Дмитрий Гурский,
Юрий Стрельченко

Обзор языка JSFL

Доп. материалы к статье

Одним из полезнейших нововведений Flash MX 2004 стал язык JSFL. Что это такое и зачем нужно? JSFL (JavaScript Flash Language) — это скриптовый язык среды разработки Flash MX 2004. Используя его, можно создавать собственные команды меню, инструменты, эффекты и прочие расширения. Любое действие, которое вы можете выполнить в среде разработки «вручную», может быть проделано посредством JSFL. Вопреки распространенному заблуждению, JSFL не является частью ActionScript. Более того, они связаны ничуть не в большей степени, чем, например, ActionScript и JavaScript.

JSFL является скриптовым языком программы Macromedia Flash MX 2004, а ActionScript — это скриптовый язык Macromedia Flash Player. JSFL, как и ActionScript и JavaScript, основан на стандарте ECMA-262. Поэтому вам его будет освоить чрезвычайно просто. У JSFL такой же, как у этих языков синтаксис, объектная модель, ему присущи такие неспецифичные классы, как Array, String, Math.

Особенно близок JSFL к JavaScript (отсюда и его название). Дело в том, что для описания среды разработки Flash была избрана та же объектная модель документа (DOM), которую использовали в свое время программисты Netscape для браузера. Поэтому, если вы владеете JavaScript, многое в JSFL вам покажется знакомым.

Код JSFL сохраняется, как и код ActionScript, в обычных текстовых файлах, имеющих расширение *.JSFL. Для обозначения таких файлов есть специальный значок, показанный на рис. 1.

Рис. 1. Значок JSFL-файла

Создать JSFL-файл можно в любом текстовом редакторе. Однако если у вас установлена профессиональная версия Flash, то использовать их нет смысла. Дело в том, что в ней имеется особый режим редактирования кода JSFL. Войти в него можно, выбрав в меню Create New появляющейся при открытии программы панели пункт Flash JavaScript File. Данный режим более удобен, чем внешний текстовый редактор, так как в нем все элементы JSFL отображены в дереве языка в левой части панели Actions, а также работает подсветка кода и выводятся всплывающие подсказки.

Язык JSFL по своей структуре является своего рода зеркальным отражением среды разработки Flash. Такие привычные понятия, как документ, временная шкала, инструмент, библиотека существуют в нем в форме объектов. Например, инструмент панели Tools олицетворяет объект класса ToolObj. Документ принадлежит окну программы, библиотека — это часть документа, клип — элемент библиотеки. Иерархия элементов, присущая среде разработки, сохраняется и в JSFL. Обратиться к элементу уровня N можно только через содержащий его элемент уровня N-1. На рисунке 2 показана иерархическая структура объектов JSFL. Читая в данном обзоре о некотором классе или объекте, возвращайтесь к этой схеме, чтобы лучше понять, какое место он занимает в языке. В результате, вы сможете сложить для себя целостное представление о структуре JSFL, что позволит с легкостью освоить язык, пользуясь одним лишь словарем от Macromedia.

