх" представляет то, что кнопка мыши была отпущена. Звездочка представляет возникновение события.

Использование нескольких событий одновременно

Различные события могут произойти одновременно в вашем проекте, например, пользователь может иметь одновременно нажатую кнопку мыши (вызов события от нажатия мыши) и перемещать курсор мыши по экрану (вызов события movie clip - mouseMove). При написании сценария необходимо учитывать, что все эти события должны между собой сочетаться оптимальным образом.

В упражнениях, вы будете управлять несколькими событиями, таким образом, чтобы, например, моделировать в программе команды для шара во время игры в пул и события удара кием во время оной. Хотя существуют некоторые весьма сложные способы сделать это, используя только один тип обработчиков событий, то в случае комбинирования различных типов событий возможно упростить процесс создания подобного проекта.

Использование методов обработчика события

Сценарии реагируют на следующие виды запускающих механизмов взаимодействия: взаимодействие с мышью, график времени, достигающий определенного фрейма, образцов movie clip, выводимых на сцену. Но используя методы обработчика события, вы можете заставить ваши сценарии

выполниться, когда звук заканчивает проигрываться, когда изменены параметры сцены, или когда текст в текстовом поле изменяется. Вы можете даже использовать методы обработчика события для расширения функциональных возможностей событий, о которых узнали ранее.
Хотя методы обработчика события и стандартные обработчики события используются для одной и той же основной цели (то есть выполнять сценарий, когда происходит некое событие вашем проект), но осуществить их возможно немного по-другому.
К настоящему времени, вы знаете, как записать сценарий, который будет выполнен как результат события. Например, следующий сценарий присоединен к экземпляру объекта movie clip, названному myMovieClip и будет выполнен всякий раз, когда левая клавиша мыши нажата, в то время как экземпляр объекта присутствует в сцене:

```
onClipEvent(mouseDown) {
_rotation = 45;
}
```

Этот сценарий будет поворачивать тот экземпляр объекта на 45 градусов, в момент нажатия левой клавиши мыши.
Используя метод обработчика события, следующий сценарий был бы помещен во фрейм графика времени, чтобы достигнуть той же самой цели, поворачивая myMovieClip всякий раз, когда нажата левая клавиша мыши (лкм).

```
myMovieClip.onMouseDown = function() {
myMovieClip._rotation = 45;
}
```

Вместо того, чтобы использовать onClipEvent, чтобы определить обработчик события (как показано в первом сценарии), здесь используется точка (.) для того, чтобы отделить название объекта (в этом случае myMovieClip) от события, на который он должно реагировать. Использовании function() – функции – это то, что это является необходимой частью синтаксиса чтобы осуществить метод обработчика события, как показано выше.
Как только вы становитесь знакомыми с функциями, если бы вы хотели бы, чтобы специфический был выполнен при определении метода обработчика события, вы изменили бы вышеупомянутый синтаксис следующим способом:

```
myMovieClip.onMouseDown = nameOfFunction;
```

Действия во второй строке сценария (между изогнутыми фигурными скобками) определяют, какие действия проихзойдут, после того, как указанное событие произойдет.
Обратите внимание, что в первом сценарии, никакой целевой путь (путь к экземпляру объекта) не определен (_rotation = 45), то во втором целевой путь определен (myMovieClip._rotation = 45). Так как первый сценарий присоединен к myMovieClip, никакой целевой путь не необходим, так как в качестве целевого объекта автоматически передается имя movie clip’а на котором сценарий был записан (аналогично вы могли использовать слово this). Однако, так как второй сценарий находится на фрейме, вы нуждаетесь в

указании пути, чтобы идентифицировать объект, который должен повернуться на 45 градусов. При использовании методов обработчика события таких, как описан здесь, вы должны знать о целевых путях в вызванных действиях. Есть конечно финт ушами, а именно прибегнуть к слову this – т.е указанию на текущий обънект и тогда, посольку обработчик события писался на movie clip с определенным ранее именем экземпляра, то об этом имении узнает и сценарий. Тем не менее, пока вы учитесь и не очень хорошо владете адресацией к объектам, лучше использовать полный целевой путь, но как следствие сделать свой код менее гибким, в тоже время лишенных многих ошибок для новичков.

Так как этот сценарий описывает, как экземпляр объекта myMovieClip реагирует на событие onMouseDown, то экземпляр объекта должен существовать в сцене во время метода обработчика события, и быть определен со своим индивидуальным именем. Это даст возможность прикрепить определенные функциональные возможности к экземпляру объекта. Той же самой лексемой, методы обработчика события, назначенные на объекты будут удалены, когда объект покинет сцену. Если объект появляется в сцене снова, любые методы обработчика события будут должны быть определены снова. Как всегда возможны некоторые тонкости, дело в том, что Flash неявным образом именует все экземпрляры, находящиеся на сцене помере их создания на оной: instance1, instance2…, такми образом, можно не описывая парамер Instance name, обратиться к любому объекту, находящемуся на сцене.

Использование методов обработчика события

Все стандартные обработчики события имеют эквивалентные методы обработчика события, как продемонстрировано ниже:

```
on(press) это buttonName.onPress или
movieClipName.onPress
on(release) это buttonName.onRelease или
movieClipName.onRelease
on(enterFrame) это movieClipName.onEnterFrame
```

Кроме того, следующие методы обработчика события существуют, но не имеют никаких стандартных эквивалентов следующие события:

Нажатия

```
nameOfClipOrButton.onKillFocus
nameOfClipOrButton.onSetFocus
```

Звук

```
nameOfSoundObject.onLoad
nameOfSoundObject.onSoundComplete
```

Текстовые Поля

```
nameOfTextField.onChanged
nameOfTextField.onKillFocus
nameOfTextField.onScroller
```

```
nameOfTextField.onSetFocus
```

LoadVars Объект

```
nameOfLoadVarsObject.onLoad
```

XML

```
nameOfXMLObject.onData
nameOfXMLObject.onLoad
```

XML Socket

```
nameOfXMLSocketObject.onClose
nameOfXMLSocketObject.onConnect
nameOfXMLSocketObject.onData
nameOfXMLSocketObject.onXML
```

Как вы можете видеть, вы можете использовать многочисленные события, чтобы вызвать сценарий. Так как некоторые из этих объектов неосязаемы (например, Звук, LoadVars, XML, и так далее), определяя методы обработчика события (на ключевом кадре графика времени) - единственный способ выполнить сценарий, когда событие происходит относительно того объекта (цели) (в отличие от кнопок и образцов movie clip, которые вы можете выбрать на сцене и прикрепить сценарии непосредственно к ним).

Прикрепляя сценарий к кнопке или к экземпляру movie clip внутри вашего проекта, необходимо использовать правильный обработчик события. Помотрите на следующий сценарий:

```
on(press){
gotoAndPlay(5);
}
```

Если бы вы должны были прикрепить этот сценарий к кнопке, кнопка реагировала бы только на событие нажатия, выполняя единственное действие, когда событие произошло. Предположите, что есть экземпляр объекта кнопки на сцене, названный myButton. Помещая следующий сценарий на фрейме 1 из главного графика времени (принимая кнопку существует в том же фрейме), вы определяете, как та кнопка будет реагировать на некоторые события:

```
myButton.onPress = function() {
stopAllSounds();
}
myButton.onRelease = function() {
myMovieClip._xscale = 50;
}
```

Когда кнопка нажата, она остановит все звуки; когда отпущена, это приведет к масштабированию myMovieClip к 50 процентам его первоначального размера.

Однако, перемещая этот график времени на другой ключевой кадр, который содержит, следующий сценарий (принимая кнопку существует в Фрейме 2) - Вы изменили бы функцию кнопки полностью:

```
myButton.onPress = null
myButton.onRelease = null
```

```
myButton.onRollOver = function() {
stopAllSounds();
}
myButton.onRollOut = function() {
myMovieClip._xscale = 50;
}
```

Используя пустой указатель null, вы препятствуете кнопке продолжать реагировать на onPress или onRelease событие - и вместо этого инструктируете ее реагировать на два недавно определенных события.

Как вы можете видеть, используя методы обработчика события, мы можем изменить функциональные возможности!

Методы обработчика события также удобно использовать для динамически созданных объектов. Так как действие динамического создания объекта вовлекает помещение объекта в ваш проект, и его при первоначальном создании проекта там не было, то используя стандартные обработчики события нет возможности задать какое-либо действие созданному динамически объекту - вам просто не на чем писать слово onClipEvent(). В этом методы обработчика события становятся наиболее полезны. Помотрите на следующий типовой сценарий, чтобы видеть, как вы могли осуществить это:

```
_root.createEmptyMovieClip("newClip", 1);
_root.newClip.onEnterFrame = function(){
myVariable++;
}
_root.newClip.onMouseMove = function(){
myCursorClip._x = _root._xmouse;
myCursorClip._y = _root._ymouse;
}
```

Как вы можете видеть, после создания экземпляра объекта movie clip, названного newClip, мы немедленно используем методы обработчика события для определения того, что экземпляр объекта сделает, когда произойдут требуемые события.

Использование обрабочков события. Использование слушателей (listener)

Предположим, что вы хотели чтобы ваш объект (звук, movie clip, текстовое поле, массив), реагировал на некоторые события, вызываемые в вашем проекте. Вы сначала создали бы метод обработчика события для того объекта:

```
myTextField.onMouseDown = function(){
myTextField.text="";
}
```

Здесь мы установили чтобы текстовое поле, названное myTextfield, реагировало, когда мышь нажата. onMouseDown - не стандартное событие для данного типа объектов. Ранее никогда не было, чтобы текстовое поле

реагировало на on и далее такого не будет. Для текстового поля, чтобы оно начало реагировать на этот событие, вы должны зарегистрировать текстовое поле как слушатель (listener) для события. Вы можете сделать это, добавляя следующий код:

```
Mouse.addListener ("myTextField");
```

Здесь текстовое поле зарегистрировано с объектом Mouse, так как onMouseDown событие - слушатель того объекта. Теперь, всякий раз, когда мышь придавлена, метод обработчика события мы создавали, ранее выполнится. Вы можете установить любое количество объектов к единственному случаю слушателя. Если вы больше не хотите, чтобы объект использовал данное событие, вы можете зделать и это, используясследующий синтаксис:

```
Mouse.removeListener ("myTextField"):
```

Слушатели только дают вам немного больше гибкости для того, чтобы писать сценарий и ваши проекты могли реагировать на большее количество действий.

Не все события могут быть обработаны как слушатели. Достаточно, знать, какие события перечислены как слушатели в инструментарии Действий (Actions). Если событие не перечислено как слушатель, другие объекты не могут быть использованы с ним.

В этом уроке, вы:

- Рассмотрите использование обработчиков событий в сценариях.
- Научитесь использовать события мыши/кнопки, чтобы управлять МС'ами внутри проекта.
- Научитесь добавлять управление клавиатуры к проекту.
- Создадите несинхронизированные представления, используя события фрейма (frame, кадр).
- Научитесь использовать событие movie clip, чтобы создать интерактивный проект.
- Научитесь использовать несколько событий одновременно для выполнения поставленной задачи.
- Научитесь использовать методы обработчика события.
- Узнаете о listener'ах и о их использовании.

Использование событий от левой клавиши мыши (MOUSE EVENTS)

1. Откройте MouseEvents1.fla в папке Lesson02/Assets. Раскройте панель со сценой (Scene panel) и Property inspector. Внимательно рассмотрите содержание сцены.
2. Заблокируйте слой с кнопкой сканера (Scanner button) и щелкните два раза левой клавишей мыши (лкм) по МС с именем экземпляра scanner, находящегося в центре сцены (stage) для того, чтобы отредактировать его или рассмотреть его содержимое.

3. Вернитесь обратно на сцену (иногда подобный переход может называться переходом к основной линии времени – main timeline). При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт:

```
stop ();
Mouse.hide();
startDrag ("hand", true);
message.text = "Please place your finger on the
scanner and press down.
Release after 2 seconds.";
```

4. Не закрывая панель Actions, разблокируйте слой Scanner button и выберите кнопку, которая находится на этом слое (которая выглядит как квадратная стеклянная пластинка), и добавьте следующий скрипт:

```
on (rollOver) {
 scanner.gotoAndPlay ("Active");
 message.text = "Please press on the screen to
continue.";
}
```

5. Добавьте скрипт сразу после предыдущего:

```
on (rollOut) {
 scanner.gotoAndPlay ("Inactive");
 message.text = "Please place your finger on the
scanner and press down.
Release after 2 seconds.";
}
```

6. Добавьте еще скрипт:

```
on (press) {
 scanner.gotoAndPlay ("Scan");
 message.text = "Now scanning...";
}
```

7. И еще скрипт:

```
on (dragOut) {
 scanner.gotoAndPlay ("Error");
 message.text = "An error has occurred. You have
removed your finger prior to
processing. Please try again.";
```

```
}
```

8. Добавьте следующий скрипт ниже:

```
on (release) {
  scanner.gotoAndPlay ("Process");
  message.text = "Processing your scan...";
}
```

9. Если у вас до сих пор была открыта панель Scene, то это было не зря, тем, кто закрыли эту панель, необходимо ее открыть снова для того, чтобы при помощи этой панели перейти на сцену с именем Playroom.
10. При открытой панели Actions выберите первый кадр на слое Actions и напишите следующий скрипт:

```
startDrag ("hand", true);
```

11. Не сворачивая и не закрывая панель Actions, выберите маленький кусочек дерева (с instance name name – bump) и добавьте следующий скрипт:

```
on (dragOver) {
  this._alpha = this._alpha - 10;
}
```

12. С раскрытой панелью Actions выберите MC с именем card и запишите следующий скрипт:

```
on (releaseOutside) {
  this._x = _root._xmouse;
  this._y = _root._ymouse;
}
```

13. При раскрытой панели Actions выберите пустой MC (movie clip), который находится за пределами сцены (над ней) и введите следующие строчки кода:

```
on (keyPress "<Space>") {
  _root.bump._alpha = 100;
}
on (keyPress "<Left>") {
  _root.gotoAndStop (1);
}
```

14. В меню выполните следующую команду Control > Test Movie, для того чтобы просмотреть проект в действии.
15. Закройте окно просмотра и сохраните проект как MouseEvents2.fla.

Использование событий кадра (FRAME EVENTS)

1. Откройте FrameEvents1.fla в папке Lesson02/Assets и затем откройте Property inspector.
2. Щелкните дважды лкм по MC с instance name name picture, который находится в центре, чтобы подробно рассмотреть его содержимое и отредактировать его.
3. Раскрыв панель Actions, выберите первый кадр слоя Actions и введите следующий код:

```
stop ();
_root.date.text = "June 15th";
_root.country.text = "Scotland";
_root.caption.text = "Isn't this a beautiful place?";
```

4. Затем, не закрывая панель Actions, выберите кадр 2 в слое Actions и добавьте скрипт:

```
_root.date.text = "June 16th";
_root.country.text = "Italy";
_root.caption.text = "The food was excellent!";
```

5. Затем с раскрытой панелью Actions выберите кадры 3, 4, и 5 в слое с именем Actions и введите следующий скрипт соответственно:

В 3-м кадре:

```
_root.date.text = "June 17th";
_root.country.text = "Greece";
_root.caption.text = "We went for a boat ride.";
```

В 4-м кадре:

```
_root.date.text = "June 18th";
_root.country.text = "France";
_root.caption.text = "We took another boat ride.";
```

В 5-м кадре:

```
_root.date.text = "June 19th";
_root.country.text = "Monaco";
_root.caption.text = "The mountains were amazing!";
```

6. Выберите 6-й кадр в слое Actions и, не закрывая панель Actions, введите скрипт:

```
gotoAndStop (1);
```

7. Вернитесь на основной уровень (сцену). При развернутой панели Actions выберите ключевой кадр (keyframe), находящийся на 140 кадре, и введите следующий скрипт:

```
warning.text = "The picture will change in 3
seconds.";
indicator.gotoAndPlay ("On");
```

8. Затем, не закрывая панель Actions, выберите ключевой кадр (keyframe), который находится на 160 кадре, и добавьте нижеприведенный скрипт:

```
warning.text = "The picture will change in 2
seconds.";
```

9. После этого, не закрывая панель Actions, выберите ключевой кадр (keyframe), который находится на 180 кадре, и добавьте нижеприведенный скрипт:

```
warning.text = "The picture will change in 1
second.";
```

10. Затем, не закрывая панель Actions, выберите ключевой кадр (keyframe), который находится на 200 кадре, и добавьте нижеприведенный скрипт:

```
pictures.nextFrame();
warning.text = "";
gotoAndPlay (1);
```

11. С развернутой панелью Actions выберите треугольную кнопку на панели управления действием и добавьте следующий скрипт:

```
on (release) {
  play ();
}
on (rollOver) {
  balloon.text = "Play";
}
on (rollOut) {
  balloon.text = "";
```

```
}
```

12. Затем с развернутой панелью Actions выберите квадратную кнопку на панели управления действием и добавьте следующий скрипт:

```
on (release) {
  stop ();
}
on (rollOver) {
  balloon.text = "Stop";
}
on (rollOut) {
  balloon.text = "";
}
```

13. Затем с развернутой панелью Actions выберите кнопку в виде сдвоенных треугольников, располагающуюся на панели управления действием и добавьте следующий скрипт:

```
on (release) {
  gotoAndPlay (1);
  pictures.gotoAndStop (1);
  warning.text = "";
}

on (rollOver) {
  balloon.text = "Rewind";
}

on (rollOut) {
  balloon.text = "";
}
```

14. В меню выберите Control > Test Movie для просмотра проекта в действии.
15. Закройте окно предварительного просмотра и сохраните проект под именем FrameEvents2.fla.

Использование событий, записываемых на Movie Clip (CLIP EVENTS)

1. Откройте файл ClipEvents1.fla, находящийся в папке Lesson02/Assets. Откройте или разверните Property inspector.
2. Дважды щелкните лкм по маленькому кружочку в левом верхнем углу рабочей области, который находится сразу над сценой (stage), для того, чтобы отредактировать его или просмотреть его содержимое.