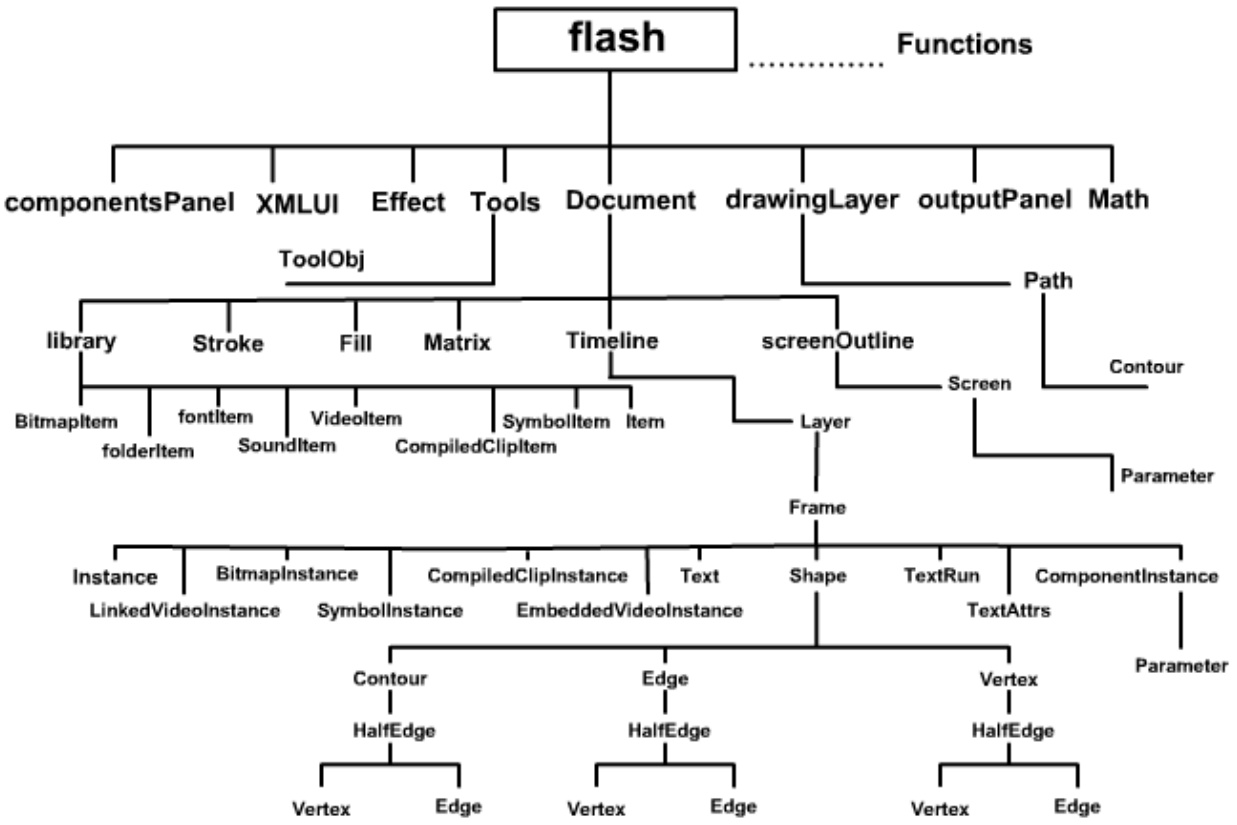

Рис.2. Иерархия объектов JSFL

На первом уровне в иерархии объектов JSFL (помимо глобальных функций) располагается только один объект — flash (к нему также можно обратиться по сокращенному имени «fl»). Объекту flash соответствует окно программы. Так как все элементы среды разработки принадлежат окну Flash, дочерними по отношению к объекту flash являются все остальные специфичные объекты JSFL.

Собственные методы объекта flash в основном решают те же задачи, что и команды меню File среды разработки. Используя их, можно создать новый документ или закрыть существующий, сохранить документ или закрыть окно программы. Свойства же объекта flash предназначены главным образом для доступа к объектам JSFL второго уровня. Например, описывающие открытые документы объекты класса Document хранятся в массиве flash.documents.

Для примера приведем код, создающий новый документ, затем сохраняющий его на диске C как Example.FLA, после чего закрывающий окно Flash:

```
flash.createDocument();
flash.saveDocument(flash.documents[0], "file:///C|/Example.FLA");
flash.quit(false);
```

Чтобы задействовать данный код, сохраните его в JSFL-файле. Затем просто выполните по значку файла двойной щелчок мышью. Запустится Flash, и все указанные действия будут проделаны. Выполнить код JSFL можно и командой из среды разработки, для чего служит пункт Run Command меню Commands.

Ко второму уровню в иерархии объектов JSFL относится около 10 видов объектов, важнейшими среди которых являются объект Tools и объекты класса Document. Объект Tools описывает панель инструментов среды разработки. Объектам класса Document соответствуют открытые в среде разработки документы.

Объекты Document — это, пожалуй, самые важные элементы JSFL. Огромное количество принадлежащих им дочерних элементов, методов, свойств позволяют управлять всеми присущими FLA-файлу компонентами. 90% JSFL — это элементы, подчиненные объектам класса Document. Только дочерних объектов первого уровня у объектов Document 7. Объектов же более

низкой иерархии десятки, причем степень их вложенности может доходить до 8 (см. схему выше)!

Объекты класса Document, описывающие открытые FLA-документы, хранятся в массиве documents объекта flash. Порядок их расположения определяется последовательностью, в которой они были открыты. На практике, как правило, бывает необходимо работать с документом, находящимся в фокусе. Получить ссылку на соответствующий ему объект класса Document позволяет метод getDocumentDOM() объекта flash.

У класса Document имеется просто колоссальное количество свойств и методов. Используя их, можно создать фильм любой сложности только посредством одного лишь кода JSFL! Так как описать все свойства и методы класса Document в рамках данного обзора невозможно, приведем лишь несколько примеров.

Следующий код меняет размер рабочего поля документа, а также частоту кадров:

```
var doc=flash.getDocumentDOM();
doc.width=200, doc.height=200;
doc.frameRate=24;
```

Чтобы запустить режим тестирования, следует набрать:

```
flash.getDocumentDOM().testMovie();
```

Команда рисует круг:

```
flash.getDocumentDOM().addNewOval({left:100, top: 100, right:200, bottom: 200});
```

Провести операцию перевода выделенного текста в кривые и заливку можно следующими двумя строчками (дважды повторяем команду Break Apart):

```
flash.getDocumentDOM().breakApart();  // Разбиваем текст на отдельные буквы
flash.getDocumentDOM().breakApart();  // Преобразуем буквы в графику
```

Чтобы выделить все элементы на рабочем поле, наберите:

```
flash.getDocumentDOM().selectAll();
```

Чтобы создать экземпляр хранящегося в библиотеке клипа и поместить его в точку с координатами (100, 100), следует использовать приблизительно такой код:

```
var item=flash.getDocumentDOM().library.items[0];
flash.getDocumentDOM().addItem({x:100, y:100}, item);
```

Следующая строчка аналогична команде File/Save среды разработки:

```
flash.getDocumentDOM().save();
```

Произвести импорт SWF-файла в ту же директорию, где сохранен FLA-документ, можно, набрав:

```
flash.getDocumentDOM().exportSWF("", true);
```

Строчка выполняет те же действия, что и команда Publish меню File:

```
flash.getDocumentDOM().publish();
```

Не все команды, связанные с настройками FLA-документа, принадлежат классу Document. Некоторые из них для большей стройности языка были отнесены к классам, описывающим отдельные элементы FLA-документа: библиотеку, временную шкалу, опции линии и заливки. Объекты этих классов являются дочерними по отношению к объекту Document, и получить к ним доступ можно или через специальные свойства, или посредством особых методов типа get. Например, чтобы обратиться к объекту класса Timeline, которому соответствует временная шкала, нужно использовать функцию getTimeline() класса Document:

```
var timeline=flash.getDocumentDom().getTimeline();
```

Операции над временной шкалой посредством JSFL можно проводить те же, что и «вручную». К примеру, чтобы сделать десятый кадр ключевым, следует набрать:

```
flash.getDocumentDOM().getTimeline().insertKeyframe(10);
```

Создать направляющий слой ниже выделенного слоя можно следующей строчкой:

```
flash.getDocumentDOM().getTimeline().addNewLayer("Напр_слой","guide",false);
```

Продолжаем спускаться по лестнице вложенности объектов ветви Timeline. Слоям временной диаграммы соответствует отдельный класс Layer. Каждый слой описывает индивидуальный объект этого класса. Хранятся эти объекты в массиве layers объекта Timeline. Чем выше располагается слой, тем меньший индекс будет у управляющего им объекта Layer. Пример:

```
flash.getDocumentDOM().getTimeline().addNewLayer("Напр_слой","guide",false);
flash.trace(flash.getDocumentDOM().getTimeline().layers[1].name);
// Выводит: Напр_слой
```

Классу Layer присущ ряд свойств, задающих характеристики слоя. Например, за то, будут ли отображаться объекты слоя, отвечает свойство visible:

```
// Делаем слой невидимым
flash.getDocumentDOM().getTimeline().layers[0].visible=false;
```

Логично предположить, что кадры временной диаграммы также должны описываться объектами отдельного класса. Так оно и есть. Всеми многочисленными опциями кадров можно управлять посредством свойств класса Frame. Хранятся описывающие кадры слоя объекты класса Frame в массиве frames соответствующего объекта класса Layer. Индекс объекта в массиве будет на единицу меньшим, чем номер управляемого им кадра. Например, чтобы применить к размещенному на первом кадре первого слоя звуку эффект затухания, следует набрать:

```
flash.getDocumentDOM().getTimeline().layers[0].frames[0].soundEffect = 'fade in';
```