3. Вернитесь на основную временную линейку (на сцену, main timeline). В слое Lights, щелкните по кнопке Show Layer (отображается красным значком X) для обнаружения большого черного квадрата, закрывающего всю рабочую часть сцены. Дважды щелкните лкм по нему, чтобы отредактировать его или просмотреть его содержание.
4. Вернитесь на основной уровень. В слое Lights, нажмите кнопку Hide Layer для того, чтобы скрыть большую черную область, которая, как было выяснено ранее, является Movie Clip'ом и имеет instance name name – lights. С открытой панелью Actions выберите 1-й кадр на слое Actions и прикрепите к нему следующий скрипт:

```
message.text = "All Clear";
stop ();
```

5. При открытой панели Actions выберите кнопку с надписью Tiny и запишите следующий скрипт:

```
on (release) {
  if (message.text == "All Clear") {
    size = 50;
    gotoAndStop ("Burglar");
  } else {
    burglar._xscale = 50;
    burglar._yscale = 50;
  }
}
```

6. Затем при открытой панели Actions выберите кнопку с надписью Small и запишите следующий скрипт:

```
on (release) {
  if (message.text == "All Clear") {
    size = 75;
    gotoAndStop ("Burglar");
  } else {
    burglar._xscale = 75;
    burglar._yscale = 75;
  }
}
```

7. После этого при открытой панели Actions выберите кнопку с надписью Normal и запишите следующий скрипт:

```
on (release) {
  if (message.text == "All Clear") {
```

```
   size = 100;
   gotoAndStop ("Burglar");
  } else {
   burglar._xscale = 100;
   burglar._yscale = 100;
  }
}
```

8. Не закрывая панель Actions, выберите кнопку No Burglar и запишите следующий скрипт:

```
on (release) {
 gotoAndStop ("Clear");
}
```

9. При раскрытой панели Actions раскройте панель времени, и на основной временной линейке времени (main timeline) выделите ключевой кадр, имеющий метку (label) Burglar - грабитель, выберите MC burglar и прицепите к нему скрипт:

```
onClipEvent (load) {
 startDrag (this, true);
 this._xscale = _root.size;
 this._yscale = _root.size;
 _root.lights.gotoAndStop("bottom");
 _root.mouseXPosition.text = _root._xmouse;
 _root.mouseYPosition.text = _root._ymouse;
 _root.message.text = "ALERT!";
}
```

10. Как только закончится предыдущий кусок кода, введите новый фрагмент:

```
onClipEvent (enterFrame) {
 _root.time++;
 _root.timeAmount.text = _root.time;
 _root.timer._rotation = _root.timer._rotation + 1;
}
```

11. Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (mouseMove) {
 if                  (_root._xmouse                    >
Number(_root.mouseXPosition.text) + 10) {
   _root.lights.gotoAndStop("right");
   _root.message.text = "Intruder is moving East";
```

```
  }       else       if       (_root._xmouse       <
Number(_root.mouseXPosition.text) - 10) {
    _root.lights.gotoAndStop("left");
    _root.message.text = "Intruder is moving West";
  }       else       if       (_root._ymouse       >
Number(_root.mouseYPosition.text) + 10) {
    _root.lights.gotoAndStop("bottom");
    _root.message.text = "Intruder is moving South";
  }       else       if       (_root._ymouse       <
Number(_root.mouseYPosition.text) - 10) {
    _root.lights.gotoAndStop("top");
    _root.message.text = "Intruder is moving North";
  }
    _root.mouseXPosition.text = _root._xmouse;
    _root.mouseYPosition.text = _root._ymouse;
}
```

12.Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (unload) {
  _root.time = 0;
  _root.timeAmount = _root.time;
  _root.timer._rotation = 0;
  _root.message.text = "All Clear";
  _root.mouseXPosition.text = "";
  _root.mouseYPosition.text = "";
  _root.lights.gotoAndStop ("Off");
}
```

13.Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (mouseDown) {
  this.gotoAndStop("right");
  _root.message.text = "Intruder is confused";
}
```

14.Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (mouseUp) {
  this.gotoAndStop("left");
  _root.message.text = "Intruder is running";
}
```

15.Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (keyDown) {
```

```
  _root.siren.gotoAndStop("on");
  _root.message.text = "Backup has been called";
}
```

16. Добавьте следующий скрипт сразу после изложенного выше:

```
onClipEvent (keyUp) {
  stopAllSounds ();
  _root.siren.gotoAndStop("off");
  _root.message.text = "Silent alarm activated";
}
```

17. В меню выберите Control > Test Movie для того, чтобы протестировать функциональность вашего проекта.
18. Закройте окно предварительно просмотра и сохраните проект под именем ClipEvents2.fla.

Гармоничное сочетание различных событий

1. Откройте OrchestrateEvents1.fla в папке Lesson02/Assets.
2. Раскройте панель Actions, выберите невидимую кнопку (может отображаться голубым цветом) и напишите следующий скрипт:

```
on (press) {
  ball._x = 360;
  ball._y = 180;
  power = 0;
  powerAmount.text = power;
  hitAmount = 0;
  trackMouse = true;
  mouseStart = _root._xmouse;
}
```

3. Не закрывая панель Actions, добавьте следующий скрипт после окончания предыдущего:

```
on (dragOut) {
  stick.gotoandPlay ("PullBack");
}
```

4. Не закрывая панель Actions, добавьте еще один скрипт:

```
on (releaseOutside) {
  stick.gotoandStop ("Starting");
  mouseEnd = _root._xmouse;
  hitAmount = mouseEnd - mouseStart;
```

```
  trackMouse = false;
}
```

5. Теперь при раскрытой панели Actions выберите movie clip с instance name ball и напишите следующий скрипт:

```
onClipEvent (enterFrame) {
  if (_root.hitAmount > 5) {
    this._x = this._x - 10;
    _root.hitAmount = _root.hitAmount - 2;
  }
}
```

6. Добавьте нижеприведенный скрипт, как только первый закончится:

```
onClipEvent (mouseMove) {
  if (_root.trackMouse == true) {
    _root.power = _root._xmouse -_root.mouseStart;
    _root.powerAmount.text = _root.power;
  }
}
```

7. Выберите Control > Test Movie.
8. Закройте окно предварительного просмотра, чтобы вернуться в среду для создания приложений, и сохраните проект OrchestratingEvents2.fla.

Использование методов обработчика событий (EVENT HANDLER)

1. Откройте CarParts1.fla, находящийся в папке Lesson02/Assets.
2. Раскройте панель Actions, выберите первый кадр и напишите туда следующий скрипт:

```
text1.onChanged = function(){
  car._x += 2;
  car.wheel1._rotation += 10;
  car.wheel2._rotation += 10;
  arrow1._x = text1._x + text1.textWidth;
}
```

3. При раскрытой панели Actions запишите следующий скрипт сразу за вышеприведенным:

```
text1.onSetFocus = function(){
  wheelClip.onEnterFrame = function(){
    wheelClip._rotation += 30;
```

```
  }
  speedClip.onPress = null;
  fanClip.onEnterFrame = null;
}
```

4. При раскрытой панели Actions запишите следующий скрипт сразу за вышеприведенным:

```
text2.onChanged = function(){
  car._x += 2;
  car.wheel1._rotation += 10;
  car.wheel2._rotation += 10;
  arrow2._x = text2._x + text2.textWidth;
}
```

5. При раскрытой панели Actions запишите следующий скрипт после вышеприведенного:

```
text2.onSetFocus = function(){
  wheelClip.onEnterFrame = null;
  speedClip.onPress = function(){
    speedClip.play();
  }
  fanClip.onEnterFrame = null;
}
```

6. При раскрытой панели Actions запишите следующий скрипт сразу за вышеприведенным:

```
text3.onChanged = function(){
  car._x += 2;
  car.wheel1._rotation += 10;
  car.wheel2._rotation += 10;
  arrow3._x = text3._x + text3.textWidth;
}
text3.onSetFocus = function(){
  wheelClip.onEnterFrame = null;
  speedClip.onPress = null;
  fanClip.onEnterFrame = function(){
    fanClip._rotation += 20;
  }
}
```

7. Выберите Control > Test Movie.
8. Закройте окно предварительно просмотра и сохраните проект под именем CarParts2.fla.

Использование путей к объектам (TARGETING)

Использование путей к текущему Move Clip'у (THIS MOVIE)

1. Откройте файл Target1.fla из папки Lesson03/Assets.
2. Выберите 1-й кадр на основной линейке времени и запишите на этот кадр следующий скрипт:

```
startingColor = random (4) + 1;
this.gotoAndStop (this.startingColor);
```

3. Дважды щелкните лкм по МС, находящемуся в середине сцены для того, чтобы отредактировать его.
4. С раскрытой панелью Actions выберите невидимую кнопку и напишите следующий скрипт:

```
on (press) {
  startDrag (this);
  gotoAndStop ("Speak");
  balloon.text = words;
}
on (release) {
  stopDrag ();
  this.gotoAndStop ("Quiet");
}
```

5. Вернитесь на основную временную дорожку. Откройте панель Library, перетащите еще несколько экземпляров movie clip'ов Hatfield на сцену. Выберите в меню Control > Test Movie, чтобы протестировать, как будет работать сцена при наличии нескольких экземпляров.
6. Закройте тесовое окно для того, чтобы вернуться в среду создания проекта. При раскрытой панели Actions выберите один из экземпляров МС, находящихся на сцене, и добавьте следующий скрипт:

```
onClipEvent (load) {
  words = "My name is Derek";
  this._xscale = 75;
  _yscale = 75;
}
```

7. Затем с раскрытой панелью Action выберите следующий экземпляр МС и задайте ему скрипт:

```
onClipEvent (load) {
```

```
  words = "My name is Ashlie";
  _x = 400;
  _y = 300;
}
```

8. Затем с раскрытой панелью Action выберите еще один экземпляр MC и задайте ему скрипт:

```
onClipEvent (load) {
  words = "My name is Kathy";
}
onClipEvent (mouseMove) {
  this._rotation = this._rotation + .5;
}
```

9. Выберите Control > Test Movie, чтобы проверить изменения, внесенные вами в процессе доводки приложения.
10. Закройте окно просмотра и сохраните файл с именем Target2.fla.

Использование путей к главному Movie Clip'у (THE MAIN MOVIE)

1. Откройте rootTarget1.fla из папки Lesson03/Assets.
2. Раскройте панель Actions, выберите кнопку рядом с изображением знака минус и напишите следующий скрипт:

```
on (release) {
  _root._xscale = _root._xscale - 10;
  _root._yscale = _root._yscale - 10;
}
```

3. При открытой панели Actions выберите кнопку, рядом с которой находится знак плюс, и добавьте этот скрипт:

```
on (release) {
  _root._xscale = _root._xscale + 10;
  _root._yscale = _root._yscale + 10;
}
```

4. Выделите обе кнопки (вместе с текстовым содержимым рядом с ними) и скопируйте их. Щелкните дважды лкм по одному из экземпляров MC для того, чтобы отредактировать его. Вставьте скопированные ранее кнопки на слой 'Change root buttons' текущего MC и позиционируйте их чуть ниже центра текущего MC.
5. Выберите Control > Test movie для того, чтобы протестировать проект.

6. Закройте окно просмотра и сохраните с именем rootTarget2.fla.

Использование адресации к родительском у Movie Clip'у (PARENT MOVIE)

1. Откройте файл parentTarget1.fla из папки Lesson03/Assets.
2. Щелкните дважды лкм по экземпляру MC'a, находящегося на сцене, для того, чтобы отредактировать его. Откройте панель Library (библиотека), выделите слой Child Clip, из библиотеки перетащите экземпляр movie clip'a с именем Hatfield Child на сцену, чуть правее графики текущего MC.
3. Не закрывая панель Actions, выберите дочерний (child) movie clip и добавьте следующий скрипт:

```
onClipEvent (load) {
  this.words = _parent.words + "'s kid";
}
```

4. В меню выберите Control > Test Movie для того, чтобы узнать, как работает проект на данной стадии разработки.
5. Закройте тестирующее приложение, чтобы вернуться в среду создания проекта. Не закрывая панель Library, щелкните дважды лкм по MC'у с именем Swirl Clip:
6. С раскрытой панелью Actions выберите кадры 1, 2, 3, и 4, и соответственно, напишите на них следующие скрипты:

На кадре 1:

```
_parent._x = _parent._x - 1;
_parent._xscale = _parent._xscale - 1;
```

На кадре 2:

```
_parent._y = _parent._y - 1;
_parent._yscale = _parent._yscale - 1;
```

На кадре 3:

```
_parent._x = _parent._x + 2;
_parent._xscale = _parent._xscale - 1;
```

На кадре 4:

```
_parent._y = _parent._y + 2;
_parent._yscale = _parent._yscale - 1;
```

7. Вернитесь на основную сцену. С развернутой панелью Library выделите слой с названием Swirl Clip, перетащите MC с именем Swirl Clip из библиотеки на сцену. Теперь снова выберите Control > Test Movie для тестирования проекта на текущем этапе создания.
8. Закройте окно предварительного просмотра и вернитесь в среду создания проектов. Выберите MC с именем Swirl Clip на основной линейке времени (сцене), в меню выполните команду Edit > Cut. Щелкните дважды лкм на movie clip'е с instance name Hatfield для того, чтобы перейти на его линейку времени (timeline). Выделите слой (layer) с названием Swirl Clip и вставьте вырезанный ранее MC Swirl Clip при помощи команды меню Edit > Paste в этот слой. Теперь снова выберите в меню Control > Test Movie, чтобы еще раз протестировать проект.

В принципе, вы можете поэкспериментировать, используя адресацию к родительским объектам аналогично следующему:

```
_parent._parent._alpha = 50;
```

9. Закройте окно просмотра и сохраните ваш проект под именем parentTarget2.fla.

Использование адресации к произвольным Movie Clip'ам

1. Откройте файл movieclipTarget1.fla, находящийся в папке Lesson03/Assets.
2. Раскройте Инспектор свойств (Property inspector), выберите каждый из экземпляров (instance name) MC'ов, находящихся на сцене (stage), и назовите их в соответствии со значением тех слов, которые они произносят в процессе работы.

Например, один из MC произносит фразу после загрузки: "I'm Derek", ну и соответственно MC будет назван `Derek`. Со всеми остальными MC'ами проделайте то же самое.

3. Дважды щелкните лкм по каждому из экземпляров MC, находящихся на сцене. На временной линейке внутри из MC'ов будет находиться MC с именем Hatfield Child. Откройте Property inspector, выберите экземпляр и задайте ему имя в поле instance name– myKid. После этой операции вернитесь на основную линейку времени.

После предпринятых вами операций в проекте у вас присутствуют шесть экземпляров MC'ов, к которым можно обращаться из любой части вашего проекта. Абсолютные пути при этом будут выглядеть следующим образом:

```
_root.Derek
_root.Derek.myKid
_root.Kathy
_root.Kathy.myKid
_root.Ashlie
_root.Ashlie.myKid
```

4. Раскройте панель Actions и выберите MC с именем (instance name name) Kathy и добавьте следующий скрипт, находящийся в конце уже имеющегося скрипта (он выглядит следующим образом: `this._rotation = this._rotation + .5;`):

```
myKid._rotation = myKid._rotation + 20;
_root.Derek.myKid._xscale = _root.Derek.myKid._xscale
+ .5;
_root.Derek.myKid._yscale = _root.Derek.myKid._yscale
+ .5;
_root.Ashlie.myKid._y = _root.Ashlie.myKid._y - .5;
```

5. Выберите Control > Test Movie для того, чтобы протестировать проект в действии.
6. Закройте окно просмотра и сохраните проект под именем movieclipTarget2.fla.

Использование адресации к внешним Flash файлам, расположенным на уровнях (MOVIES ON LEVELS)

1. Откройте файл backgroundControl1.fla из папки Lesson03/Assets.
2. Откройте панель Actions, выберите первый кадр основной линейки времени, запишите скрипт:

```
_visible = false;
```

3. При раскрытой панели Actions, выберите круглую кнопку Exit (со значком X на ней) и запишите следующий скрипт:

```
on (release) {
  _visible = false;
}
```

4. Не закрывая панель Actions, выберите многоугольную невидимую кнопку (представлена в виде светло-голубой области) наверху диалогового окна и запишите скрипт:

```
on (press) {
  _alpha = 50;
```

```
  startDrag (this);
}
on (release) {
  _alpha = 100;
  stopDrag ();
}
```

5. Экспортируйте этот файл под именем backgroundControl.swf в папку Lesson03/Assets (File > Export) или опубликуйте этот файл в аналогичный swf стандарт (File > Publish).
6. Сохраните ваше творчество под именем backgroundControl2.fla.
7. Откройте textBox1.fla из папки Lesson03/Assets.
8. Разверните панель Actions, выберите первый кадр на основной линейке времени и запишите на этот кадр следующий скрипт:

```
_visible = false;
```

9. Пока панель Actions еще открыта, выберите круглую кнопку Exit (со значком X на ней) и добавьте следующий скрипт:

```
on (release) {
  _visible = false;
}
```

10. Не закрывая панель Действий (Actions), выберите многоугольную невидимую кнопку (как и раньше, она выглядит как полупрозрачная светло-голубая область), находящуюся наверху диалогового окна, и добавьте на нее следующий скрипт:

```
on (press) {
  _alpha = 50;
  startDrag (this);
}
on (release) {
  _alpha = 100;
  stopDrag ();
}
```

11. Экспортируйте или опубликуйте этот файл под именем textBox.swf в папку Lesson03/Assets.
12. Сохраните вашу работу под именем textBox2.fla.
13. Откройте файл levelTarget1.fla из папки Lesson03/Assets.
14. При раскрытой панели Actions выберите первый кадр и введите в него приведенный ниже скрипт:

```
loadMovieNum ("backgroundControl.swf", 1);
```

```
loadMovieNum ("textBox.swf", 2);
```

15. При раскрытой панели Actions выберите кнопку слева сцены, которая представляет собой компьютерную иконку (пиктограмму), и добавьте на нее следующий скрипт:

```
on (release) {
  _level1._visible = true;
}
```

16. При раскрытой панели Actions выберите кнопку в левой части сцены, которая представляет собой бумажный свиток, и добавьте на нее следующий скрипт:

```
on (release) {
  _level2._visible = true;
}
```

17. При раскрытой панели Actions выберите кнопку в виде стрелки влево левее цифры 1 внизу рабочей области и добавьте на нее следующий скрипт:

```
on (release) {
  _level1._xscale = _level1._xscale - 5;
  _level1._yscale = _level1._yscale - 5;
}
```

18. При раскрытой панели Actions, выберите кнопку в виде стрелки вправо левее цифры 1 внизу рабочей области и добавьте на нее следующий скрипт:

```
on (release) {
  _level1._xscale = _level1._xscale + 5;
  _level1._yscale = _level1._yscale + 5;
}
```

19. Не закрывая панель Actions, задайте нижеприведенные скрипты для указателей влево и вправо рядом с цифрой 2, которые находятся также в нижней части рабочей области.

На стрелочку влево:

```
on (release) {
     _level2._xscale = _level2._xscale - 5;
     _level2._yscale = _level2._yscale - 5;
}
```

На стрелочку вправо:

```
on (release) {
     _level2._xscale = _level2._xscale + 5;
     _level2._yscale = _level2._yscale + 5;
}
```

20. В меню выберите Control > Test Movie для проверки функционирования вашего проекта.
21. Закройте окно предварительного просмотра и вернитесь в среду, в которой вы создавали проект. Запишите ваше творение под именем levelTarget2.fla.

Использование адресации к Movie Clip'ам, загруженным на уровни (MOVIE CLIP INSTANCE NAMES ON LEVELS)

1. Откройте файл с именем levelTarget2.fla из папки Lesson03/Assets.
2. Нажмите на кнопку Show Layer (в виде большого знака X) на слое Colors Clip для того, чтобы увидеть содержимое этого слоя.
3. Дважды щелкните лкм по экземпляру movie clip с instance name name color для того, чтобы просмотреть его содержимое или отредактировать его.
4. Вернитесь на основную временную линейку (сцену). Откройте Property inspector, выберите текстовое поле (text field), находящееся в нижней левой части сцены после надписи "Enter Text:"

Посмотрите в Property inspector, в нем вы увидите, что данное текстовое поле имеет имя (instance name) inputText.

5. Раскройте панель Actions, выберите кнопку на виртуальном рабочем столе (desktop), представляющую собой бумажный свиток, и введите следующий скрипт ниже строки с кодом `_level2._visible = true;`:

```
_level2.inputText.text = _level0.inputText.text;
```

6. С раскрытой панелью Actions выберите первый кадр на основной линии времени и введите следующий скрипт сразу за строчкой с кодом `loadMovieNum ("textBox.swf", 2);`:

```
_level0.colors._alpha = 50;
```

7. Экспортируйте или опубликуйте этот файл в SWF под именем levelTarget.swf в папку Lesson03/Assets.
8. Сохраните файл под именем levelTarget3.fla.
9. Откройте файл backgroundControl2.fla из папки Lesson03/Assets.

10. При раскрытой панели Actions выберите квадратную кнопку в самой дальней левой части рабочей области рядом с текстовым полем alphaAmount и запишите на нее следующий скрипт:

```
on (rollOver) {
  alphaAmount.text = 0;
}
on (release) {
  _level0.colors._alpha = 0;
}
on (release, rollOut) {
  alphaAmount.text = _level0.colors._alpha;
}
```

11. При раскрытой панели Actions, выберите красную кнопку Color Swatch (первая слева) и запишите на нее следующий скрипт:

```
on (rollOver) {
  colorName.text = "Purple";
}
on (release) {
  currentColor.gotoAndStop ("Purple");
  _level0.colors.gotoAndStop ("Purple");
}
on (release, rollOut) {
  colorName.text = "Please select a color";
}
```

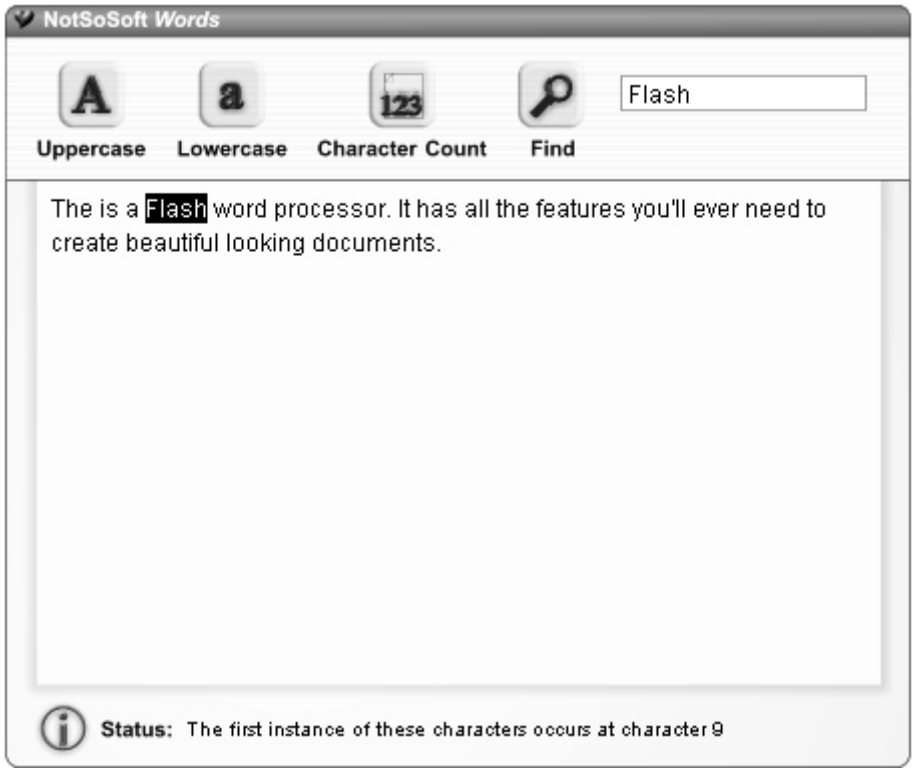
12. Экспортируйте или опубликуйте этот файл в SWF под именем backgroundControl.swf в папку Lesson03/Assets.
13. Сохраните вашу работу под именем backgroundControl3.fla.
14. Найдите файл levelTarget.swf, созданный вами чуть ранее. Щелкните по нему дважды лкм для того, чтобы его запустить его и проиграть.

Использование стандартных объектов и классов ActionScript

Каждый день вы используете объекты, чтобы исполнить любое действие. Объекты - элементы, разработанны, чтобы исполнить разные потребности в стандартных объектах, Но вы можете использовать их, чтобы исполнить собственные задачи.

Объекты в Macromedia Flash подобны реальным объектам. Каждый из многих типов объектов Flash выполняет определенные потребности в ваших приложениях.

Этот программный текстовый процессор - один из проектов, которые вы создадите в этом пособии.



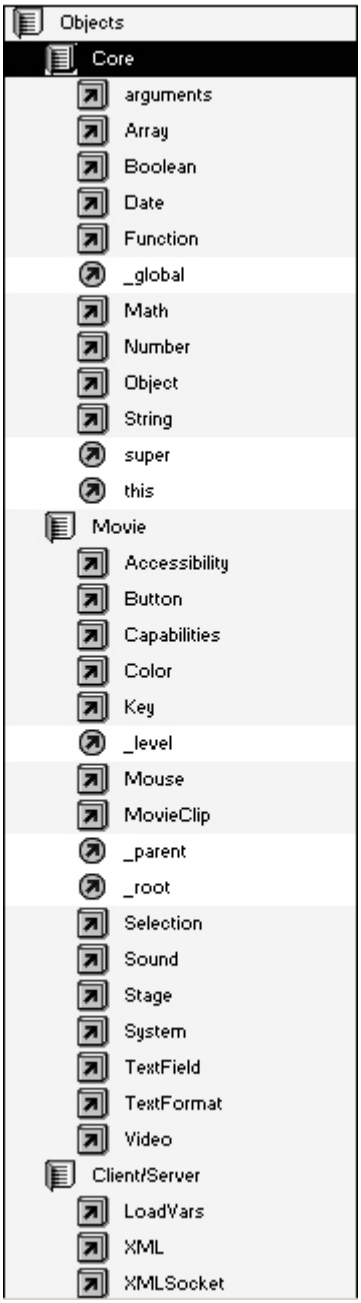
В этом уроке, вы:

- Изучите то, чем объекты являются и почему они полезны.
- Познакомитесь с различными объектами, доступными в ActionScript.
- Используете объект Color.
- Создадите интерактивную сцену, используя Ключевой объект.
- Создадите текстовый процессор, используя свойства и методы объектов String и Selection.

Приемущества от использования стандартных объектов

Вы можете осмыслять объекты Flash, как элементы объектов плотника, используемые определенным способом, чтобы формировать приложение или исполнять некоторую задачу. Если вы думаете об объектах как об инструментальных средствах и строительных материалах для вашего проекта, вы сможете понять их приемущества.

ActionScript имеет множество предварительно подготовленных объектов для различных целей. Однако, если вы не можете найти объект, который соответствует вашим потребностям, вы можете создать собственный и программа, и программа сделает всё что вы захотите. Объекты (Object) определены в соответствии с двумя первичными характеристиками: свойства и методы.



Свойства

Многие, но не все, объекты имеют значения свойств, которые представляют характеристики объекта. В реальном мире, автомобиль имеет свойства подобно цвету, марки, модель, лошадиная сила, и так далее. Если тот автомобиль был объектом ActionScript, вы могли бы определить это следующим образом:

```
car.color = "красный";
car.make = "Фольксваген";
car.model = "Жук";
car.horsepower = 200;
```

В программе есть несколько объектов, которые имеют свойства. Например, объект MovieClip имеет значения свойства, которые представляют его прозрачность, видимость, горизонтальную позицию, вертикальную позицию,

вращение {циклический сдвиг}, и другие (_x, _y, _xscale, _yscale, _rotation, _visible, _alpha). Изменение любого из этих свойств затрагивает появление movie clip или функциональные возможности, также как у автомобиля, изменение расцветки или изменение ее механизма изменили бы составляющие. Вы можете использовать значения свойства различных объектов в ваших сценариях, чтобы установить значения в другом месте. Давайте предполагать, что сценарий в вашем проекте перемещает ваш автомобиль на скорости, основанной на ее лошадиных силах. То строка сценария могла бы напомнить следующее:

```
speedFactor = car.horsepower;
```

Давайте посмотрим на еще один пример свойства и как это используется в ActionScript.

Длина строки - свойство объекта String. Например, длина "Flash" - 5, потому что содержит пять символов. В ActionScript, это было бы написано следующим образом:

```
name = "Flash";
lengthOfName = name.length;
```

Первая строка программы создает переменную названную name {именем}, значение которого - "Flash" строка. Вторая строка создает переменную, названную lengthOfName, чье значение - это свойства длины объекта (5) . Хотя названиям свойства, связанным с movie clip обычно предшествует символ подчеркивания (_alpha, _rotation, и так далее), не, все объектные названия свойств следуют за этим соглашением. Причина для этого состоит в том, что соглашение является пережитком от Flash 4 (когда свойства были введены, и ActionScript очень отличался от современного вида).

ПРИМЕЧАНИЕ

Объекты могут хранить массивы, переменные, и даже другие объекты - все из которых рассматриваются свойствами.

Методы

Метод представляет задачу, которую объект может исполнить. Если вы думаете о видеомагнитофоне как о объекте, его методы включают способности, чтобы запустить, делать запись, останавливаться, перематываться, быстро ускоряться, делать паузу, и т.д. Метод состоит из его названия, сопровождаемого набором круглых скобок. Методы нашего объекта видеомагнитофона выглядели бы следующим образом:

```
игра(); - play()
перемотка();
запись(); - record()
```

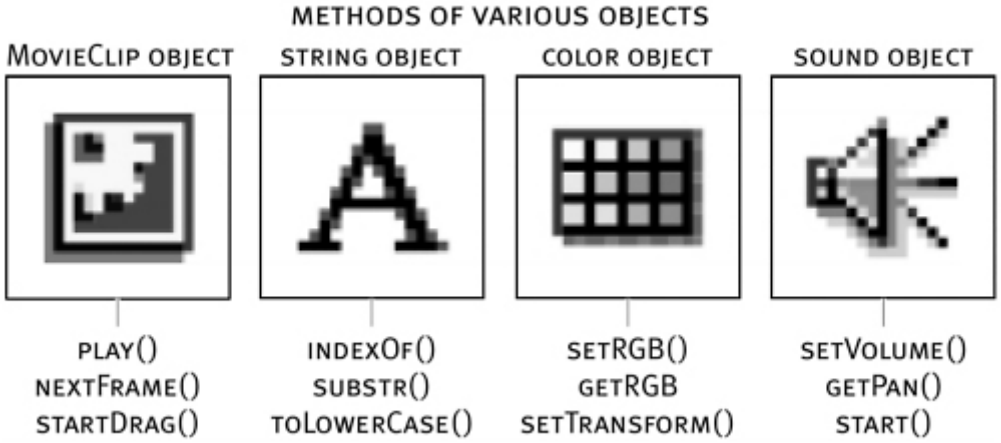
Чтобы вызывать метод объекта, вы должны сначала указать объект, сопровождаемый точкой, тогда название метода:

```
myVCR.record ();
```

Это означает: объект, названный myVCR начинать делать запись.
Круглые скобки, включенные с методом иногда позволяют вам вызывать его в уникальном пути используя параметр или устанавливя одно из значений параметра. Используя наш пример видеомагнитофона снова, скажем, вы хотели делать запись телешоу на канале 8 с 22:00 до 23:00 9 сентября. Сценарий, требуемый исполнять эту задачу мог бы выглядеть следующим образом:

```
myVCR.record (8, 22:00, 23:00, 9 сентября);
```

Как вы можете видеть, запятые отделяют различные параметры метода. Имейте в виду, что значения параметра могут быть интенсивно кодированы , или они могут быть динамическими значениями типа переменных. Вы можете даже использовать другие методы как параметры.
Хотя много объектов ActionScript имеют методы, которые принимают параметры, не все делают. Некоторые просто исполняют задачи, которые не требуют никаких специальных параметров настройки. Например, метод объекта MovieClip stop()просто заставляет график времени останавливаться - ничего больше не требуется.
Каждый объект имеет уникальный набор методов - который имеет смысл, потому что каждый объект имеет определенную природу и используется только в определенных пользователем целях.



Объектные методы в ActionScript исполняют все виды задач, включая следующее:
Получение и установка значений
Выполнение преобразований
Индикация является ли сравнение истинным или ложным
Активизация или дезактивация
Управление кое-чем (строка или число)

ПРИМЕЧАНИЕ

Объект не должен содержать никаких свойств или методов, которые будут использованы объектом. Однако, объект без свойств или методов бесполезен.

Использование стандартного объекта Цвет (COLOR OBJECT)

1. Откройте Clown1.fla из папки Lesson04/Assets.
2. При раскрытой панели Actions выберите красную кнопку и запишите на нее следующий скрипт:

```
on (release) {
 hairColor = new Color(hair);
 hairColor.setRGB(0xCC0000);
}
```

3. При раскрытой панели Actions выберите желтую кнопку и добавьте нижеприведенный скрипт:

```
on (release) {
 hairColor = new Color(hair);
 hairColor.setRGB(0xFFCC00);
}
```

4. При раскрытой панели Actions выберите зеленую кнопку и добавьте нижеприведенный скрипт:

```
on (release) {
 hairColor = new Color(hair);
 hairColor.setRGB(0x009900);
}
```

5. При раскрытой панели Actions выберите синюю кнопку и добавьте нижеприведенный скрипт:

```
on (release) {
 hairColor = new Color(hair);
 hairColor.setRGB(0x336699);
}
```

6. Выберите в меню Control > Test Movie. Пощелкайте по всем четырем кнопкам, для того чтобы пронаблюдать изменения на экране.
7. Закройте окно предварительного просмотра и вернитесь в среду создания проекта. При раскрытой панели Actions выберите кнопку в виде цветов радуги и добавьте нижеприведенный скрипт:

```
on (release) {
 R = random(256);
 G = random(256);
```

```
  B = random(256);
  colorHexString                                          =
R.toString(16)+G.toString(16)+B.toString(16);
  colorHex = parseInt(colorHexString,16);
  hairColor = new Color(hair);
  hairColor.setRGB(colorHex);
}
```

8. Выберите в меню Control > Test Movie, щелкните по кнопке с цветами радуги несколько раз.
9. Закройте тестовое окно и сохраните проект под именем Clown2.fla.

Использование стандартного объекта Ключ для создания интерактивности (KEY OBJECT)

1. Откройте файл balloon1.fla из папки Lesson04/Assets.
2. При раскрытой панели Actions, выберите movie clip с именем hot-air balloon, в виде воздушного шара, и добавьте нижеприведенный скрипт:

```
onClipEvent (load) {
  speed = 3;
}
```

3. Пока MC – воздушный шар еще выбран, добавьте нижеприведенный скрипт в конце предыдущих инструкций:

```
onClipEvent (enterFrame) {
  if (key.isDown(key.RIGHT)) {
    _x += speed;
  } else if (key.isDown(key.LEFT)) {
    _x -= speed;
  }
}
```

4. Добавьте следующий ActionScript в событие на клип - enterFrame:

```
if (key.isDown(key.UP)) {
  _y -= speed;
} else if (key.isDown(key.DOWN)) {
  _y += speed;
}
```

5. Выберите в меню Control > Test Movie. Попробуйте использовать кнопки влево (left), вправо (right), вверх (up) и вниз (down),

выполненные в виде стрелочек на клавиатуре, чтобы двигать и контролировать воздушный шар.

6. Закройте тестовое окно и сохраните проект под именем balloon2.fla.

Использование объекта строка (STRING OBJECT) и объекта выделение (SELECTION OBJECT)

1. Откройте файл wordProcessor1.fla из папки Lesson04/Assets.
2. При раскрытой панели Actions, выберите кнопку перевода теста из строковых в заглавные буквы (выполнена с заглавной буквой A) и добавьте следующий скрипт:

```
on (release) {
 inputField.text = inputField.text.toUpperCase();
}
```

3. При раскрытой панели Actions выберите кнопку для перевода букв из заглавных в строковые (с маленькой буквой a) и добавьте следующий скрипт:

```
on (release) {
inputField.text = inputField.text.toLowerCase();
}
```

4. При раскрытой панели Actions выберите кнопку подсчета числа символов (с надписью '123' на ней) и добавьте следующий скрипт:

```
on (release) {
 status.text = "There are " + inputField.text.length
+ " characters in the
current document";
}
```

5. При раскрытой панели Actions выберите кнопку поиска (с увеличительным стеклом на ней) и добавьте следующий скрипт:

```
on (release) {
 result = inputField.text.indexOf(findField.text);
 if (findField.text != "" && result ) {
  status.text = "The first instance name of these
characters occurs at character
" + result;
  Selection.setFocus("_root.inputField");
  Selection.setSelection(result,      result      +
findField.text.length);
 } else {
```

```
    status.text = "That string could not be found";
  }
}
```

6. Выберите в меню Control > Test Movie.
7. Закройте тестовое окно и сохраните проект под именем wordProcessor2.fla.

Описание и использование функций (FUNCTIONS)

В этом уроке, вы:

- Научитесь создавать и вызывать функции.
- Научитесь добавлять параметры функции.
- Научитесь использовать локальные переменные

Создание функций

1. Откройте файл television1.fla из папки Lesson05/Assets.
2. Выберите первый кадр на слое Actions, находящийся на основной линейке времени (сцене). При раскрытой панели Actions, добавьте следующий ActionScript:

```
tvPower = false;
function togglePower () {
  if (tvPower) {
    newChannel = 0;
    tvPower = false;

  } else {
    tvPower = true;
    newChannel = 1;
  }
  tv.screen.gotoAndStop(newChannel + 1);
  remote.light.play();
}
```

3. Щелкните дважды лкм по пульту дистанционного управления для того, чтобы отредактировать его и просмотреть его содержимое. При раскрытой панели Actions выберите кнопку Power (вкл/выкл) и добавьте этот ActionScript:

```
on (release) {
  _root.togglePower();
}
```

4. Выберите в меню Control > Test Movie. Пощелкайте по кнопке Вкл/Выкл и посмотрите, как она работает.

5. Закройте тестовое окно и сохраните проект под именем television2.fla.

Создание функций с параметрами

1. Откройте television2.fla из папки Lesson05/Assets.
2. Выберите первый кадр на слое Actions, находящийся на основной линейке времени (сцене), и откройте панель Actions. Добавьте этот ActionScript в тот же кадр, в котором вы находитесь после окончания предыдущей функции:

```
function changeTheChannel (newChannel) {
  if (tvPower) {
    currentChannel = newChannel;
    tv.screen.gotoAndStop(newChannel + 1);
    remote.light.play();
  }
}
```

3. Щелкните дважды лкм по movie clip'у с именем remote, находящимся на основной линейке времени (сцене), для того, чтобы отредактировать его и просмотреть его содержимое. При раскрытой панели Actions, выберите круглую кнопку под цифрой 1 на пульте дистанционного управления и добавьте этот ActionScript:

```
on (release) {
  _root.changeTheChannel(1);
}
```

4. Выберите в меню Control > Test Movie. Протестируйте клип.
5. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр на слое Actions.
6. Не закрывая и не сворачивая панель Actions, измените функцию (function) `togglePower()` так, чтобы она выглядела следующим образом:

```
function togglePower () {
  if (tvPower) {
    changeTheChannel(0);
    tvPower = false;
  } else {
    tvPower = true;
    changeTheChannel(1);
  }
}
```

7. Выберите первый кадр на слое Actions, находящийся на основной линейке времени (сцене), и откройте панель Actions. Вставьте следующую строчку ActionScript в этот кадр сразу за строчкой с кодом `tvPower = false;` :

```
numberOfChannels = 6;
```

8. Добавьте следующий ActionScript в окончании предыдущего скрипта в данном кадре:

```
function channelUp () {
  if (currentChannel + 1 <= numberOfChannels) {
    changeTheChannel(currentChannel + 1);
  }
}
```

9. Добавьте следующий ActionScript в текущем кадре, как только предыдущая часть скрипта закончится:

```
function channelDown () {
  if (currentChannel - 1 >= 1) {
    changeTheChannel(currentChannel - 1);
  }
}
```

10. Щелкните дважды лкм по movie clip'у с instance name remote (пульт ДУ), находящийся на основной линейке времени (сцене) для того, чтобы отредактировать его и просмотреть его содержимое. При раскрытой панели Actions выберите кнопку со стрелкой вверх, добавьте этот ActionScript:

```
on (release) {
  _root.channelUp();
}
```

11. При раскрытой панели Actions выберите кнопку со стрелкой вверх и добавьте этот ActionScript:

```
on (release) {
  _root.channelDown();
}
```

12. Выберите в меню Control > Test Movie, затем включите телевизор при помощи пульта ДУ и протестируйте работоспособность кнопок вверх и вниз для того, чтобы изменять каналы на ТВ.

13. Закройте тестовое окно и сохраните проект под именем television3.fla.

Использование локальных переменных (LOCAL VARIABLES) и создание функций, возвращающих результат

1. Откройте файл television3.fla из папки Lesson05/Assets.
2. При раскрытой панели Actions выберите первый кадр и введите следующий ActionScript сразу за фразой `numberOfChannels = 6;`:

```
channelNames =
["","News","Classics","Family","Cartoons","Horror","W
esterns"];
```

3. Пока данный кадр (первый кадр) еще выбран, введите следующий ActionScript в конце предыдущего участка кода:

```
function displayCableText () {
if (currentChannel != 0) {
    var displayText = "You are viewing " +
    channelNames[currentChannel] +
".";
} else {
    var displayText = "";
}
return displayText;
}
```

4. Не закрывая и не сворачивая панель Actions, измените функцию `changeTheChannel()` при помощи вставки следующей линии кода за пятой строчкой внутри определения функции:

```
cableBox.cableDisplay.text = displayCableText();
```

5. Выберите в меню Control > Test Movie. Нажмите кнопку включения телевизора и измените каналы.
6. Закройте тестовое окно и сохраните проект под именем television4.fla.

Создание пользовательских объектов (Objects)

В этом уроке, вы:

- Узнаете, почему объекты так полезны
- Узнаете о создании связей родитель/ребенок, когда один объект содержит другой

- Создадите пользовательский объектный класс
- Поработаете с прототипом объекта класса
- Узнаете о наследовании
- Создадите подкласс объекта
- Узнаете, как просмотреть свойства объекта
- Создатите новые методы объекта
- Расширите методы встроенных объектов Flash
- Создадите новые методы встроенных объектов Flash
- Зарегистрируете мувиклип к пользовательскому объектному классу, чтобы элементы клипа наследовали определенные функциональные возможности

Понимание механизмов создания объектов

Использование объектов как контейнеров (Containers)

1. Отройте Flash и выберите в меню File > New для того, чтобы создать новый файл. При раскрытой панели Actions выберите первый кадр и добавьте следующий скрипт:

```
person1 = new Object();
```

2. Поместите следующие строки скрипта сразу за строчкой, которую вы добавили в предыдущем шаге:

```
person1.gender = "male";
trace (person1.gender);
trace (person1.age);
```

3. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
4. В окне Output, которое откроется автоматически, будет отображено следующее:

```
male
undefined
```

5. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр. Замените три строчки скрипта, созданные в шаге № 2, на следующие строки скрипта:

```
person1.age = 16;
person1.legs = 2;
person1.name = "Justin";
```

6. Добавьте следующее условие `if` в конце предыдущего скрипта и добавьте ActionScript:

```
if (person1.age >= 21){
  trace("I'm going to the Disco");
}else{
  trace("I'm going to BurgerHouse");
}
```

7. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
8. В окне Output, открытом автоматически, будет выведено следующее:

```
I'm going to BurgerHouse
```

9. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр. Замените условие `if`, добавленное в предыдущем шаге, следующими строчками скрипта:

```
person1.head = new Object()
person1.head.eyes = 2;
person1.head.memories = new Array();
```

10. Поместите следующие строки скрипта в конце предыдущего:

```
person1.head.memories[0] = "I was born into the
world.";
person1.head.memories[1] = "I learned the word 'NO.'
It was a very
productive day.";
person1.head.memories[2] = "I learned that if I make
a bunch of noise, I get
what I want.";
```

11. Поместите следующие строки кода в конце предыдущего скрипта:

```
trace(person1.head.memories[1]);
```

Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.

В окне Output, открывающемся автоматически, появится следующий текст:

```
I learned the word 'NO.' It was a very productive
day.
```

12.Закройте тестовое окно и сохраните проект под именем object1.fla.

Отношения между родительским и дочерним объектами (Parent/Child)

1. Open objects1.fla из папки Lesson06/Assets. При раскрытой панели Actions, выберите первый кадр. Замените действие `trace,` которое было добавлено в шаге 9 предыдущей части, и добавьте следующий скрипт по окончании уже имеющегося скрипта:

```
delete person1.head;
trace(person1.head);
trace(person1.head.memories[1]);
trace(person1.name);
```

2. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
3. В окне Output, открывающееся автоматически, будет отображено следующее:

```
undefined
undefined
Justin
```

4. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр. Добавьте следующие строчки скрипта после окончания скрипта `trace(person1.name);` :

```
person1.head = new Object();
trace(person1.head);
trace(person1.head.memories[1]);
```

5. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
6. В окне Output, открывающееся автоматически, будет выведено следующее:

```
undefined
undefined
Justin
[object Object]
undefined
```

7. Закройте тестовое окно и сохраните проект под именем object2.fla.

Создание пользовательского класса и создание экземпляра (INSTANCE NAMES) этого класса

1. Откройте Flash и выберите File > New для создания проекта. При раскрытой панели Actions выберите первый кадр и добавьте нижеприведенный скрипт:

```
Person = function(){
}
Person(); //executes as function
person1 = new Person();//creates an instance name of
the person class
```

2. Модифицируйте class Person и добавьте нижеприведенный скрипт:

```
Person = function(){
  this.name = "Justin";
  this.age = 16;
  this.legs = 2;
  this.head = new Object()
  this.head.eyes = 2;
  this.head.memories = new Array();
}
```

3. Добавьте следующий скрипт line в конце предыдущего участка кода:

```
person1 = new Person();
```

4. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания. При открытом окне просмотра выберите в меню Debug > List Variables.
5. В открывающемся окне Output скрипт, написанный ранее, приведет к отображению следующей информации на дисплее:

```
Variable _level0.person1 = [object #2] {
     name:"Justin",
     age:16,
     legs:2,
     head:[object #3, class 'Object'] {
     eyes:2,
     memories:[object #4, class 'Array'] []
     }
}
```

6. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и измените определение класса Person на следующий:

```
Person = function(name, age){
  this.name = name;
  this.age = age;
  this.legs = 2;
  this.head = new Object()
  this.head.eyes = 2;
  this.head.memories = new Array();
}
```

7. Измените линии скрипта, добавленные в шаге 3 (в первом кадре), и добавьте еще одну строчку:

```
person1 = new Person ("Justin", 16);
person2 = new Person ("Derek", 29);
```

8. Измените класс Person на следующее определение:

```
_global.Person = function(name, age){
  this.name = name;
  this.age = age;
  this.legs = 2;
  this.head = new Object()
  this.head.eyes = 2;
  this.head.memories = new Array();
}
new Person();
```

9. Если бы вы не добавили `_global` внутри класса, то была бы использована адресация на основную временную линейку с использованием root. Необходимо использовать следующий синтаксис:

```
new _root.Person();
```

10. Сохраните ваш проект под именем class1.fla.

Использование наследования в ООП (GENETICALLY PROGRAMMING) и изменение их прототипов (PROTOTYPE)

1. Откройте файл class1.fla из папки Lesson06/Assets. При раскрытой панели Actions выберите первый кадр. Модифицируйте линии скрипта, задающие создание класса (class) Person, на следующие:

```
_global.Person = function(name, age){
  this.name = name;
  this.age = age;
}
Person.prototype.legs = 2;
Person.prototype.head = new Object();
Person.prototype.head.eyes = 2;
Person.prototype.head.memories = new Array();
```

2. Добавьте следующие строчки скрипта в конце предыдущего участка кода:

```
person1 = new Person ("Justin", 16);
person2 = new Person ("Derek", 29);
trace (person1.legs);
trace (person2.legs);
```

3. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания:

В окне Output, открывающемся автоматически, будет выведено следующее:

```
2
2
```

4. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и измените значение `Person.prototype.legs` с 2 на 5. Затем выберите в меню Control > Test Movie для того, чтобы протестировать проект снова.

В окне Output, открывающемся автоматически, будет выведено следующее:

```
5
5
```

5. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions, выберите первый кадр и измените значение `Person.prototype.legs` на 2. Сохраните ваш проект под именем class2.fla.

Создание подклассов (SUBCLASS)

1. Откройте файл class2.fla из папки Lesson06/Assets. При раскрытой панели Actions выберите первый кадр. Замените два оператора `trace` actions в конце скрипта, затем добавьте следующий скрипт:

```
_global.Doctor = function(almaMater){
  this.almaMater = almaMater;
}
Doctor.prototype.title = "Dr.";
```

2. Поместите следующие строчки кода в конце предыдущих инструкций:

```
frankenstein = new Doctor("Transylvania");
trace(frankenstein.almaMater);
trace(frankenstein.title);
trace(frankenstein.legs);
```

3. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания:

В открывшемся автоматически окне Output отобразится следующая информация:

```
Transylvania
Dr.
undefined
```

4. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и поместите следующие строчки скрипта сразу над `Doctor.prototype.title = "Dr."`:

```
Doctor.prototype = new Person();
```

5. Внизу скрипта вы все еще видите операции для вывода информации в окно в режиме трассировки(`trace`), которые были использованы ранее (в шаге 3) для вывода результата:

```
trace(frankenstein.almaMater);//              returned
"Transylvania"
trace(frankenstein.title);// returned "Dr."
trace(frankenstein.legs);// returned undefined
```

6. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания:

В окне Output, открывающемся автоматически, будет выведено следующее:

```
Transylvania
Dr.
2
```

7. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и модифицируйте определение класса Doctor на следующее:

```
_global.Doctor = function(name, age, almaMater){
  this.base = new Person(name, age);
  this.name = this.base.name;
  this.age = this.base.age;
  delete this.base;
  this.almaMater = almaMater;
}
```

8. Измените строчку кода, которая выглядит следующим образом: `frankenstein = new Doctor("Transylvania");` на нижеследующую:

```
frankenstein = new Doctor("Frankenstein", 85,
"Transylvania");
```

9. Поместите следующие строчки кода в конце предыдущих инструкций:

```
trace(frankenstein.name);
trace(frankenstein.age);
```

10. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания:

В окне Output, открывающемся автоматически, будет выведено следующее:

```
Transylvania
Dr.
2
Frankenstein
85
```

11. Закройте тестовое окно и вернитесь в среду создания приложений. Сохраните ваш проект под именем class3.fla.

Использование пользовательских методов (METHODS) для создаваемых классов

1. Откройте файл class3.fla из папки Lesson06/Assets. При раскрытой панели Actions выберите первый кадр. Удалите пять строчек кода, которые участвовали в выводе на экран информации, и затем, в конце скрипта, добавьте следующий скрипт:

```
Person.prototype.talk = function(){
 trace("Hello, my name is " + this.name + ". I am " +
this.age + " years
old.");
}
```

Альтернативный синтаксис для создания пользовательского метода может выглядеть следующим образом:

```
//Define a function first
function soonToBeMethod(){
          trace("Hello, my name is " + this.name + ".
          I am " + this.age + " years
old.");
}
//Set the function as the talk method of the Person
class
Person.prototype.talk = soonToBeMethod;
```

2. Поместите следующий скрипт в конце предыдущего участка кода:

```
person2.talk();
```

3. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе его создания.

В окне Output, открывающемся автоматически, будет выведено следующее:

```
Hello, my name is Derek. I am 29 years old.
```

4. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и измените метод (method) `talk()` на следующий:

```
Person.prototype.talk = function(tone){
 var message = "Hello, my name is " + this.name + ".
I am " + this.age + "
```

```
years old.";
  if (tone == "scream"){
    trace(message.toUpperCase());
  }else{
    trace(message);
  }
}
```

5. Добавьте следующий скрипт, как только закончится предыдущий скрипт:

```
person2.talk("scream");
```

6. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе его создания.

В окне Output, открывающемся автоматически, будет выведено следующее:

```
Hello, my name is Derek. I am 29 years old.
HELLO, MY NAME IS DEREK. I AM 29 YEARS OLD.
```

7. Закройте тестовое окно и вернитесь в среду создания приложений. При раскрытой панели Actions выберите первый кадр и измените строчки с вызовом метода `talk()` (последние две строчки кода в первом кадре (он же текущий кадр)) на следующие:

```
frankenstein.talk();
frankenstein.talk("scream");
```

8. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.

В окне Output, открывающемся автоматически, будет выведено следующее:

```
Hello, my name is Frankenstein. I am 85 years old.
HELLO, MY NAME IS FRANKENSTEIN. I AM 85 YEARS OLD.
```

9. Закройте тестовое окно и вернитесь в среду создания приложений. Сохраните ваш проект под именем class4.fla.

Использование свойств (PROPERTIES)

1. Откройте файл с именем objectProject1.fla из папки Lesson06/Assets.

2. Щелкните дважды лкм по МС'у с именем bigHead для того, чтобы отредактировать его и просмотреть его содержимое.
3. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions выберите МС с instance name bigHead и добавьте следующий скрипт:

```
onClipEvent(load){
 glassesClip._visible = false;
 eyesClip._visible = false;
 noseClip._visible = false;
 mouthClip._visible = false;
 glasses = false;
 eyes = false;
 nose = false;
 mouth = false;
}
```

4. Добавьте следующий скрипт при определении функции (function) ниже, но еще внутри события load (событие загрузки (load event)):

```
function watchGlasses(id, oldval, newval){
_root.message.text = "I now have " + id + ".";
 glassesClip._visible = newval;
 _root.smallHead.glassesClip._visible = oldval;
 _root.glassesButton.enabled = false;
}
```

5. Добавьте следующий скрипт для того, чтобы определить новые функции ниже, но также внутри события load - загрузка (событие загрузки (load event)):

```
function watchEyes(id, oldval, newval){
 _root.message.text = "I now have " + id + ".";
 eyesClip._visible = newval;
 _root.smallHead.eyesClip._visible = oldval;
 _root.eyesButton.enabled = false;
}
function watchNose(id, oldval, newval){
 _root.message.text = "I now have a " + id + ".";
 noseClip._visible = newval;
 _root.smallHead.noseClip._visible = oldval;
 _root.noseButton.enabled = false;
}
function watchMouth(id, oldval, newval){
 _root.message.text = "I now have a " + id + ".";
```

```
  mouthClip._visible = newval;
  _root.smallHead.mouthClip._visible = oldval;
  _root.mouthButton.enabled = false;
}
```

6. Добавьте следующие строчки скрипта ниже, но также внутри события load - загрузка (событие загрузки (load event)):

```
this.watch("glasses", watchGlasses);
this.watch("eyes", watchEyes);
this.watch("nose", watchNose);
this.watch("mouth", watchMouth);
```

7. Выберите кнопку mouthButton, правее MC'a с instance name bigHead, и присоедините этот скрипт:

```
on(release){
  _root.bigHead.mouth = true;
}
```

8. Добавьте следующий скрипт на кнопки:

На кнопку noseButton:

```
on(release){
     _root.bigHead.nose = true;
}
```

На кнопку eyesButton:

```
on(release){
     _root.bigHead.eyes = true;
}
```

На кнопку glassesButton:

```
on(release){
     _root.bigHead.glasses = true;
}
```

9. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
10. Закройте тестовое окно и сохраните проект под именем objectProject2.fla.

Улучшение существующих методов объектов

1. Откройте файл objectProject2.fla из папки Lesson06/Assets.
2. При раскрытой панели Actions выберите первый кадр и добавьте следующий скрипт:

```
MovieClip.prototype.oldDuplicateMovieClip =
MovieClip.prototype.duplicateMovieClip;
MovieClip.prototype.duplicateMovieClip                =
function(name, depth, moveX,
moveY){
  this.oldDuplicateMovieClip(name, depth);
  this._x = moveX;
  this._y = moveY;
}
```

3. При раскрытой панели Actions выберите маленькую кнопку в виде дух квадратов и присоедините этот скрипт:

```
on(release){
smallHead.duplicateMovieClip("myHead1",    10,    200,
150);
}
```

4. Не закрывая и не сворачивая панель Actions выберите большую кнопку в виде двух квадратов и присоедините этот скрипт:

```
on(release){
  bigHead.duplicateMovieClip("myHead2", 20, 100, 200);
}
```

5. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
6. Закройте тестовое окно и сохраните проект под именем objectProject3.fla.

Определение пользовательских методов для реконструированных объектов

1. Откройте файл objectProject3.fla из папки Lesson06/Assets.
2. При раскрытой панели Actions выберите первый кадр и добавьте следующий скрипт в конце предыдущих инструкций:

```
MovieClip.prototype.flip = function(mode){
  if (mode.toLowerCase() == "h"){
    this._xscale = -this._xscale;
```

```
  }else if (mode.toLowerCase() == "v"){
   this._yscale = -this._yscale;
  }
}
```

3. При раскрытой панели Actions выберите кнопку в виде стрелок вверх-вниз, находящуюся рядом с маленькой кнопкой в виде сдвоенного квадрата, и присоедините этот скрипт:

```
on(release){
  smallHead.flip("v");
}
```

4. Не закрывая и не сворачивая панель Actions выберите кнопку в виде стрелок влево-вправо рядом с кнопкой, куда только что вы записали строчки кода, и присоедините этот скрипт:

```
on(release){
  smallHead.flip("h");
}
```

5. Добавьте те же самые строчки скрипта, которые были представлены выше для кнопок в виде стрелочек вверх-вниз и влево-вправо, находящихся рядом с большой кнопкой в виде сдвоенного квадрата. Необходимо только заменить `smallHead` на `bighead` .
6. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
7. Закройте тестовое окно и сохраните проект под именем objectProject4.fla.

Регистрирование классов (REGISTERING CLASSES)

1. Откройте файл registerClass1.fla из папки Lesson06/Assets.
2. Выберите в меню Window > Library для того, чтобы открыть панель библиотеки (Library panel). Найдите movie clip с именем Icons. Щелкните пкм (или Control-click на Mac) на клипе и выберите Linkage (связывание) из выпадающего контекстного меню.

Отметьте, что выбран параметр "Export in first frame".

3. Щелкните по кнопке Cancel (Отмена) для того, чтобы закрыть диалоговое окно Linkage Properties (свойства связывания). Закройте Library panel. Находясь на сцене, щелкните дважды лкм по movie clip’у с instance name Icons для того, чтобы отредактировать его и просмотреть его содержимое.

4. При раскрытой панели Actions выберите первый кадр и добавьте следующий скрипт:

```
_global.Custom1 = function(){
}
Custom1.prototype = new MovieClip()
```

5. Добавьте нижеприведенный скрипт в конце предыдущих инструкций:

```
Custom1.prototype.scale = function(amount){
  this._xscale = amount;
  this._yscale = amount;
}
```

6. Добавьте следующий скрипт, как только закончится предыдущий скрипт:

```
Custom1.prototype.onLoad = function(){
  this.gotoAndStop(this._name);
}
```

7. Добавьте нижеприведенный скрипт, как только закончится предыдущий скрипт:

```
Custom1.prototype.onPress = function(){
  thisX = this._x;
  thisY = this._y;
  startDrag(this, true);
  this.scale(70);
}
```

8. Добавьте нижеприведенный скрипт, как только закончится предыдущий скрипт:

```
Custom1.prototype.onRelease = function(){
  stopDrag();
  if (this.hitTest("_root.bigHead")){
    this._x = thisX;
    this._y = thisY;
    _root.bighead[this._name] = true;
    this.scale(100);
    this.enabled = false;
  }else{
    this._x = thisX;
    this._y = thisY;
```

```
   this.scale(100);
  }
}
```

9. Поместите следующую строчку в конце предыдущего участка кода:

```
Object.registerClass("icons", Custom1);
```

10. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на текущем этапе создания.
11. Закройте тестовое окно и вернитесь в среду создания приложений. Вернитесь внутрь movie clip‘a с именем Icons. При раскрытой панели Actions выберите первый кадр и переместите весь написанный ранее код внутри `#initclip` и `#endinitclip`, как показано ниже:

```
#initclip
//code added in previous steps
#endinitclip
```

12. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на текущем этапе создания.
13. Закройте тестовое окно и вернитесь в среду создания приложений. Перейдите на Сцену 1 (Scene 1). Выберите Window > Library для того, чтобы открыть Library panel, и найдите символ (symbol) movie clip с именем HeadClip внутри данной библиотеки. Щелкните пкм (or Control-click on a Mac) по этому символу и выберите Linkage из выпадающего контекстного меню. После этого выберите опцию Export for ActionScript и дайте ему имя идентификатор (idname - identifier name) headClip.
14. При раскрытой панели Actions выберите кнопку Go to Scene 2 и добавьте следующий скрипт внутри обработчика событий `on(release)` перед тем, как выполнить команду перехода на новую сцену `nextScene();`:

```
_global.Custom2 = function(){
}
Custom2.prototype = new MovieClip();
Custom2.prototype.onEnterFrame = function(){
  this._rotation += 5;
}
Object.registerClass("Icons", Custom2);
Object.registerClass("HeadClip", Custom2);
```

15. Выберите Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.

16. Закройте тестовое окно и сохраните проект под именем registerClass2.fla.

Использование динамических данных (Dynamic Data)

В этом уроке, вы:

- Изучите возможности динамических данных.
- Научитесь создавать поля текста.
- Научитесь создавать переменные.
- Научитесь создавать массивы.
- Научитесь хранить информацию в объектах.
- Научитесь восстанавливать информацию из хранилища динамически.

Создание переменных (VARIABLES)

1. Откройте newsFlash1.fla из папки Lesson07/Assets.
2. Выберите первый кадр на слое Actions находящийся на основной линейке времени (сцене). Откройте панель Actions и введите следующий ActionScript:

```
// ----Monday--------
monday = new Object();
monday.date = "Monday, August 20 2001";
// ----Tuesday--------
tuesday = new Object();
tuesday.date = "Tuesday, August 21 2001";
```

3. Сохраните ваш проект под именем newsFlash2.fla.

Создание массивов (ARRAYS)

4. При открытым newsFlash2.fla выберите первый кадр на слое Actions. Добавьте следующий ActionScript сразу за `monday.date="Monday, August 20 2001` variable set:

```
monday.weather = new Array();
monday.weather[0] = "rainy";
monday.weather[1] = "Very wet";
monday.weather[2] = 85;
monday.weather[3] = 62;
```

5. Пока первый кадр еще выбран, вставьте следующий ActionScript после `tuesday.date="Tuesday, August 21 2001",` :

```
tuesday.weather = new Array();
```

```
tuesday.weather[0] = "sunny";
tuesday.weather[1] = "Beautiful Day!";
tuesday.weather[2] = 90;
tuesday.weather[3] = 73;
```

6. Создайте массив, содержащий новые значения для различных категорий, путем записывания ActionScript сразу за `monday.weather[3]=62`:

```
monday.entertainment = new Array();
monday.entertainment[0] = "MTV is 20!";
monday.entertainment[1] = "The popular TV network MTV
has now been on the
air for 20 years…";
monday.entertainment[2] = "Jobe Makar";
monday.politics = new Array();
monday.politics[0] = "Jesse Ventura Wins";
monday.politics[1] = "Former WWF wrestler Jesse
Ventura is elected governor of Minnesota…";
monday.politics[2] = "Happy Camper";
monday.sports = new Array();
monday.sports[0] = "Head Tennis";
monday.sports[1] = "The Head Atlantis tennis racquet
is one of the most popular racquets in history…";
monday.sports[2] = "Jane Doe";
monday.technology = new Array();
monday.technology[0] = "BajillaHertz Processors!";
monday.technology[1] = "The bajillaHertz processor
has just hit the shelves and is faster than light…";
monday.technology[2] = "John Doe";
```

7. Добавьте новые команды ActionScript сразу после `tuesday.weather[3]=73`:

```
tuesday.entertainment = new Array();
tuesday.entertainment[0] = "Amazing Sci-Fi";
tuesday.entertainment[1] = "Sentrillion Blazers is
the must see sci-fi movie of the year!…";
tuesday.entertainment[2] = "Jobe Makar";
tuesday.politics = new Array();
tuesday.politics[0] = "Tax Refund";
tuesday.politics[1] = "Bush issues large tax
refund…";
tuesday.politics[2] = "John Doe";
tuesday.sports = new Array();
tuesday.sports[0] = "Ryder Cup Begins";
```

```
tuesday.sports[1] = "The European golf tournament you
have been waiting for
has just begun…";
tuesday.sports[2] = "Jane Doe";
tuesday.technology = new Array();
tuesday.technology[0] = "KatrillaHertz Processor";
tuesday.technology[1] = "The katrillahertz processor
just out and is twice
as fast as the bajillahertz chip…";
tuesday.technology[2] = "John Doe";
```

8. Сохраните ваш проект под именем newsFlash3.fla.

Создание динамических текстовых полей (DYNAMIC TEXT FIELDS) и вывод в них информации

1. Пока newsFlash3.fla еще открыт, переместите (playhead) курсор времени на timeline в кадр с меткой (label) refresh.
2. Выберите слой (layer) News Box. Выберите инструмент Текст (Text tool) из панели инструментов (toolbar) и откройте Property inspector. Выберите "Dynamic Text" из выпадающего списка.
3. Создайте текстовое поле (text field) справа от иконки с погодой путем щелчка лкм по сцене. Встройте шрифт (font), для этого щелкните лкм по кнопке Character на Property inspector и затем выбора радио-кнопки All Characters. Подгоните ширину этого text field примерно 150 pixel'ов, измените фонт (font) текста на _sans и с размером шрифта 14 пунктов (pt, пт).
4. Пока text field еще выбран, задайте его instance name в Property inspector, например - weatherBlurb.
5. Пока у вас еще выбран Text tool, щелкните и потяните нажатой лкм для того, чтобы создать еще одно text field, отображаемое светло-голубым цветом линий окантовки на сцене. Измените текущий фонт на _typewriter и размер шрифта на 20, выберите опцию bold (жирный шрифт). Используя Property inspector, выберите опцию center (для выравнивания по центру) для данного текста. Дайте instance name для данного текстового поля (text field) - headline.
6. Щелкните лкм и потяните мышкой для того, чтобы создать еще одно text field под text field headline приблизительно того же размера, после чего текстовое поле приобретет светло-голубую окантовку. Измените текущий шрифт на _sans, и задайте размер 12, после чего поставьте черный цвет (color). Выберите Multiline (многостроковый режим) из выпадающего списка, и затем выберите Word wrap (текст с переносами). Присоедините instance name article для данного text field.

7. Щелкните лкм и потяните для создания еще одного текстового поля (text field), размер которого должен быть примерно 130 pixels по ширине. Текстовое поле будет правее Author под светло-голубой областью. В Property inspector выберите Single Line (в одну строчку). Дайте этому текстовому полю (text field) имя экземпляра (instance name) - author.
8. Щелкните лкм и потяните для создания еще одного текстового поля (text field) в нижней левой части экрана, рядом с темно-серой панелью. Измените текущий шрифт на _sans, белый, полужирный, размер 14 пт. Дайте этому текстовому полю (text field) имя экземпляра (instance name) date.
9. Щелкните лкм и потяните для создания еще одного текстового поля (text field), теперь левее голубой стрелки, другое правее этой же стрелки в разделе Weather на этом сайте. Задайте размер шрифта 13 пт. Задайте instance name low для левого текстового поля и high для правого текстового поля.
10. Сохраните ваш проект под именем newsFlash4.fla.

Поиск данных

1. Проверьте, что newsFlash4.fla еще открыт, выберите первый кадр на слое Actions. Откройте панель Actions и введите следующие две строчки, задающие значения двух переменных:

```
day = "monday";
section = "entertainment";
```

2. Выберите кадр с меткой (frame label) refresh, а затем слой (layer) Actions. Введите данный ActionScript в этот кадр:

```
date.text = this[day].date;
```

3. Пока данный кадр еще выбран, добавьте следующий ActionScript после выше приведенного скрипта:

```
days.gotoAndStop(day);
icon.gotoAndStop(this[day].weather[0]);
weatherBlurb.text = this[day].weather[1];
```

4. Добавьте следующий ActionScript в выбранный кадр:

```
high.text = this[day].weather[2];
low.text = this[day].weather[3];
```

5. Теперь добавьте завершающие действия в выбранный текущий кадр:

```
headline.text = this[day][section][0];
article.text = this[day][section][1];
author.text = this[day][section][2];
```

6. Выберите кадр с меткой (frame label) sit, а затем слой (layer) Actions. При раскрытой панели Actions, добавьте действие `stop();`.
7. Выберите в меню Control > Test Movie.
8. Закройте тестовое окно. Добавьте этот ActionScript:

```
on (release) {
  section = "entertainment";
  this.gotoAndPlay("refresh");
}
```

9. Выберите кнопку Politics и добавьте этот ActionScript:

```
on (release) {
  section = "politics";
  this.gotoAndPlay("refresh");
}
```

10. Выберите кнопку Sports и добавьте этот ActionScript:

```
on (release) {
  section = "sports";
  this.gotoAndPlay("refresh");
}
```

11. Выберите кнопку Technology и добавьте этот ActionScript:

```
on (release) {
  section = "technology";
  this.gotoAndPlay("refresh");
}
```

12. Выберите в меню Control > Test Movie. Пощелкайте по кнопкам категорий для просмотра новостных лент и поизменяйте авторов.
13. Закройте тестовое окно и щелкните дважды лкм по movie clip'у, находящемуся в правом нижнем, углу экрана. На слое (layer) Buttons выберите каждую из кнопок и добавьте следующий скрипт, используя соответствующий день:

```
on (release) {
  _parent.day = "monday";
  _parent.gotoAndPlay("refresh");
}
```

14. Выберите в меню Control > Test Movie. Пощелкайте по M и T, находящимся в правом нижнем углу экрана и отследите изменения, которые при этом происходят. Отметьте, что будет меняться погода на день.
15. Закройте тестовое окно и сохраните проект под именем newsFlash5.fla

Операции с данными (Data)

Операции с числовыми значениями (NUMERICAL DATA) при использовании MATH

1. Откройте файл tempConverter1.fla из папки Lesson08/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт:

```
function changeTemp () {
}
```

3. В начале определения функции (function), которую вы начали только что писать, необходимо создать следующие переменные (variables):

```
boilingPoint = 212;
absoluteZero = -460;
```

4. Для гарантированного ввода температуры в разумных пределах добавьте следующий скрипт ниже строчки `absoluteZero = -460`:

```
if (temperature.text > boilingPoint) {
  temperature.text = boilingPoint;
} else if (temperature.text < absoluteZero) {
  temperature.text = absoluteZero;
}
fahrenheit.text=temperature.text;
```

5. Сразу за условием, добавленным выше, добавьте следующий скрипт, содержащий выражение (формулу) для перевода значений температуры из системы Фаренгейта (Fahrenheit) в значения по Цельсию (Celsius):

```
celsius.text = Math.round((5 / 9) * fahrenheit.text -
17.777);
```

6. Добавьте следующие строчки скрипта, как только закончится предыдущий скрипт:

```
scalePercent = (Math.abs(absoluteZero - Fahrenheit.text) /
Math.abs(absoluteZero - boilingPoint)) * 100;
mercury._yScale=scalePercent;
```

7. При раскрытой панели Actions выберите кнопку (button) Convert и добавьте следующий скрипт:

```
on (release, keyPress "<Enter>") {
  changeTemp();
}
```

8. Выберите в меню Control > Test Movie. Ведите значения температур и нажимайте на соответствующие кнопки.
9. Закройте тестовое окно и сохраните проект под именем tempConverter2.fla.

Операции со строками (STRINGS)

1. Откройте файл madlibs1.fla из папки Lesson08/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт:

```
function generate () {
}
```

3. Пока первый кадр еще выбран, добавьте следующий скрипт, содержащий четыре строчки, которые будут использованы при определении функции (function) `generate()`:

```
verb1.text = verb1.text.toLowerCase();
verb2.text = verb2.text.toLowerCase();
adjective.text = adjective.text.toLowerCase();
noun.text = noun.text.toLowerCase();
```

4. Добавьте следующий скрипт в конце текущей функции `generate()` function, сразу за `noun.text = noun.text.toLowerCase()`:

```
properNoun.text =
properNoun.text.substr(0,1).toUpperCase() +
properNoun.text.substr(1).toLowerCase();
```

5. Добавьте следующий скрипт в качестве финишной черты для функции (function) `generate()`:

```
paragraph.text = "You " + verb1.text + " into love
with " + properNoun.text
+ " when you " + verb2.text + " " + properNoun.text +
" eating a " +
adjective.text + " " + noun.text + ".";
```

6. При раскрытой панели Actions выберите кнопку (button) Submit и добавьте следующий скрипт:

```
on (release, keyPress "<Enter>") {
  generate();
}
```

7. Выберите в меню Control > Test Movie.
8. Закройте тестовое окно и сохраните проект под именем madlibs2.fla.

Использование условной логики (Conditional Logic)

Использование реакции на несколько условий (CONDITIONS)

1. Откройте файл rocketLaunch1.fla из папки (folder) Lesson09/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name weather и добавьте следующий скрипт:

```
onClipEvent (load) {
  randomWeather = random (3);
  if (randomWeather == 0) {
    conditions = "Sunny";
    _root.rocket.noThrust = 3;
    _root.rocket.thrust = 6;
  } else if (randomWeather == 1) {
    conditions = "Rainy";
    _root.rocket.noThrust = 2;
    _root.rocket.thrust = 4;
  } else {
    conditions = "Night";
    _root.rocket.noThrust = 1;
    _root.rocket.thrust = 2;
  }
}
```

3. При раскрытой панели Actions добавьте следующий скрипт внутри события загрузки - `событие загрузки (load event)`, чуть ниже последней части условия `if`:

```
gotoAndStop (conditions);
_root.weatherText.text = "Weather: " + conditions;
```

4. Выберите в меню Control > Test Movie для проверки функционирования вашего проекта на данном этапе его создания.
5. Находясь в режиме тестирования, выберите Debug > List Variables.
6. Закройте окно предварительного тестирования и вернитесь в среду создания приложений, затем сохраните ваш проект под именем rocketLaunch2.fla.

Определение границ

1. Откройте файл rocketLaunch2.fla из папки Lesson09/Assets.
2. При раскрытой панели Actions выберите одну из двух красных полосок наверху экрана и добавьте нижеприведенный скрипт:

```
onClipEvent (enterFrame) {
  if (_x < 0) {
    direction = "right";
  } else if (_x > 550) {
    direction = "left";
  }
  if (direction == "right") {
    _x = _x + 3;
  } else {
    _x = _x - 3;
  }
}
```

3. Выберите в меню Control > Test Movie для проверки функционирования проекта.
4. Закройте окно предварительного тестирования, вернитесь в среду создания приложений и сохраните ваш проект под именем rocketLaunch3.fla.

Переключение питания Вкл/Выкл

1. Откройте файл rocketLaunch3.fla из папки Lesson09/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name rocket и добавьте следующий скрипт:

```
onClipEvent (load) {
```

```
  speed = noThrust;
}
```

3. Добавьте следующий скрипт ниже скрипта, который вы только что добавили:

```
onClipEvent (enterFrame) {
  if (launch) {
    _y = _y - speed;
    if (_y < _root._y) {
      launch = false;
      gotoAndStop ("off");
      _root.status.gotoAndStop ("success");
      _root.sounds.gotoAndPlay ("success");
    }
  }
}
```

4. При раскрытой панели Actions выберите кнопку Launch (запуск) и добавьте нижеприведенный скрипт:

```
on (release) {
  rocket.launch = true;
  rocket.gotoAndStop ("on");
  sounds.gotoAndPlay ("launch");
}
```

5. Добавьте нижеприведенный скрипт в окончание написанного выше скрипта:

```
on (press) {
  rocket.launch = false;
  rocket.gotoAndStop ("off");
  status.gotoAndStop ("off");
  sounds.gotoAndPlay ("intro");
  rocket._x = 98;
  rocket._y = 352;
}
```

6. Выберите в меню Control > Test Movie для тестирования функционирования проекта.
7. Закройте окно предварительного тестирования, вернитесь в среду создания приложений, сохраните ваш проект под именем rocketLaunch4.fla.

Реакция на действия пользователей (USER INTERACTION)

1. Откройте файл rocketLaunch4.fla из папки Lesson09/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name rocket и добавьте следующий скрипт сразу, как только предыдущий скрипт завершится:

```
onClipEvent (keyDown) {
  if (launch && key.isDown(key.SPACE)) {
  speed = thrust;
  _root.thrustBoost.gotoAndStop ("on");
  }
}
```

3. Добавьте следующий скрипт сразу за участком кода, который был написан на предыдущем этапе создания проекта:

```
onClipEvent (keyUp) {
  if (!key.isDown(key.SPACE)) {
  speed = noThrust;
  _root.thrustBoost.gotoAndStop ("off");
  }
}
```

4. Выберите в меню Control > Test Movie для тестирования функциональности проекта на данном этапе его создания.
5. Закройте окно предварительного тестирования, вернитесь в среду создания приложений и сохраните ваш проект под именем rocketLaunch5.fla.

Определение столкновения (COLLISION)

1. Откройте фай rocketLaunch5.fla из папки Lesson9/Assets.
2. Щелкните дважды лкм по одной из красных полосок, находящихся вверху сцены, для редактирования содержимого МС'а и просмотра его содержимого.
3. При раскрытой панели Actions выберите один из экземпляров МС'а в виде красных полосок и добавьте следующий скрипт:

```
onClipEvent (enterFrame) {
  if (hitTest("_root.rocket")) {
    _root.rocket._x = 98;
    _root.rocket._y = 352;
    _root.rocket.launch = false;
    _root.rocket.gotoAndStop ("off");
    _root.status.gotoAndStop ("abort");
```

```
    _root.sounds.gotoAndPlay ("abort");
  }
}
```

4. При раскрытой панели Actions выберите другой экземпляр MC'a и поместите на нее аналогичный вышеизложенному скрипт.
5. Выберите в меню Control > Test Movie для тестирования проекта на текущем этапе его создания.
6. Закройте окно предварительного тестирования, вернитесь в среду создания приложений и сохраните ваш проект под именем rocketLaunch6.fla.

Автоматическое исполнение скриптов при помощи циклов (Loops)

В этом уроке вы:

- Узнаете о том, как полезно использовать циклы
- Узнаете о типах циклов
- Установите условия циклов
- Создадите вложенный цикл
- Используете прерывания циклов

Написание и использование условий циклов (LOOP CONDITIONS)

1. Откройте файл pictureShow1.fla из папки Lesson10/Assets.
2. При раскрытой панели Actions выберите клип с instance name dropDownList и добавьте следующий скрипт:

```
onClipEvent (load) {
  buttonNames = ["Paris","New York","London"];
  item._visible = false;
}
```

3. Сразу за последней строчкой ActionScript внутри события загрузки клипа `load` clip event введите следующую функцию:

```
function populateList () {
  spacing = item._height + 2;
  numberOfButtons = buttonNames.length;
}
```

4. Добавьте следующий ActionScript в функцию (function) `populateList()` сразу за ее последней строчкой кода:

```
var i = -1;
while (++i < numberOfButtons) {
  name = "item" + i;
  item.duplicateMovieClip(name, i);
  this[name].itemName.text = buttonNames[i];
  this[name]._x=0;
  this[name]._y = i * spacing;
  this[name].pictureID = i + 1;
}
```

5. Щелкните дважды лкм клип с instance name dropDownList для того, чтобы отредактировать его и просмотреть его содержимое. Выберите кнопку (button) Menu, и при раскрытой панели Actions добавьте следующий скрипт

```
on (release) {
  populateList();
}
```

6. Выберите в меню Control > Test Movie. Щелкните лкм по кнопке (button) Menu для того, чтобы протестировать ваш скрипт.
7. Закройте тестовое окно и сохраните проект под именем pictureShow2.fla.

Использование вложенных циклов (NESTED LOOPS)

1. Откройте файл pictureShow2.fla из папки Lesson 10/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name dropDownList и добавьте следующий скрипт в виде функции (function) внутри события загрузки (событие загрузки (load event)):

```
function itemClicked (pictureID) {
  picToDuplicate = "pictures" + pictureID;
  xSpacing = 160;
  ySpacing = 120;
  xStart = 190;
  yStart = 30;
}
```

3. Добавьте следующий скрипт внизу функции `itemClicked()`, включая его тем самым внутрь определения функции function:

```
v = 0;
i = -1;
while (++i < 2) {
j = -1;
```

```
while (++j < 2) {
     ++v;
     name = "picture" + v;
     _root[picToDuplicate].duplicateMovieClip(name,
     v);
     _root[name]._x = xStart + i * xSpacing;
     _root[name]._y = yStart + j * ySpacing;
     _root[name].gotoAndStop(v);
}
}
```

4. Создайте функцию (function) `removeButtons()` в конце действия обработчика событий `событие загрузки (load event)` для текущего (выбранного ранее) movie clip'a:

```
function removeButtons() {
  var i = -1;
  while (++i < numberOfButtons) {
    name = "item" + i;
    this[name].removeMovieClip();
  }
}
```

5. Щелкните дважды лкм по movie clip'y с instance name dropDownList для того, чтобы отредактировать его и просмотреть его содержимое. Щелкните дважды лкм по movie clip'y с instance name item для того, чтобы также отредактировать его и просмотреть его содержимое. Выберите белую кнопку, после чего откройте панель Actions и добавьте следующий скрипт:

```
on (release) {
  _parent.itemClicked(pictureID);
}
```

6. Выберите в меню Control > Test Movie для тестирования на работоспособность вашей работы.
7. Закройте тестовое окно и сохраните проект под именем pictureShow3.fla.

Выходы из цикла (LOOP EXCEPTIONS)

1. Откройте файл с именем phoneNumberSearch1.fla из папки Lesson10/Assets.
2. При раскрытой панели Actions выберите первый кадр в слое (layer) Actions и добавьте следующий скрипт:

```
info = [["John","919-555-5698"],["Kelly","232-555-
3333"],["Ross","434-555-
5655"]];
```

3. Добавьте следующий скрипт определения функции (function) сразу под массивом `info`:

```
function search () {
 matchFound = false;
 i = -1;
 while (++i < info.length) {
 }
}
```

4. Добавьте следующий скрипт в цикл `while` в функцию (function) `search()`:

```
if (info[i][0].toLowerCase() !=
name.text.toLowerCase()) {
 continue;
 }
 result.text = info[i][1];
 matchFound = true;
 break;
}
```

5. Добавьте последнее условие `if` в качестве последнего действия функции (function) `search()`:

```
if (!matchFound) {
 result.text = "No Match";
}
```

6. При раскрытой панели Actions выберите невидимую кнопку (button) invisible над "search" и добавьте следующий скрипт:

```
on (release, keyPress "<Enter>") {
 search();
}
```

7. Выберите в меню Control > Test Movie. Введите John, Kelly, или Ross в поле поиска и нажмите кнопку поиска - Search. Введите любое другое имя и затем снова нажмите кнопку поиска.
8. Закройте тестовое окно и сохраните проект под именем phoneNumberSearch2.fla.

Использование XML и Flash

В этом уроке вы:

- Кратко изучите формат XML.
- Научитесь отправлять и загружать XML с сервера.
- Научитесь создавать новые объекты XML.
- Изучите грамматический разбор XML документа.
- Научитесь использовать методы, свойства и события объекта XML.
- Научитесь соединяться с простым socket сервером используя Flash.

Использование объекта XML (XML OBJECT)

1. Откройте файл с именем LoginRegister1.fla из папки Lesson12/Assets.
2. При раскрытой панели Actions выберите первый кадр и добавьте команду `stop()`.
3. При раскрытой панели Actions на шкале времени (timeline) выберите первый кадр, затем выберите кнопку (button) Login и добавьте нижеприведенный скрипт:

```
on (release) {
  _root.gotoAndStop("login");
}
```

4. При раскрытой панели Actions на шкале времени (timeline) на первом кадре, выберите кнопку (button) Register и добавьте нижеприведенный скрипт:

```
on (release) {
  _root.gotoAndStop("register");
}
```

5. Переместите курсор времени (playhead) на кадр с меткой (label) Login. На этой метке (label) вы сможете найти два текстовых поля (text field) - userName и password, а также кнопку на сцене. При раскрытой панели Actions выберите кадр на слое (layer) Actions на данной метке и добавьте следующие строчки скрипта:

```
loginURL                                          =
"http://yourdomain.com/projects/tfts/using_xml/UserLo
gin.asp";
```

6. Пока еще выбран тот же кадр (frame), добавьте функцию (function), приведенную ниже:

```
function loginSubmit () {
```

```
  xmlToSend =
"<Login><UserName>"+userName.text+"</UserName><Passwo
rd>"+password.text+"</Password></Login>";
  objToSend = new XML(xmlToSend);
  objToReceive = new XML();
  objToReceive.onLoad = loginResponse;
  objToSend.sendAndLoad(loginURL, objToReceive);
  _root.gotoAndStop("waiting");
}
```

7. Добавьте это определение функции (function) сразу за вышеприведенным скриптом:

```
function loginResponse () {
  var                        response                        =
objToReceive.firstChild.firstChild.firstChild.nodeVal
ue;
  if (response == "Login Correct") {
    _root.gotoAndStop("login success");
  } else if (response == "Login Incorrect") {
    _root.gotoAndStop("login failed")
  }
}
```

Запомните, что отклик от UserLogin.asp имеет следующий формат:

```
<Login>
  <Message>Login Correct|Login Incorrect</Message>
</Login>
```

8. Выберите кнопку (button) Отправить - Submit и вставьте в нее вызов функции:

```
on (release, keyPress "<Enter>") {
  loginSubmit();
}
```

9. Переместите курсор времени (playhead) в кадр с меткой (label) register. На этой метке (label) вы найдете три текстовых поля (text field) - userName, email и password, а также кнопку (button) на сцене (stage). При раскрытой панели Actions выберите кадр (frame) на слое (layer) Actions на метке (label), о которой только что шла речь, и добавьте следующий скрипт:

```
registrationURL                                          =
"http://yourdomain.com/using_xml/AddUser.asp";
```

10. Пока этот кадр еще выбран добавьте в него определение функции (function):

```
function registrationSubmit () {
xmlToSend="<Register><UserName>"+userName.text+"</Use
rName><Email>"+email.te
xt+"</Email><Password>"+password.text+"</Password></R
egister>";
 objToSend = new XML(xmlToSend);
 objToReceive = new XML();
 objToReceive.onLoad = registrationResponse;
 objToSend.sendAndLoad(registrationURL,
objToReceive);
 _root.gotoAndStop("waiting");
}
```

11. Добавьте в него определение функции (function) сразу за после вышеперечисленного:

```
function registrationResponse () {
 var                    response                    =
objToReceive.firstChild.firstChild.firstChild.nodeVal
ue;
 if (response == "User Inserted") {
  _root.gotoAndStop("registration success");
 } else if (response == "User Exists") {
  _root.gotoAndStop("registration failed")
 }
}
```

Запомните, что страница AddUser.asp возвращает документ в следующем формате:

```
<Register>
 <Message>User Inserted|User Exists</Message>
</Register>
```

12. Добавьте следующий вызов функции (function call) на кнопку (button) Submit:

```
on (release, keyPress "<Enter>") {
 registrationSubmit();
}
```

13. Выберите в меню Control > Test Movie для тестирования на работоспособность вашей работы. Нажмите кнопку (button) Register

и отошлите некоторую информацию. Откройте снова файл и повторите попытку залогиниться.

14. Закройте тестовое окно и сохраните проект под именем loginRegister2.fla

Введение в Сокет Сервер (SOCKET SERVERS)

1. Загрузите и установите JRE (для пользователей Windows), например с http://www.sun.com. Выберите Downloads > Java Technology > Java 2 Runtime Environment v 1.3.1 (или выше). Загрузите инсталлятор (installer) и инструкции по установке, которым желательно следовать.
2. Скопируйте все файлы урока на папку на жестком диске (hard drive, HDD). Откройте MS-DOS prompt (Приглашение MS-DOS) или Command Prompt (Командную строку) в зависимости от того, какой версией Windows вы пользуетесь. Выполняется это следующим образом: Start > All Programs (или Program Files) > Accessories > Command Prompt (или MS-DOS).
3. Перейдите в директорию (папку, directory), содержащую все файлы к данному уроку, путем набора команды cd и полного пути к этой директории, например:

cd C:\Documents and Settings\Jobe Makar\Desktop\FlashFiles). Затем нажмите клавишу Enter.

4. В консольном окне наберите java AquaServer 9999 и нажмите клавишу Enter.

   ElectroServer находится по следующему адресу:

   www.electrotank.com/ElectroServer

   Unity находится по следующему адресу: www.moock.org/unity/

Использование объекта XML Socket (XMLSOCKET OBJECT)

1. Откройте файл chat1.fla из папки Lesson12/Assets.
2. При раскрытой панели Actions выберите первый кадр и добавьте нижеприведенный скрипт:

```
stop();
function setChatText (msg) {
  chatText += msg;
  chatBox.htmlText = chatText;
  scrollBar.setScrollPosition(chatBox.maxscroll);
}
```

3. Добавьте определение функции (function) в тот же кадр, сразу за функцией (function), определенной в предыдущем шаге:

```
function initializeChat (itWorked) {
  if(itWorked) {
    _root.gotoAndStop("login");
  } else {
    _root.gotoAndStop("failed");
  }
}
```

4. Теперь добавьте следующий скрипт определение функции (function) в тот же кадр, сразу за функцией определенной в предыдущем шаге создания проекта:

```
function received (info) {
  var                          from                          =
info.firstChild.firstChild.attributes.from;
  var                        message                         =
info.firstChild.firstChild.firstChild.nodeValue;
  setChatText("<BR><B><FONT FACE=\"arial\" SIZE=\"15\"
COLOR=\"#333333\">"+from     +"</B></FONT>:     <FONT
FACE=\"arial\" SIZE=\"15\"
COLOR=\"#999999\">"+message+"</FONT>");
}
```

5. Данные принимаются сервером (сообщения чата – chat message) идентично тому, как данные передаются на сервер. Они должны быть отформатированы следующим образом:

```
<doc>
  <message from="jobe">Hello world</message>
</doc>
```

6. Для того, чтобы пользователь узнал о разрыве связи (коннекции, connection), добавьте это определение функции (function) в текущий кадр:

```
function connectionClosed () {
  setChatText("<BR><B><FONT FACE=\"arial\" SIZE=\"15\"
COLOR=\"#666666\">The
connection has been terminated.</B></FONT>");
}
```

7. Для разрыва коннекции, инициированного пользователем, добавьте это определение функции (function):

```
function closeConnection () {
  server.close();
}
```

8. Добавьте это определение функции (function) в тот же кадр:

```
function send (messageToSend) {
  server.send("<doc><message
from=\""+screenName+"\">"+messageToSend+"</message></
doc>");
}
```

9. Этот текст форматируется по образу XML документа, выглядящего следующим образом:

```
<doc>
  <message from="jobe">Hello</message>
</doc>
```

10. Добавьте следующие строчки скрипта сразу за участком кода, добавленного вами на предыдущем шаге:

```
XML.prototype.ignorewhite = true;
server = new XMLSocket();
server.onConnect = initializeChat;
server.onXML = received;
server.onClose = connectionClosed;
server.connect("localhost", 9999);
```

11. Переместите курсор времени (playhead) в кадр с меткой (label) failed. При раскрытой панели Actions выберите кнопку (button) Try Again и присоедините этот скрипт:

```
on (release) {
  _root.gotoAndStop(1);
}
```

12. Переместите курсор времени (playhead) в кадр с меткой (label) login, в котором вы найдете текстовое поле (text field) с названием nameField и кнопку (button) на сцене (stage). При раскрытой панели Actions выберите кадр на слое (layer) Actions, имеющий метку login, и добавьте следующий скрипт:

```
selection.setFocus("nameField");
```

13. Пока курсор времени (playhead) выбран и выбран кадр с меткой login, а также раскрыта панель Actions выберите кнопку (button) Submit, и добавьте этот ActionScript:

```
on (release, keyPress "<Enter>") {
  if (nameField.text != "") {
    screenName = nameField.text;
    _root.gotoAndStop("chat");
  }
}
```

14. Переместите курсор времени (playhead) в кадр с меткой (label) chat, в котором вы найдете два текстовых поля (text field) - chatBox и chatMessage и две кнопки (button) на сцене (stage). При раскрытой панели Actions выберите кадр (frame) на слое (layer) Actions с указанной меткой (label), и добавьте следующие строчки скрипта:

```
setChatText("<FONT FACE=\"arial\" SIZE=\"20\"
COLOR=\"#666666\"><b>You have just joined the
chat!</b></FACE>");
selection.setFocus("chatMessage");
```

15. При раскрытой панели Actions выберите кнопку (button) Send, и добавьте нижеприведенный скрипт:

```
on (release,keyPress "<enter>") {
  if (chatMessage.text != "") {
    send(chatMessage.text);
    chatMessage.text = "";
  }
}
```

16. При раскрытой панели Actions выберите кнопку (button) Close Connection, находящуюся на сцене (stage), и добавьте нижеприведенный скрипт:

```
on (release) {
  closeConnection();
}
```

17. запустите AquaServer на порту (port) 9999.
18. Выберите в меню Control > Test Movie для тестирования того, что было только что создано: залогиньтесь (Log in) и отправьте несколько сообщений в чат (chat).
19. Сохраните ваш проект под именем chat2.fla

Проверка правильности (Validating) и форматирование (Formatting) данных (Data)

В этом уроке вы узнаете:

- определение требования подтверждения
- задание метода, чтобы обращаться с ошибками, найденными в процессе подтверждения
- создавать функции для того, чтобы утвердить строки, числа, и последовательности
- форматировать динамические данные, используя HTML
- создавать динамическе области текста
- форматировать текст, используя объекты TextFormat

Ошибки управления (HANDLING ERRORS)

1. Откройте файл validate1.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте нижеприведенный скрипт:

```
stop ();
errors = new Array();
```

3. Добавьте это определение функции (function) как только закончится предыдущий скрипт:

```
function clearForm () {
  name.text = "";
  email.text = "";
  state.text = "";
  zip.text = "";
  errorLog.text = "";
  errors.length = 0;
}
```

4. Не закрывая и не сворачивая панель Actions выберите кнопку (button) Clear, находящуюся внизу экрана, и добавьте нижеприведенный скрипт:

```
on (release) {
  clearForm();
}
```

5. Сохраните этот файл под именем validate2.fla.

Проверка правильности строк (STRING)

1. Откройте файл validate2.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте это определение функции (function) в конце предыдущих инструкций:

```
function validateName () {
  if (name.text.length < 2 || isNaN(name.text) == 
false) {
    errors.push("Please enter a valid name.");
  }
}
```

3. Не закрывая и не сворачивая панель Actions, добавьте это определение функции (function) в конце предыдущих инструкций:

```
function validateForm () {
  errorLog.text = "";
  errors.length = 0;
  validateName();
  if (errors.length > 0) {
    errorLog.htmlText = "<b>These errors were 
found:</b><br>";
    var i = -1;
    while (++i < errors.length) {
      errorLog.htmlText += errors[i] + newline;
    }
  } else {
    gotoAndStop ("Confirm", 1);
  }
}
```

4. Не закрывая и не сворачивая панель Actions, выберите кнопку (button) Submit внизу экрана и добавьте нижеприведенный скрипт:

```
on (release) {
  validateForm();
}
```

5. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
6. Закройте тестовое окно и вернитесь в среду создания приложений, сохраните этот файл под именем validate3.fla.

Проверка правильности последовательностей (SEQUENCES)

1. Откройте файл validate3.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте это определение функции (function) в конце предыдущих инструкций:

```
function validateEmail () {
 if (email.text.indexOf("@") < 2) {
  errors.push(" \"@\" missing in email or in the
wrong place.");
 }
 if          (email.text.lastIndexOf(".")          <=
(email.text.indexOf("@") + 2)) {
  errors.push(" \".\" missing in email or in the
wrong place.");
 }
 if (email.text.length < 8) {
  errors.push("Email address not long enough.");
 }
}
```

3. Добавьте следующий скрипт вызова функции (function call) сразу за строчкой `validateName()` вызова функции (function call) в определении функции (function) `validateForm()`:

```
validateEmail();
```

4. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
5. Закройте тестовое окно и вернитесь в среду создания приложений, сохраните этот файл под именем validate4.fla.

Проверка правильности (VALIDATING) против списка выбора (LIST OF CHOICES)

1. Откройте файл validate4.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте это определение функции (function) в конце предыдущих инструкций:

```
function validateState () {
 states = ["California", "Indiana", "North Carolina",
"Oklahoma"];
 matchFound = false;
 i = -1;
```

```
  while (++i < states.length) {
    if (state.text == states[i]) {
      matchFound = true;
    }
  }
  if (!matchFound) {
    errors.push("Please enter a valid state.");
  }
}
```

3. Добавьте следующий скрипт вызова функции (function call) сразу за вызовом функции (function call) `validateEmail()`:

```
validateState();
```

4. Выберите в меню Control > Test Movie для того, чтобы протестировать проект.
5. Закройте тестовое окно и вернитесь в среду создания приложений, сохраните этот файл под именем validate5.fla.

Проверка правильности (VALIDATING) чисел (NUMBERS)

1. Откройте файл validate5.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions, выберите первый кадр на слое (layer) Actions и добавьте это определение функции (function) в конце предыдущих инструкций:

```
function validateZip () {
  if (zip.text.length != 5 || isNaN(zip.text) == true)
{
    errors.push("Please enter a valid zip.");
  }
}
```

3. Добавьте следующий скрипт вызова функции (function call) сразу за вызовом функции (function call) `validateState(`:

```
validateZip();
```

4. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
5. Закройте тестовое окно и вернитесь в среду создания приложений, сохраните этот файл под именем validate6.fla.

Динамическое форматирование текста с использованием HTML

1. Откройте файл validate6.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions, выберите первый кадр на слое Actions и вставьте следующие строчки скрипта после выражения `}else{`, но выше строчки с выражением `gotoAndStop ("Confirm", 1);` внутри определения функции (function)`validateForm()`:

```
name = name.text;
email = email.text;
state = state.text;
zip = zip.text;
```

3. При открытой панели Сцена (Scene panel) найдите сцену (scene) Confirm.
4. При открытом инспекторе свойств (Property inspector) выберите confirmText текстовое поле (text field), находящееся в центре сцены (stage).
5. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте нижеприведенный скрипт:

```
confirmMessage = new Array();
```

6. Добавьте нижеприведенный скрипт сразу за текущей строчкой скрипта:

```
confirmMessage.push("<font size = \"15\" color =
\"#FF9900\"><b>Thank you " + name +
"!</b></font><br>");
```

7. Добавьте следующий скрипт сразу за текущей строчкой скрипта:

```
confirmMessage.push("This product has been registered
to <font color =
\"#3399CC\"><u><a href = \"mailto:" + email + "\">" +
name +
"</a></u></font>. ");
```

8. Добавьте следующий скрипт сразу за текущей строчкой скрипта:

```
confirmMessage.push("We hope you enjoy using
<b><i>NotSoSoft LastPage</i></b> as much as we
enjoyed developing it for you.<br>");
```

9. Добавьте следующий скрипт сразу за текущей строчкой скрипта:

```
confirmMessage.push("There are many users in the
state of <b>" + state +
"</b> that have had great success with our
product.<br>");
```

10. Добавьте следующий скрипт сразу за текущей строчкой скрипта:

```
confirmMessage.push("For more information about this
product, please visit
our website at <font color = \"#3399CC\"><u><i><a
href
=\"http:\\www.notsosoft.com\">www.notsosoft.com</a></
i></u></font>.");
```

Эти строчки будут преобразованы в следующее:

```
For more information about this product, please visit
our site at <font
color = "#3399CC"><u><i><a
href="http:\\www.notsosoft.com">
www.notsosoft.com</a></i></u></font>.
```

11. Добавьте следующий скрипт сразу за текущей строчкой скрипта:

```
i = -1;
while (++i < confirmMessage.length) {
  confirmText.htmlText += confirmMessage[i];
}
```

12. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
13. Закройте тестовое окно и вернитесь в среду создания приложений, сохраните этот файл под именем validate7.fla.

Создание и управление текстовым полем (TEXT FIELD) с использованием ACTIONSCRIPT

1. Откройте файл flashWriter1.fla из папки Lesson13/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте нижеприведенный скрипт:

```
_root.createTextField("movingField", 10, 150, 80,
200, 20);
with (movingField){
```

```
  autoSize = true;
  html= true;
  multiline = true;
  text = "Enter text here";
  type = "input";
  wordWrap = true;
}
```

3. Добавьте следующий скрипт ниже конца скрипта, добавленного на предыдущем шаге создания проекта:

```
_root.createTextField("statusField",  20,  150,  80,
100, 20);
with (statusField){
  autoSize = true;
  background = true;
  html = true;
  multiline = false;
  selectable = false;
  text = "0";
  type = "dynamic";
  wordWrap = false;
}
```

4. Добавьте следующий скрипт определения функции (function) сразу за окончанием скрипта, добавленного на предыдущем шаге создания проекта:

```
function updateStatus(){
  statusField._x = movingField._x;
  statusField._y       =       (movingField._y       +
movingField._height) + 10;
  statusField.text = movingField.length;
}
```

5. Добавьте следующий скрипт определения функции (function) сразу за окончанием скрипта, добавленного на предыдущем шаге создания проекта:

```
function reshapeBox(){
  with(box){
    _x = movingField._x;
    _y = movingField._y;
    _width = movingField._width;
    _height = movingField._height;
  }
```

```
}
```

6. Добавьте следующий скрипт вызова функции (function call) сразу за окончанием скрипта, добавленного на предыдущем шаге создания проекта:

```
updateStatus();
reshapeBox();
```

7. Добавьте следующий скрипт сразу за вызовом функции (function call), добавленной на предыдущем шаге создания проекта:

```
movingField.onChanged = function(){
  movingField._x += 4;
  if (movingField._x + movingField._width > 500){
    movingField._x = 150;
  }
  reshapeBox();
  updateStatus();
}
```

8. Выберите в меню Control > Test Movie для просмотра проекта на данном этапе его создания.
9. Закройте тестовое окно и сохраните проект под именем flashWriter2.fla.

Использование объекта форматирования текста (TEXTFORMAT OBJECT)

1. Откройте файл flashWriter2.fla из папки Lesson13/Assets.
2. При открытом инспекторе свойств (Property inspector) выберите компоненту (component) Checkbox.
3. Например, если компонента выбрана, следующий скрипт будет задавать значение (value) `true` (истина) для переменной (variable) `currentValue`:

```
currentValue = applyToAll.getValue();
```

4. При раскрытой панели Actions выберите первый кадр на слое (label) Actions и добавьте нижеприведенный скрипт, как только закончится предыдущий скрипт:

```
myFonts = TextField.getFontList();
```

5. Добавьте следующие строчки скрипта сразу за окончанием предыдущей части скрипта:

```
styleStatus = new TextFormat()
with(styleStatus){
  font = myFonts[random(myFonts.length)];
  color = 0x858273;
  size = 16;
  bold = true;
}
```

6. Измените определение функции (function) `updateStatus()`, как показано ниже:

```
function updateStatus(){
  statusField._x = movingField._x;
  statusField._y       =       (movingField._y       +
movingField._height) + 10;
  statusField.text = movingField.length;
  statusField.setTextFormat(styleStatus);
}
```

7. На первый кадр, как только закончится предыдущий, добавьте следующий скрипт:

```
style1 = new TextFormat()
with(style1){
  align = "left";
  bold = false;
  color = 0x1A1917;
  font = myFonts[random(myFonts.length)];
  indent = 0;
  italic = true;
  leading = 0;
  leftMargin = 20;
  rightMargin = 20;
  size = 20;
  target = null;
  underline = false;
  url = null;
}
```

8. На первый кадр, как только закончится предыдущий, добавьте следующий скрипт:

```
style2 = new TextFormat()
with(style2){
  align = "center";
  bold = false;
```

```
  color = 0xCC0000;
  font = myFonts[random(myFonts.length)];
  indent = 0;
  italic = false;
  leading = 15;
  leftMargin = 0;
  rightMargin = 0;
  size = 14;
  target = null;
  underline = false;
  url = null;
}
```

9. Переместите вызовы функции (function call) `updateStatus()` и `reshapeBox()` из строчек, находящихся над строкой в которой написано `myFonts = TextField.getFontList();`, в конец скрипта, находящегося на первом кадре.
10. При раскрытой панели Actions выберите Кнопку 1 (Button 1) и добавьте нижеприведенный скрипт:

```
on(release){
  if (_root.applyToAll.getValue()){
    movingField.setTextFormat(style1);
    movingField.setNewTextFormat(style1);
  }else{
    movingField.setNewTextFormat(style1);
  }
  updateStatus();
  reshapeBox();
}
```

11. Для избежания избыточности поместите аналогичный код на Кнопки 2 , 3 и 4 (Buttons 2, 3, 4), но измените строчки, содержащие значения `style1` на `style2` , `style3` , and `style4` , соответственно.
12. Выберите Control > Text Movie для того, чтобы протестировать проект.
13. Закройте тестовое окно и сохраните проект под именем flashWriter3.fla.

Динамический контроль за Муви Клипами (Movie Clip)

Использование ATTACHMOVIE()

1. Откройте файл scrollingList1.fla из папки Lesson14/Assets.

2. Откройте библиотеку (library) и найдите movie clip, названый score list info bar. Щелкните пкм и выберите из выпадающего контекстного меню Linkage (Связывание). Выберите Export for ActionScript (Экспорт для ActionScript) в диалоговом окне Linkage Properties (Свойства связывания) после чего введите infoBar в тестовое поле для идентификатора в библиотеке (Identifier field).
3. Выберите первый кадр в слое (layer) Actions, находящийся на основной линейке времени (сцене). Откройте панель Actions и введите данный ActionScript:

```
list =
["Adrastea","Amalthea","Ananke","Callisto","Carme","E
lara","Europa","Ganymede",
"Himalia","Io","Leda","Lysithea","Metis","Pasiphae","
Sinope","Thebe"];
```

4. Пока первый кадр еще выбран, добавьте следующий код:

```
function buildList () {
  spacing = 30;
}
```

5. Для того, чтобы поставить приаттаченные (перетянутые из библиотеки) МС'ы в требуемые координаты и задать соответствующий текст, добавьте следующий ActionScript для определение функции (function):

```
var i = -1;
while (++i < list.length) {
  name = "infoBar" + i;
  y = i * spacing;
  display.list.attachMovie("infoBar", name, i);
  display.list[name]._y = y;
  display.list[name].moonName.text = list[i];
  display.list[name].moonNum.text = i + 1;
}
```

6. После выполнения функции (function), добавьте, `buildList();` в конце предыдущих инструкций на первый кадр (ниже определения функции (function)).
7. Выберите в меню Control > Test Movie для тестирования того, что вы только что создали.
8. Закройте тестовое окно и сохраните проект под именем scrollingList2.fla.

Создание кнопок с непрерывной обратной связью (CONTINUOUS-FEEDBACK BUTTONS)

1. Откройте файл scrollingList2.fla из папки Lesson14/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте следующий скрипт сразу за строчками, создающими list arrow:

```
startingY = display.list._y;
```

3. Для того, чтобы задать часть рабочей области, которая будет прокручиваться, добавьте `bottom=120;` в первый кадр сразу за строчкой скрипта, добавленного на предыдущем шаге создания проекта.
4. Начните определять функцию (function), которая будет использоваться для прокрутки (scroll movie clip'a). Для этого добавьте следующий ActionScript для определения функции (function) в первый кадр сразу за the `buildList()`:

```
function scroll (direction) {
  speed = 10;
}
```

5. Добавьте условие `if/else` внутри определения функции (function), сразу за `speed = 10`:

```
if (direction == "up") {
} else if (direction == "down") {
}
```

6. В верхней ("up" leg) части условия (`if`) необходимо ввести:

```
if (display.list._y - speed + display.list._height >
(startingY + bottom)) {
  display.list._y -= speed;
} else {
  display.list._y    =    (startingY    +    bottom)    -
display.list._height;
}
```

7. Теперь в нижней части условия оператора ("down" portion) `if/else` введите новое условие `if/else if`:

```
if (display.list._y + speed < startingY) {
  display.list._y += speed;
} else {
```

```
  display.list._y = startingY;
}
```

8. При открытой библиотеке (library) щелкните дважды лкм по movie clip'y с instance name list arrow, находящемуся в окне библиотеки (library).
9. При раскрытой панели Actions, выберите кнопку (button) и добавьте нижеприведенный скрипт:

```
on (press) {
  buttonPressed = "yes";
}
on (release, releaseOutside, dragOut) {
  buttonPressed = "";
}
```

10. При открытой библиотеке (library), щелкните дважды лкм по movie clip'y с именем scroll, который также находится в окне библиотеки (library).
11. При раскрытой панели Actions, выберите movie clip с instance name down и добавьте нижеприведенный скрипт:

```
onClipEvent (enterFrame) {
  if (buttonPressed == "yes") {
    _root.scroll("down");
  }
}
```

12. Не закрывая и не сворачивая панель Actions, выберите movie clip с instance name up и введите следующее:

```
onClipEvent (enterFrame) {
  if (buttonPressed == "yes") {
    _root.scroll("up");
  }
}
```

13. Выберите в меню Control > Test Movie для тестирования вашего проекта.
14. Закройте тестовое окно и сохраните проект под именем scrollingList3.fla.

Использование методов рисования (DRAW METHODS)

1. Откройте файл draw1.fla in the Lesson14/Assets folder.

2. При раскрытой панели Actions выберите movie clip с instance name controller, находящийся на основной линейке времени (сцене), и добавьте следующий скрипт:

```
onClipEvent (mouseUp) {
 down = false;
}
```

3. Добавьте это событие клипа (clip event) `mouseDown` в конце предыдущих инструкций:

```
onClipEvent (mouseDown) {
 if
(_root.canvas.hitTest(_root._xMouse,_root._yMouse)) {
   x = _root._xMouse;
   y = _root._yMouse;
   _root.holder.moveTo(x,y);
   down = true;
 }
}
```

4. Добавьте это Событие загрузки (`onload event`) в конце предыдущих инструкций:

```
onClipEvent (load) {
 currentColor = "7F6696";
   _root.createEmptyMovieClip("holder",100);
}
```

5. Начните описывать (определять) функцию (function) `draw()` при помощи добавления следующего скрипта в событие загрузки (`load event`):

```
function draw() {
  _root.holder.lineStyle(0, parseInt(currentColor,16),
100);
 x = _root._xmouse;
 y = _root._ymouse;
 _root.holder.lineTo(x, y);
}
```

6. Добавьте событие на клип (clip event) `mouseMove`, как только закончится предыдущий скрипт:

```
onClipEvent (mouseMove) {
 updateAfterEvent();
```

```
 if   (down   &&   _root.canvas.hitTest(_root._xmouse,
_root._ymouse)) {
   draw();
  }
}
```

7. Выберите в меню Control > Test Movie. Начните рисовать.
8. Закройте тестовое окно и сохраните проект под именем draw2.fla.

Z-сортировка имен экземпляров (INSTANCE NAMES) instance name MOVIE CLIP'ов

1. Откройте файл draw2.fla из папки Lesson14/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name window1 и добавьте нижеприведенный скрипт:

```
onClipEvent (load) {
 this.gotoAndStop("color");
}
```

3. Не закрывая и не сворачивая панель Actions, выберите movie clip с instance name window2 и добавьте следующий скрипт:

```
onClipEvent (load) {
 this.gotoAndStop("admin");
}
```

4. Щелкните дважды лкм window1 и перейдите в кадр с меткой (label) color. Добавьте следующий скрипт на кнопку (button) – первую кнопку слева (над первым карандашом слева):

```
on (release) {
 _parent.controller.currentColor = "7F6696";
}
```

5. Выберите оставшиеся кнопки – button (слева направо) и добавьте им следующие скрипты:

Добавьте на вторую кнопку (button):

```
on (release) {
    _parent.controller.currentColor = "FAC81C";
}
```

Добавьте на третью кнопку (button):

```
on (release) {
    _parent.controller.currentColor = "E72638";
}
```

Добавьте на четвертую кнопку (button):

```
on (release) {
    _parent.controller.currentColor = "1091CB";
}
```

Добавьте на пятую кнопку (button):

```
on (release) {
    _parent.controller.currentColor = "1FA900";
}
```

Добавьте на шестую кнопку (button):

```
on (release) {
    _parent.controller.currentColor = "BC0077";
}
```

6. Не закрывая и не сворачивая панель Actions, выберите оранжевую полосу за которую будет осуществляться перетаскивание окошка (dragbar) и которая, в свою очередь, является кнопкой (button), находящейся наверху окошка, и добавьте следующий скрипт:

```
on (press) {
  startDrag (this);
  _root.controller.swap(_name);
}
```

7. Теперь добавьте следующий скрипт на кнопку (button):

```
on (release) {
  stopDrag ();
}
```

8. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions выберите movie clip с instance name controller и добавьте следующий скрипт в событие загрузки `(load event)`:

```
topDog = "window1";
```

9. Добавьте следующий скрипт определения функции (function) в событие загрузки `(load event)`:

```
function swap (name) {
  _root[name].swapDepths(_root[topDog]);
  topDog = name;
}
```

10. Выберите в меню Control > Test Movie для тестирования проекта. Попробуйте перетаскивать окошко. Измените цвета линий.
11. Закройте тестовое окно и сохраните проект под именем draw3.fla.

Использование технологии DRAGG AND DROPP для MOVIE CLIP'ов

1. Откройте файл draw3.fla из папки Lesson14/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name controller и добавьте это определение функции (function) в событие загрузки (load event):

```
function buildIconList () {
  spacing = 85;
  iconY = 360;
  iconX = 70;
  var i = -1;
  while (++i < _root.icon._totalFrames) {
    newName = "icon" + i;
    _root.icon.duplicateMovieClip(newName, 10000 + i);
    _root[newName]._x = iconX + i * spacing;
    _root[newName]._y = iconY;
    _root[newName].gotoAndStop(i + 1);
  }
}
```

3. Добавьте `buildIconList();` в нижней части события загрузки (load event).
4. Щелкните дважды лкм по movie clip'у с instance name icon для того, чтобы отредактировать его и просмотреть его содержимое. При раскрытой панели Actions выберите невидимую кнопку (button) и добавьте нижеприведенный скрипт:

```
on (press) {
startDrag (this);
}
```

5. Не закрывая и не сворачивая панель Actions, добавьте следующий скрипт на кнопку (button):

```
on (release) {
```

```
stopDrag ();
  _root.controller.iconReleased(_name);
}
```

6. Вернитесь назад на основную линейку времени (main timeline, сцена) и выберите movie clip с instance name icon. При раскрытой панели Actions добавьте следующий скрипт

```
onClipEvent (load) {
    homeX = _x;
    homeY = _y;
}
```

7. Выберите clip controller и добавьте эту функцию (function) в событие загрузки (load event):

```
function iconReleased (name) {
  if
(_root.canvas.hitTest(_root._xmouse,_root._ymouse)) {
   ++v;
   newName="object" + v;
   _root[name].duplicateMovieClip(newName, v);

_root[newName].gotoAndStop(_root[name]._currentFrame)
;
    _root[newName].iconButton.enabled = false;
   root[newName]._xscale = 250;
   _root[newName]._yscale = 250;
  }
  _root[name]._x = _root[name].homeX;
  _root[name]._y = _root[name].homeY;
}
```

8. Выберите в меню Control > Test Movie для тестирования на работоспособность вашей работы. Перетащите иконку.
9. Закройте тестовое окно и сохраните проект под именем draw4.fla.

Удаление динамически созданного содержимого

1. Откройте файл draw4.fla из папки Lesson14/Assets.
2. При раскрытой панели Actions выберите controller movie clip instance name и добавьте это определение функции (function) в событие загрузки (load event):

```
function clearContent () {
  _root.holder.clear()
```

```
  i = 0;
  while (++i <= v) {
    name = "object" + i;
    _root[name].removeMovieClip();
  }
  v = 0;
}
```

3. Дважды щелкните лкм по movie clip'у с instance name window2 для того, чтобы отредактировать его и просмотреть его содержимое. Перейдите в кадр с меткой (label) admin.
4. При раскрытой панели Actions выберите кнопку (button) Print и добавьте нижеприведенный скрипт:

```
on (release) {
  printAsBitmap ("_root", "bmovie");
}
```

5. При раскрытой панели Actions выберите кнопку (button) Clear и введите данный ActionScript:

```
on (release) {
  _parent.controller.clearContent();
}
```

6. Выберите в меню Control > Test Movie для тестирования проекта. Нарисуйте несколько линий и затем нажмите кнопку Clear.
7. Закройте проект и сохраните ваш проект под именем draw5.fla.

Динамизм проекта, основанный на времени и кадрах

Определение текущей даты и времени

1. Откройте файл makeMyDay1.fla из папки Lesson15/Assets.
2. При открытом инспекторе свойств (Property inspector) выберите верхнее текстовое поле (text field), currentDay. Нажмите на кнопке (button) в инспекторе свойств (Property inspector) задания типов символов для вызова диалогового окна Character Options.
3. При раскрытой панели Actions выберите первый кадр на слое (layer) Actions и добавьте следующий скрипт:

```
stop();
today = new Date();
```

4. Добавьте следующие две строчки скрипта, как только закончится предыдущий скрипт:

```
nameOfDays = ["Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday",
"Friday", "Saturday"];
nameOfMonths = ["January", "February", "March",
"April", "May", "June",
"July", "August", "September", "October", "November",
"December"];
```

5. Добавьте следующую строчку скрипта в конце предыдущего участка кода:

```
currentDay.text = nameOfDays[today.getDay()];
```

6. Добавьте следующую строчку скрипта в конце предыдущего участка кода:

```
currentMonth.text = nameOfMonths[today.getMonth()];
```

7. Добавьте следующую строчку скрипта в конце предыдущего участка кода:

```
currentDate.text = today.getDate();
```

8. Добавьте следующую строчку скрипта в конце предыдущего участка кода:

```
currentYear.text = today.getFullYear();
```

9. Выберите в меню Control > Test Movie для того, чтобы просмотреть проект на данном этапе его создания.
10. Закройте тестовое окно и сохраните проект под именем makeMyDay2.fla.

Определение времени перехода

1. Откройте файл makeMyDay2.fla из папки Lesson15/Assets.
2. Щелкните дважды лкм по movie clip’у с instance name clock для того, чтобы отредактировать его и просмотреть его содержимое.
3. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions выберите кнопку (button) Start и добавьте следующий скрипт:

```
on (release) {
  clock.alarmOn = true;
  clock.startingTime = getTimer();
}
```

4. Не закрывая и не сворачивая панель Actions, выберите movie clip с instance name clock и добавьте нижеприведенный скрипт:

```
onClipEvent (enterFrame) {
  if (alarmOn) {
}
```

5. Добавьте следующие строчки скрипта внутрь условного оператора:

```
totalTimeToAlarm = (Number(_root.minutesToAlarm.text)
* 60) +
Number(_root.secondsToAlarm.text);
```

6. Добавьте следующую строчку скрипта сразу за строчкой, добавленной в предыдущем шаге:

```
secondsElapsed = Math.round((getTimer () -
startingTime) / 1000);
```

7. Добавьте следующую строчку скрипта сразу за строчкой, добавленной в предыдущем шаге:

```
alarmHand._rotation = totalTimeToAlarm / 10;
```

8. Добавьте следующую строчку скрипта сразу за строчкой, добавленной в последнем шаге:

```
secondHand._rotation = secondsElapsed * 6;
minuteHand._rotation = secondsElapsed / 10;
```

9. Добавьте следующий скрипт в виде условного оператора сразу за действиями, добавленных в последнем шаге:

```
if (secondsElapsed == totalTimeToAlarm) {
  activateAlarm();
}
```

10. Добавьте следующий скрипт сразу за окончанием предыдущей части скрипта:

```
onClipEvent (load) {
  function activateAlarm () {
    alarmOn = false;
    secondHand._rotation = 0;
    minuteHand._rotation = 0;
    alarmHand._rotation = 0;
    alarmSound = new Sound();
```

```
    alarmSound.attachSound("annoying");
    alarmSound.start( 0, _root.numberOfLoops.text );
  }
}
```

11. Выберите в меню Control > Test Movie для тестирования данного проекта на текущем этапе его создания.
12. Закройте тестовое окно и сохраните ваш проект под именем makeMyDay3.fla.

Контроль за скоростью проигрывания и переходы по временной шкале (TIMELINE)

1. Откройте файл makeMyDay3.fla из папки Lesson15/Assets.
2. При раскрытой панели Actions выберите кнопку (button) "Get phone messages" и добавьте нижеприведенный скрипт:

```
on (release) {
  nextFrame();
}
```

3. При открытой панели Сцена (Scene panel) выберите сцену (scene) Messages из списка сцен, чтобы сделать ее видимой и доступной для доработки в среде Flash.
4. Щелкните дважды лкм по movie clip'у с instance name messages для того, чтобы отредактировать его и просмотреть его содержимое.
5. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions выберите movie clip с instance name messages и добавьте следующий скрипт:

```
onClipEvent (enterFrame) {
  if (action == "ff") {
    gotoAndStop (_currentframe + 3);
  } else if (action == "rew") {
    gotoAndStop (_currentframe - 10);
  } else if (action == "play") {
    play ();
  } else if (action == "stop") {
    stop();
  }
}
```

6. При раскрытой панели Actions выберите кнопку (button) Play и добавьте следующий скрипт:

```
on (release) {
```

```
 messages.action = "play";
}
```

7. При раскрытой панели Actions выберите кнопку (button) Stop и добавьте следующий скрипт:

```
on (release) {
 messages.action = "stop";
}
```

8. При раскрытой панели Actions выберите кнопку (button) FF и добавьте нижеприведенный скрипт:

```
on (press) {
 messages.action = "ff";
 fastSound = new Sound();
 fastSound.attachSound("rewind");
 fastSound.start(0, 50);
}
on (release) {
 messages.action = "play";
 fastSound.stop();
}
```

9. При раскрытой панели Actions выберите кнопку (button) Rew и добавьте нижеприведенный скрипт:

```
on (press) {
 messages.action = "rew";
 fastSound = new Sound();
 fastSound.attachSound("rewind");
 fastSound.start(0, 50);
}
on (release) {
 messages.action = "play";
 fastSound.stop();
}
```

10. Выберите в меню Control > Test Movie для того, чтобы протестировать проект на данном этапе создания.
11. Закройте тестовое окно и сохраните ваш проект под именем makeMyDay4.fla.

Отслеживание проигрывания (TRACKING) и прогресса во время загрузки (DOWNLOADING PROGRESSION)

1. Откройте файл preloader1.fla из папки Lesson15/Assets.
2. Щелкните дважды лкм по movie clip'у с instance name preloader для того, чтобы отредактировать его и просмотреть его содержимое.
3. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions выберите movie clip с instance name preloader и добавьте нижеприведенный скрипт:

```
onClipEvent (enterFrame) {
 framesLoaded = Math.ceil ((_parent._framesloaded /
_parent._totalframes)
  * 100);
 gotoAndStop (framesLoaded);
 info.text = framesLoaded + "% completed";
 if (framesLoaded >= 90) {
   _root.gotoAndPlay (2);
 }
}
```

4. Выберите в меню Control > Test Movie для того, чтобы посмотреть проект на данном этапе. Затем проверьте функциональные возможности проекта, для этого выберите Debug >56K, затем выберите View > Show Streaming.
5. Закройте тестовое окно и сохраните ваш проект под именем preloader2.fla.

Программирование звука (Sound)

Создание объекта звук (SOUND OBJECT)

1. Откройте файл basketball1.fla из папки Lesson16/Assets.
2. Дважды щелкните лкм по movie clip'у с instance name basketball для того, чтобы открыть его и просмотреть его содержимое.
3. Выберите Edit > Edit Document для возврата на основную временную линейку (сцену).
4. Выберите movie clip с instance name basketball, откройте панель Actions, затем добавьте следующий скрипт

```
onClipEvent (load) {
 bounce = new Sound (this);
}
```

5. Выберите в меню Control > Test Movie для того, чтобы посмотреть, как будет работать проект.
6. Закройте окно предварительного просмотра и вернитесь в среду создания приложений. Сохраните текущий файл под именем basketball2.fla.

Перетаскивание MOVIE CLIP'а внутри заданных границ

1. Откройте файл basketball2.fla из папки Lesson16/Assets.
2. Выберите movie clip с instance name basketball, затем откройте панель Actions. После строчек скрипта, создающих Sound object для громкого удара (который был создан в предыдущем разделе), добавьте следующие строчки скрипта:

```
leftBoundary = 60;
rightBoundary = 490;
topBoundary = 220;
bottomBoundary = 360;
```

3. После седьмой строчки скрипта находится строчка с фигурными скобками (}), за ней добавьте следующий скрипт:

```
onClipEvent (mouseMove) {
  if (_root._xmouse > leftBoundary && _root._xmouse <
rightBoundary &&
_root._ymouse  >  topBoundary  &&  _root._ymouse  <
bottomBoundary) {
    startDrag (this, true);
  } else {
    stopDrag ();
  }
}
```

4. Выберите в меню Control > Test Movie для того, чтобы просмотреть как работают операторы.
5. Закройте окно предварительного просмотра и вернитесь в среду создания приложений. Сохраните текущий файл под именем basketball3.fla.

Задание громкости звука (VOLUME)

1. Откройте файл basketball3.fla из папки Lesson16/Assets.
2. Выберите movie clip с instance name basketball, затем откройте панель Actions. После строчек скрипта задающих стороны границ для движения movie clip'а (которые были созданы в предыдущей части) добавьте следующие строчки скрипта:

```
boundaryHeight = bottomBoundary - topBoundary;
```

3. Поместите следующие строчки скрипта после обработчика событий `startDrag (this, true)`:

```
topToBottomPercent = ((((_root._ymouse - topBoundary)
/ boundaryHeight) *
100) / 2) + 50;
```

4. Поместите следующие строчки скрипта после строчки, которая была только что добавлена (в которой была создана переменная `topToBottom`):

```
bounce.setVolume(topToBottomPercent);
```

5. Поместите следующие строчки скрипта после строчки, которая была только что добавлена (в которой задавалась громкость звукового объекта (Sound object) `bounce`):

```
this._xscale = topToBottomPercent;
this._yscale = topToBottomPercent;
```

6. Выберите в меню Control > Test Movie.
7. Закройте окно предварительного просмотра и вернитесь в среду создания приложений. Сохраните текущий файл под именем basketball4.fla.

Контролирование прокрутки звука

1. Откройте файл basketball4.fla из папки Lesson16/Assets.
2. Выберите movie clip с instance name basketball, затем откройте панель Actions. После строчки кода, в которой записано `boundaryHeight = bottomBoundary - topBoundary` , добавьте следующую строчку кода:

```
boundaryWidth = rightBoundary - leftBoundary;
```

3. После строчки кода, в которой записано `boundaryWidth = rightBoundary - leftBoundary`, добавьте следующую строчку скрипта:

```
quadrantSize = boundaryWidth / 2;
```

4. После строчки кода, в которой записано `quadrantSize = boundaryWidth / 2` , добавьте следующий скрипт:

```
centerPoint = rightBoundary - quadrantSize;
```

5. После строчки кода, в которой записано `quadrantSize = boundaryWidth / 2` , добавьте следующие строчки скрипта:

```
panAmount = ((_root._xmouse - centerPoint) /
quadrantSize) * 100;
bounce.setPan (panAmount);
```

6. Выберите в меню Control > Test Movie для того, чтобы просмотреть работоспособность проекта.
7. Закройте окно предварительного просмотра, вернитесь в среду создания приложений. Сохраните текущий файл под именем basketball5.fla.

Присоединение звука из библиотеки (ATTACHING) и контроль звука при проигрывании

1. Откройте файл basketball5.fla из папки Lesson16/Assets.
2. Выберите Window > Library для того, чтобы открыть библиотеку (Library panel).
3. Щелкните лкм по Sound 0, тем самым выбирая его. Из меню Library Option, выберите Linkage (Связывание).
4. Выберите Export for ActionScript (Экспорт для ActionScript) из опций Linkage и дайте этому звуку идентификатор (identifier name) Sound0.
5. Повторите шаги 3 и 4 для Sound 1 и Sound 2 из той же библиотеки (library).
6. Выберите movie clip с instance name basketball, затем откройте панель Actions. После строчки скрипта в которой создавался звуковой объект (Sound object) для громкого удара - вы работали с ним чуть ранее, добавьте следующую строчку скрипта:

```
dynaSounds = new Sound();
```

7. Добавьте следующие строчки скрипта, как только закончится предыдущий скрипт:

```
onClipEvent (mouseDown) {
 randomSound = random (3);
 randomLoop = random (2) + 1;
 dynaSounds.attachSound ("Sound" + randomSound);
 dynaSounds.start (0, randomLoop);
}
```

8. Добавьте следующие строчки скрипта, как только закончится предыдущий скрипт:

```
onClipEvent (keyDown) {
 dynaSounds.stop();
}
```

9. Выберите в меню Control > Test Movie для просмотра действий в проекте.
10. Закройте окно предварительного просмотра, вернитесь в среду создания приложений. Сохраните текущий файл под именем basketball6.fla.

Загрузка внешних данных

Загрузка внешних файлов внутрь Movie Clip'ов (LOADING INTO A TARGET)

1. Откройте файл virtualaquarium1.fla из Lesson17/Assets folder.
2. Используя стандартный эксплорер для вашей операционной системы, посмотрите на содержимое директории (папки, folder, system's directory) под названием Lesson17/Assets.

В этой директории вы найдете файлы, которые вам предстоит загружать динамически. Директория содержит в частности файлы: background0.swf, background1.swf, background2.swf, banner0.swf, banner1.swf, banner2.swf.

3. Вернитесь во Flash. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт:

```
backgrounds = new Array ("background0.swf",
"background1.swf",
"background2.swf");
```

4. Поместите следующее определением функции (function) ниже скрипта, который вы только что добавили:

```
function randomBackground() {
 randomNumber = random (backgrounds.length);
 loadMovie (backgrounds[randomNumber],
"_root.background");
}
```

5. Теперь поместите следующий вызов функции (function call) сразу за определением функции (function) добавленноым в предыдущем шаге:

```
randomBackground();
```

6. Выберите в меню Control > Test Movie.
7. Закройте тестовое окно. Сохраните файл под именем virtualaquarium2.fla.

Динамическая загрузка JPG-ов

1. Откройте файл virtualaquarium2.fla из папки Lesson17/Assets.
2. Используя стандартный эксплорер для вашей операционной системы, посмотрите на содержимое директории (папки, folder, system's directory) под названием Lesson17/Assets.

   Найдите следующие графические файлы стандарта JPG:

   image0.jpg

   image1.jpg

   image2.jpg

3. Вернитесь во Flash. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт сразу за массивом `backgrounds`:

```
slides = new Array(["Shark", "image0.jpg"],
["Jellyfish", "image1.jpg"],
["Seahorse", "image2.jpg"]);
```

4. Поместите следующее определение функции (function) сразу за определением функции (function) `randomBackground()`:

```
function changeSlide(number){
 if (number >= 0 && number < slides.length){
  currentSlide = number;
  _root.title.text = slides[number][0];
  loadMovie (slides[number][1],
"_root.placeholder");
  }
}
```

5. Поместите следующий вызов функции (function call) сразу за вызовом функции (function call) `randomBackground()`:

```
changeSlide(0);
```

6. При раскрытой панели Actions, выберите кнопку влево и присоедините этот скрипт:

```
on(release){
 changeSlide(_root.currentSlide - 1);
}
```

7. При раскрытой панели Actions, выберите кнопку (button) на сцене (stage) в виде указателя вправо и присоедините этот скрипт:

```
on(release){
  changeSlide(_root.currentSlide + 1);
}
```

8. Выберите в меню Control > Test Movie.
9. Закройте тестовое окно и сохраните файл под именем virtualaquarium3.fla.

Создание интерактивного заполнителя (PLACEHOLDER)

1. Откройте файл virtualaquarium3.fla из папки Lesson17/Assets.
2. При раскрытой панели Actions выберите movie clip с instance name placeholder и добавьте следующий скрипт:

```
onClipEvent(load){
  thisX = _x;
  thisY = _y;
}
```

3. Пока еще movie clip с instance name placeholder еще выбран, добавьте следующий скрипт сразу за тем участком кода, который вы только что добавили:

```
onClipEvent(mouseDown) {
  if (hitTest(_root._xmouse, _root._ymouse)){
    _root.maskClip._x = thisX;
    _root.maskClip._y = thisY;
    setMask(_root.maskClip);
    _xscale = 150;
    _yscale = 150;
    startDrag(this);
  }
}
```

4. Пока еще movie clip с instance name placeholder еще выбран, добавьте следующий скрипт сразу за участком кода, который вы только что добавили:

```
onClipEvent (mouseUp) {
  stopDrag();
  setMask(null);
  _xscale = 100;
  _yscale = 100;
```

```
  _x = thisX;
  _y = thisY;
}
```

5. Выберите в меню Control > Test Movie.
6. Вернитесь в среду создания приложений. При раскрытой панели Actions выберите movie clip с instance name maskClip и присоедините этот скрипт:

```
onClipEvent (load){
  _visible = false;
}
```

7. Выберите в меню Control > Test Movie.
8. Закройте тестовое окно и сохраните файл под именем virtualaquarium4.fla.

Загрузка файлов (MOVIES) на уровень (LEVEL)

1. Откройте файл virtualaquarium4.fla из папки Lesson17/Assets.
2. Используя стандартный эксплорер для вашей операционной системы, посмотрите на содержимое директории (папки, folder, system's directory) под названием Lesson17/Assets.

Найдите следующие файлы - movies:

banner0.swf

banner1.swf

banner2.swf

3. Вернитесь во Flash. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующую строчку скрипта сразу за строчкой, создающей массив (array) `slides` в предыдущей части:

```
banners = new Array("banner0.swf", "banner1.swf",
"banner2.swf");
```

4. Поместите следующее определение функции (function) сразу за определением функции (function)`changeSlide()`:

```
function randomBanner() {
  randomNumber = random (banners.length);
  loadMovieNum (banners[randomNumber], 1);
```

```
}
```

5. Поместите следующий вызов функции (function call) сразу за вызовом функции (function call) `changeSlide()`:

```
randomBanner();
```

6. Поместите следующие строчки скрипта сразу за вызовом функции (function call) `randomBanner()`:

```
setInterval(randomBanner, 10000);
```

7. Выберите в меню Control > Test Movie.
8. Закройте тестовое окно, сохраните файл под именем virtualaquarium5.fla.

Контроль за файлом (MOVIE), загруженным на уровень (LEVEL)

1. Откройте файл virtualaquarium5.fla из папкиLesson17/Assets.
2. Выберите File > Open. Просмотрите папку Lesson17/Assets folder, найдите файл с именем banner0.fla и откройте файл во Flash.
3. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующий скрипт:

```
_x = _level0.bannerBack._x;
_y = _level0.bannerBack._y;
```

4. Добавьте этот аналогичный скрипт в первый кадр в другие файлы с баннерами (banner movies) с именами banner1.fla и banner2.fla. Затем переэкспортируйте их в файлы с именами banner0.swf, banner1.swf, и banner2.swf, соответственно.
5. Вернитесь в virtualaquarium5.fla. Выберите в меню Control > Test Movie.
6. Закройте тестовое окно и вернитесь в среду создания основного файла (virtualaquarium5.fla).
7. При раскрытой панели Actions выберите кнопку (button), содержащую текст 'Banner On/Off' и добавьте следующий скрипт:

```
on(release) {
_level1._visible = !_level1._visible;
bannerBack._visible = !bannerBack._visible;
}
```

8. Выберите в меню Control > Test Movie.

9. Закройте тестовое окно. Сохраните файл под именем virtualaquarium6.fla.

Динамическая загрузка звуковых файлов стандарта MP3

1. Откройте файл virtualaquarium6.fla из папки Lesson17/Assets.
2. Рассмотрите содержание директории Lesson17/Assets.

Найдите следующие MP3 файлы и отметьте их имена:

music0.mp3

music1.mp3

music2.mp3

3. Вернитесь во Flash. При раскрытой панели Actions выберите первый кадр на слое Actions и измените массив (array) `slides` array как показано ниже:

```
slides = new Array(["Shark", "image0.jpg",
"music0.mp3"], ["Jellyfish",
"image1.jpg", "music1.mp3"], ["Seahorse",
"image2.jpg", "music2.mp3"]);
```

4. Добавьте следующие строчки скрипта в конце предыдущего, но внутри условного оператора if, для определения функции (function) `changeSlide()`:

```
slideSound.stop();
slideSound = new Sound();
slideSound.loadSound(slides[number][2], true);
```

5. Выберите в меню Control > Test Movie.
6. Закройте тестовое окно. Сохраните файл под именем virtualaquarium7.fla.

Реакция на динамически загруженные звуковые файлы стандарта MP3

1. Откройте файл virtualaquarium7.fla из папки Lesson17/Assets.
2. При раскрытой панели Actions выберите первый кадр на слое Actions и добавьте следующие строчки скрипта в конце предыдущего, но внутри условного оператора if для определения функции (function) `changeSlide(`:

```
slideSound.onSoundComplete = function(){
    changeSlide(currentSlide + 1);
}
```

3. Щелкните дважды лкм по movie clip'у с instance name progress для того, чтобы отредактировать его и просмотреть его содержимое.
4. Вернитесь на основную временную линейку (сцену). При раскрытой панели Actions, выберите movie clip с instance name progress и добавьте следующий скрипт:

```
onClipEvent(enterFrame){
 progressamount                                    =
Math.round(((_root.slideSound.position /
_root.slideSound.duration) * 100));
 gotoAndStop(progressAmount);
 if (progressAmount == 100){
  _visible = false;
 }else{
  _visible = true;
 }
}
```

5. Выберите в меню Control > Test Movie.
6. Закройте тестовое окно. Сохраните файл под именем virtualaquarium8.fla.

Кафедра «Компьютерные образовательные технологии» (КОТ)

История кафедры КОТ неразрывно связана с развитием университета в последнее десятилетие, с превращением СПбГУИТМО в ведущий компьютерный вуз страны. Научную и научно-методическую основу создания и развития кафедры определи два события в жизни университета:

1999 год: создание центра дистанционного обучения (ЦДО) университета.

2000 год: открытие Санкт-Петербургского регионального центра «Федерации Интернет Образования» (СПб РЦ ФИО).

ЦДО СПбГУИТМО был создан как полигон для разработки дистанционных курсов университета. Сегодня ЦДО стал не только научно-методическим центром СПбГУИТМО, но и реальным участником учебного процесса вуза. Аттестации студентов, виртуальные лаборатории в среде системы ДО СПбГУИТМО стали неотъемлемой частью учебного процесса различных кафедр университета, разумеется, и кафедры КОТ.

СПб РЦ ФИО создан в университете в рамках негосударственного образовательного проекта «Поколение.ru» для массовой переподготовки работников среднего образования в области интернет-технологий. Успешная работа в этом направлении позволила обучить к сентябрю 2005 года более 6000 учителей. Методики и программы обучения интернет-технологиям на курсах СПб РЦ ФИО стали основой многих студенческих учебных программ.

2001 год: открытие на факультете «Информационных Технологий и Программирования» кафедры КОТ.

Вначале кафедра КОТ обеспечивала учебный процесс по специальности 230201 (старый индекс 071900) – «Информационные системы и технологии». Дисциплины кафедры: «Информатика», «Информационные технологии», «Web-программирование».

2003 год:

- открытие в СПбГУИТМО новой инженерной специальности 230202 (073700) – «Информационные технологии в образовании»;
- переход кафедры КОТ в разряд выпускающих кафедр;
- разработан и утвержден учебный план по специализации «Аппаратно-программные комплексы образовательных систем»;
- произведен первый набор студентов специальности 230202.

С этого времени развитие кафедры подчинено главной цели: качественной подготовке специалистов высшей квалификации в области информационных технологий для сферы образования.

Заведующая кафедрой – к.т.н., доцент, Лисицына Л.С.
Сайт кафедры – http://dce.ifmo.ru

Для заметок

Для заметок

Web-программирование. Клиентский ActionScript.
Николаев Дмитрий Геннадьевич, Штенников Дмитрий Геннадьевич
Учебное пособие
Редакционно-издательский отдел СПбГУИТМО
Лицензия ИД № 00408 от 05.11.99
Компьютерная верстка: Николаев Д.Г., Штенников Д.Г.
Дизайн обложки: Николаев Д.Г., Штенников Д.Г.
Художественное оформление: Николаев Д.Г., Штенников Д.Г.
Подписано к печати 30.08.05. Тираж 300 экз. Заказ №
Отпечатано на ризографе в Центре издательских систем СПбГУИТМО