Наиболее интересным для практики свойством класса Frame является actionScript, которое позволяет поместить на ключевой кадр код ActionScript. Для примера приведем скрипт JSFL, который создает новый документ, помещает на первый кадр первого слоя команду ActionScript, сохраняет документ, после чего закрывает его:

```
flash.createDocument();
flash.getDocumentDOM().getTimeline().layers[0].frames[0].actionScript="trace('При
вет');";
flash.saveDocument(flash.getDocumentDOM(), "file:///C|/Trace.FLA");
flash.getDocumentDOM().close(false);
```

На кадре могут располагаться экземпляры клипов и кнопок, растровые изображения, графика, текстовые поля. В JSFL каждому такому элементу будет соответствовать объект, причем элементы разных типов будут описывать объекты разных классов. Эти классы объединяет то, что они все являются подклассами особого класса Element. Всего таких классов 8 (текстовые поля относятся к классу Text, графика к классу Shape, экземпляры клипов и кнопок к классу Instance, и т. д.). Хранятся объекты, управляющие элементами кадра, в массиве elements объекта класса Frame. Их очередность зависит от последовательности отображения элементов. Чем выше располагается элемент, тем больший у описывающего его объекта будет индекс в массиве elements.

Классы группы Element позволяют проводить основные операции с элементами кадра. Например, чтобы увеличить вдвое все текстовые поля, принадлежащие первому кадру первого слоя документа, задействуйте следующий код:

```
var frame =flash.getDocumentDOM().getTimeline().layers[0].frames[0]
var frame_el=frame.elements;
for(var i=0; i<frame_el.length; i++){
  if(frame_el[i].elementType=="text"){
    frame_el[i].width*=2, frame_el[i].height*=2;
  }
}
```

Объекты некоторых классов группы Element имеют дочерние объекты. Таким образом, объекты отдельных классов могут иметь степень вложенности 9 по сравнению с объектом flash (то есть чтобы до них «добраться», требуется «пройти» 8 объектов). Поэтому не стоит удивляться, увидев в коде JSFL строчку, подобную следующей:

```
flash.getDocumentDOM().getTimeline().layers[0].frames[0].elements[0].edges[0].get
HalfEdge(0).getVertex();
```

С объектами классов группы Element можно также встретиться, если вы работаете с выделением. Дело в том, что у объекта Document имеется массив selection, хранящий ссылки на объекты, соответствующие которым элементы в данный момент являются выделенными. Пример:

```
// Увеличиваем все выделенные объекты вдвое
var sel_objs=flash.getDocumentDOM().selection;
for(var i=0; i<sel_objs.length; i++ ){
  sel_objs[i].width*=2, sel_objs[i].height*=2;
}
```

Ветвь языка, связанную с объектом Timeline, мы просмотрели практически полностью. Сделаем несколько шагов назад и вернемся на третий уровень вложенности. Помимо объекта класса Timeline, у объекта Document имеется еще несколько дочерних объектов. Наиболее важным из них является объект класса Library, который управляет библиотекой документа. На него указывает свойство library класса Document.

В общем, класс Library предоставляет доступ к тем же операциям с библиотекой, что и одноименная панель. Например, чтобы создать экземпляр символа Clip и поместить его в точку (100, 100), нужно задействовать такую команду:

```
flash.getDocumentDOM().library.addItemToDocument({x:100, y:100}, "Clip");
```

Следующая команда вызовет режим редактирования символа Clip:

```
flash.getDocumentDOM().library.editItem( "Clip");
```

Подобно тому, как элементы, расположенные на кадре, управляются объектами классов группы Element, так и элементам в библиотеке соответствуют объекты ряда классов (класс SymbolItem — символы, класс FontItem — шрифты, BitmapItem — растровые изображения и так далее). Надклассом всех этих классов является класс Item. Хранятся объекты, управляющие элементами библиотеки, в массиве items объекта класса Library. Их порядок будет тот же, в котором отображаются значки элементов на панели Library. Пример:

```
flash.trace(flash.getDocumentDOM().library.items[0].name);  // Выводит: Clip
```

Класс Item имеет ряд довольно значимых для практики возможностей. К примеру, следующими строчками можно присвоить клипу идентификатор программного экспорта «clip»:

```
flash.getDocumentDOM().library.items[0].linkageExportForAS=true;
flash.getDocumentDOM().library.items[0].linkageIdentifier="clip";
```

Большинство подклассов класса Item «пустые». Но отдельные подклассы содержат полезные методы и свойства. Так, класс SoundItem дает возможность настроить параметры включаемого в фильм звука. Например, чтобы перевести все звуки в библиотеке в формат MP3 с битрейтом 16 Кбит/с, необходимо выполнить следующий JSFL-код:

```
var library_items=flash.getDocumentDOM().library.items;
for(var i=0; i<library_items.length; i++){
  if(library_items[i].itemType=="sound"){
    library_items[i].compressionType="MP3";
    library_items[i].bitRate="16 kbps";
  }
}
```

В общих чертах мы изучили основные элементы главной ветви дерева JSFL, связанной с объектами класса Document. Однако помимо объектов Document, ко второму уровню иерархии языка относится еще 6 типов объектов. Кратко охарактеризуем их:

- Tools. Объект, соответствующий панели инструментов среды разработки. Хранится в свойстве tools объекта flash. Его свойства и методы предназначены для проведения наиболее общих операций, связанных с инструментами. Например, чтобы получить ссылку на активный в данный момент инструмент, следует набрать:

  ```
  var tool_active=flash.tools.activeTool;
  ```

Для управления инструментами панели Tools был введен специальный класс ToolObj. Для каждого инструмента создается отдельный объект этого класса. Хранятся такие объекты в массиве toolObjs объекта Tools.

Класс ToolObj применяется, главным образом, для создания собственных инструментов. Но при помощи него можно влиять и на стандартные инструменты.

- componentsPanel. Объект управляет панелью Components.
- outputPanel. Объекту соответствует панель Output.
- Math. Объект, хранящий дополнительные математические методы (в основном, для работы с матрицами), которых нет в глобальном объекте Math.
- Effect. Класс, предназначенный для создания эффектов временной диаграммы.
- drawingLayer. Объект, описывающий гипотетический слой, на котором располагается фигура в момент ее создания при помощи любого инструмента рисования. После того как кнопка мыши отпускается, фигура переносится с гипотетического слоя на настоящий.
- XMLUI. Зачастую при создании команд бывает необходимо, чтобы пользователь мог передать коду некоторые параметры. В JSFL эта задача решается выведением специального диалогового окна (похожее окно появляется, например, когда вы активируете команду Modify ▸ Bitmap ▸ Trace Bitmap). То, как должно выглядеть это окно и какие переменные будут из него определяться, задается посредством особого XML-файла, отчего управляющий данным окном класс и имеет столь необычное название — XMLUI (сокращение от XML и User Interface).

Глобальной видимостью, помимо объекта flash, обладают все стандартные функции, классы и объекты ECMA-262: escape(), Array, String, Math и т. д. Кроме того, к первому уровню иерархии объектов JSFL относится несколько специфичных для среды разработки Flash функций. Данные функции активируются при наступлении ряда событий, откуда и их названия: mouseDown(), mouseClick(), keyUp() и т. п. Глобальных функций JSFL немного и используются они, главным образом, при создании инструментов для панели Tool. Более обстоятельно мы с ними познакомимся в проекте к этому материалу.

JSFL — не такой сложный и объемный язык, как ActionScript. Облегчает его изучение и то, что практически всем его объектам соответствуют хорошо знакомые элементы среды разработки. Поэтому с нашей стороны не будет излишне смелым попытаться реализовать какое-нибудь полезное расширение прямо сейчас. Мы попробуем написать на JSFL команду, создающую на основании выделенного текста анимацию, в которой по тексту будет бежать волна:

1. Создайте новый JSFL-файл и назовите его WaveText.
2. Перед тем как выполнить код команды, следует убедиться, что в документе имеется только один выделенный элемент, причем этот элемент — текстовое поле. Ссылки на соответствующие выделенным элементам объекты классов группы Element хранит массив selection. Значит, нам нужно проверить длину данного массива и тип хранящегося в нем элемента. Если условия, необходимые для выполнения командой своих функций, окажутся не соблюденными, следует отобразить информационную панель с сообщением об ошибке:

```
var doc=flash.getDocumentDOM();
if(doc.selection.length!=1 && doc.selection[0].elementType!="text"){
  alert ("Выделите одно текстовое поле и повторите команду");
} else {
```

3. Как мы бы создавали анимацию волнообразного текста «вручную»? Очевидно, что поместили бы текстовое поле в клип, затем разбили его на отдельные буквы посредством команды Break Apart. Потом мы бы так изменили размеры каждой буквы, чтобы получилась волна. Затем мы бы добавили еще один ключевой кадр — и поменяли на нем размеры букв так, чтобы образуемая ими волна слегка сместилась по сравнению с первым кадром. Схожим образом для создания более-менее живой анимации нам понадобилось бы заполнить 15–20 кадров. Даже если букв в строке немного, эта работа заняла бы не меньше часа (а что если их десятки?).

   Самое приятное в работе с JSFL то, что для достижения желаемого эффекта алгоритм может проделывать точно такие же действия, которые мы бы выполняли «вручную». Итак, вначале нам следует поместить выделенное поле в новый клип. Для этого нужно использовать метод класса Document convertToSymbol(type, name, centerLock), где type —

тип символа (возможные значения: «movie clip», «button», «graphic»), name — имя символа в библиотеке, centerLock — расположение точки центра («top left», «center», «bottom right», и т. д.). Имя символа должно быть уникальным, но, в то же время, читабельным. Решаем эту проблему следующим образом: запускаем цикл и формируем имя по принципу «sinText»+i, где i — целое число. Как только обнаружится, что символа с таким именем в библиотеке нет (определить это позволяет метод itemExists() класса Library), создаем необходимый клип.

```
var name="sinText", n=0;
while(doc.library.itemExists(name+n)) {
  n++;
}
doc.convertToSymbol("movie clip", name+n, "center")
```

4. Далее мы должны перейти в режим редактирования нового клипа:

```
doc.enterEditMode();
```

5. Теперь мы должны разбить текст на отдельные буквы, применив к нему команду Break Apart:

```
doc.breakApart();
```

6. Далее при помощи цикла создаем последовательность из 16 ключевых кадров. Необходимые буквы будут скопированы на них автоматически:

```
var timln=doc.getTimeline();
for(var k=1, k<16; k++){
    timln.insertKeyframe(i+1);
}
```

7. Теперь мы должны последовательно перебрать все 16 кадров, меняя размеры букв так, чтобы они образовали бегущую волну. Выделить некоторый кадр или несколько кадров на временной диаграмме позволяет метод класса Timeline setSelectedFrames(begin, end), где begin — номер кадра, начинающего подлежащую выделению последовательность, а end — номер кадра, перед которым она завершается. Отсчет кадров ведется с 0.

   После того как кадр будет выделен, размер букв должен быть изменен по закону синуса. Сделать это довольно несложно, так как объекты массива selection, соответствующие буквам, будут организованы в той же последовательности, что и сами буквы (а никаких других элементов в клипе нет). Скорость движения волны произвольна и зависит от разности фаз колебаний соседних букв. У нас разность фаз для двух соседних букв будет составлять 15°. Разность фаз колебаний последней буквы кадра N и первой буквы кадра N+1 также должна равняться 15°. Важное условие: разность фаз должна быть подобрана так, чтобы при прокручивании анимации в цикле создавалось впечатление непрерывного движения (то есть разность фаз между последней буквой последнего кадра и первой буквой первого кадра должна быть такой же, как разность фаз соседних букв).

```
var phi=Math.PI/8;  // Разность фаз
var mat=null;       // Переменная для матрицы преобразований
for(var i=0; i<15; i++){            // Перебираем все кадры
  timln.setSelectedFrames(i, i+1);  // Выделяем кадр i
  for(var j=0; j<doc.selection.length; j++){  // Перебираем буквы
    // Считываем матрицу преобразований данной буквы
    mat=doc.selection[j].matrix;
    // Модифицируем элементы матрицы, отвечающие за масштабирование
    mat.a=mat.d=Math.abs(Math.sin(phi*(i+j)));
    // Заменяем старую матрицу на модифицированную
    doc.selection[j].matrix=mat;
  }
}
```

   В JSFL для осуществления таких преобразований, как поворот, перенос, относительное масштабирование используются матрицы преобразований. Им соответствуют объекты особого класса Matrix, которые хранятся в свойстве matrix объектов классов группы Element. В приведенном выше коде мы воспользовались матрицей преобразований, переопределив элемент (0, 0) (свойство a) и (1,1) (свойство d), отвечающие за коэффициенты масштабирования по горизонтали и по вертикали.

8. После того как буквы будут организованы нужным образом, необходимо выйти из режима редактирования клипа:

```
doc.exitEditMode();
```

Готово. Сохранив JSFL-файл, откройте новый FLA-документ. Создайте статичное текстовое поле с любым текстом. Выделив его, вызовите команду sinText через окно, открываемое пунктом Run Command меню Commands. Если все было сделано верно, то окно Flash станет на несколько секунд недоступным. Затем вы увидите, что текст приобрел форму волны, и был помещен в клип. Откройте этот клип. Как и задумывалось, на его временной диаграмме имеется 16 ключевых кадров. Войдите в режим тестирования. Ура, все работает!

Наша команда не лишена недостатков. Нельзя произвольно задавать частоту колебаний, их амплитуду. Если бы этот пример не был учебным, нужно было при запуске команды выводить специальное диалоговое окно, в котором пользователь мог бы задавать параметры. Управляет этим окном особый объект JSFL второго уровня XMLUI. То, сколько в окне будет полей, как они будут называться и какие переменные с ними будут связаны, описывается в специальном XML-документе, хранящимся в той же папке, что и JSFL-файл.

Если вы создали или скачали из Сети особо полезную команду JSFL, ее стоит сделать легкодоступной. Для этого поместите ее в папку Commands папки Configuration (ее адрес может быть таким: %userprofile%\Local Settings\Application Data\Macromedia\Flash MX 2004\en\Configuration). В результате, после перезапуска программы соответствующий этой команде пункт будет добавлен в меню Commands. Изменить название или удалить JSFL-команду из меню Commands можно при помощи панели, вызываемой командой Manage Saved Commands того же меню. Загрузить дополнительные бесплатные расширения от Macromedia можно со страницы, открываемой командой Get More Commands меню Commands.

Аналогично командам встраиваются и инструменты. Для этого описывающие их файлы следует поместить в папку Tools директории Configuration.

Установка расширений, созданных как с использованием JSFL, так и без него — это операция не сложная, но требующая определенных познаний. Чтобы упростить и формализовать инсталляцию расширений, компанией Macromedia была создана небольшая бесплатная утилита Macromedia Extension Manager. У этой утилиты имеется собственный формат файлов *.MXP. Данные файлы обычно создаются простым слиянием файла расширения и описывающего его XML-документа. Macromedia Extension Manager позволяет как создавать MXP-файлы, так и инсталлировать их в систему (при этом она просто переносит их в папку Extensions папки Configuration). Считается хорошим тоном «упаковывать» расширения в MXP-формат, поэтому обязательно загрузите Extension Manager с сайта Macromedia. Тем более, подавляющее большинство расширений (команд, инструментов, компонентов), выложенных в Сети, имеет формат *.MXP.

Команды и инструменты — это самые простые элементы, которые можно создать посредством JSFL. Более сложными являются эффекты вроде стандартных эффектов временной диаграммы (Insert ▸ Timeline Effects). Для их реализации нужно, помимо кода JSFL, разрабатывать SWF-фильм с интерфейсом. И это не оговорка. Это на первый взгляд кажется удивительным, но SWF-фильм может являться частью среды разработки. Еще во Flash MX можно было создавать для компонентов собственные Инспекторы Свойств. Во Flash MX 2004 даже заставка, появляющаяся при открытии программы, есть ничто иное, как SWF-фильм, взаимодействующий с JSFL. Возникает вопрос: как ActionScript отправляет данные JSFL. А предназначена для этого особая глобальная функция MMExecute(). В качестве параметра она принимает строку с кодом JSFL. Например:

```
MMExecute("alert('Привет');"); // Команда выводит информационную панель
```

Не пытайтесь задействовать функцию MMExecute() из режима тестирования среды разработки или тем более из внешнего плеера. Она выполнит свои функции лишь в том случае, если фильм является элементом интерфейса Flash.

Кстати, функция MMExecute() очень напоминает getURL(). И та, и другая служат для того, чтобы отправлять команды скриптовому языку содержащего фильм приложения.