MerlinTwi 27-09-2006, 11:02

В ActionScript 3, можно динамически менять частоту кадров (frame rate) ролика используя класс Stage.
Класс Stage (flash.display.Stage (http://livedocs.macromedia.com/flex/2/langref/flash/display/Stage.html)) связан со сценой главного ролика и имеет свойство frameRate, которое может принимать значения от 0.01 до 1000, и определяет частоту кадров, с которой Flash плеер проигрывает ролик. Изменять это значение можно на лету.

```
// изменить частоту кадров на 12 fps:
stage.frameRate = 12;
```

MerlinTwi 27-09-2006, 11:22

ActionScript 3 полностью базируется на классах. Создавая классы, вы создаете переменные и функции (методы) которые связаны с экземпляром класса. В отличие от ActionScript 2 методы в ActionScript 3 сохраняют область видимости их класса, даже если вызываются из другого объекта, или через Function.call и Function.apply.
Например:

```
package {
import flash.display.Sprite;

public class ClassScope extends Sprite {

public function ClassScope() {
traceThis(); // "Class Instance"

var obj:Object = new Object();
obj.traceThis = traceThis;
obj.traceThis(); // "Class Instance"

traceThis.call(new Sprite()); // "Class Instance"
}

public override function toString():String {
return "Class Instance";
}

public function traceThis():void {
trace(this);
}
}
}
```

MerlinTwi 27-09-2006, 11:42

Как и в ActionScript 1,2 в ActionScript 3 есть методы для динамического рисования в отображаемых объектах (movie clips, sprites, и т.п.), которые имеют свойство graphics (flash.display.Graphics (http://livedocs.macromedia.com/flex/2/langref/flash/display/Graphics.html)). Свойство graphics выступает в роли специального слоя для рисования, который расположен под всеми дочерним клипами. Так же в ActionScript 3 добавлены новые методы для рисования прямоугольников (в том числе и со скругленными углами), окружностей и эллипсов:

```
drawCircle(x:Number, y:Number, radius:Number):void
drawEllipse(x:Number, y:Number, width:Number, height:Number):void
drawRect(x:Number, y:Number, width:Number, height:Number):void
drawRoundRect(x:Number, y:Number, width:Number, height:Number, ellipseWidth:Number,
ellipseHeight:Number):void
```

Пример:

```
// Нарисовать синий прямоугольник со скругленными углами:
var square:Sprite = new Sprite();
square.graphics.beginFill(0xFF);
square.graphics.drawRoundRect(0, 0, 100, 50, 10, 10);
square.graphics.endFill();
addChild(square);
```

MerlinTwi 27-09-2006, 21:02

В ActionScript 2 переменные со значениями объявленные в теле класса сохранялись в прототипе класса. Из-за этого возникали проблемы с переменными, которые являются ссылками на объекты (например массивы Array):

```
class myClass{
public var list:Array = [1,2,3];
}
var a = new myClass();
var b = new myClass();
trace(a.list === b.list); // true
```

Здесь a.list и b.list фактически ссылаются на один и тот же массив.
В ActionScript 3 этой проблемы нет, такие переменные будут уникальны.

MerlinTwi 27-09-2006, 21:30

В ActionScript 2 событие MouseMove срабатывало глобально, неважно над каким клипом находилась мышка, любой listener от Mouse или любой мувиклип получали событие о перемещении мышки.
В ActionScript 3 для того чтобы получать события от мышки, нужно у интерактивного клипа (InteractiveObject такие как Sprites, MovieClips, Stage) добавить слушателя событий мышки (Listener), но события будут приходить только тогда, когда мышка находится над этим клипом. Для того чтобы получать события мышки в любом месте ролика, нужно добавить Listener к Stage.
stage.addEventListener(MouseEvent.MOUSE_MOVE, onMouseMove);

Огион 28-09-2006, 11:25

10. Ключевое слово delete и члены класса (http://www.delphimaster.ru/articles/as3tips.html#garbagecollector)

MerlinTwi 28-09-2006, 12:03

В ActionScript 3 появился новый класс SimpleButton (flash.display.SimpleButton (http://livedocs.macromedia.com/flex/2/langref/flash/display/SimpleButton.html)). Теперь можно динамически с помощью ActionScript создавать кнопки (Button).
var myButton:SimpleButton = new SimpleButton();
Класс SimpleButton имеет 4 основных свойства, которые управляют возможными состояниями кнопки: upstate (кнопка не нажата), overState (мышка над кнопкой), downstate (кнопка нажата), и hitAreaState (область реагирования на мышку). Необходимо создать визуальные объекты для этих состояний и присвоить их к соответствующим свойствам.
myButton.upState = mySprite1;
myButton.overState = mySprite2;
myButton.downState = mySprite3;
myButton.hitAreaState = mySprite4;

Огион 28-09-2006, 12:11

11. Класс Dictionary (http://www.delphimaster.ru/articles/as3tips.html#dictionaryclass)
12. Метки (http://www.delphimaster.ru/articles/as3tips.html#labelstatements)

MerlinTwi 28-09-2006, 12:20

Один класс описывается в одном .as файле, так было в ActionScript 2 так остается и в ActionScript 3, но теперь есть возможность включать вспомогательные классы в тот же .as файл в дополнение к основному классу. Вспомогательные классы описываются после блока package и видны только основному классу (или функции) в блоке package, или другим вспомогательным классам, описанным в этом же файле.

```
package {
class MyClass {
function MyClass() {
var helper:MyHelper = new MyHelper();
}
}
}
class MyHelper {
function MyHelper() {
var helper:HelpersHelper = new HelpersHelper();
}
}
class HelpersHelper {
function HelpersHelper () {
}
}
```

Запомните, что только один класс может быть описан в блоке package. Вспомогательные классы описываются в этом же файле, но после package и могут использоваться только основным классом.

MerlinTwi 28-09-2006, 13:38

В ActionScript 3 добавлено два новых ключевых слова для описания прав доступа и усовершенствовано свойство private по сравнению с ActionScript 2. Итак, теперь мы имеем следующие ключевые слова для регулирования прав доступа:

public
protected
private
internal (по умолчанию)

public: то же самое, что и в ActionScript 2, все определенное как public может быть прочитано откуда угодно. Конструкторы теперь всегда public.

protected: новое ключевое слово, указывает, что метод или переменная скрыта от всех кроме классов потомков. Классы и конструкторы не могут быть определены как protected.

private: в ActionScript 2 было это ключевое свойство, но фактически работало как protected, т.е. классы потомки имели полный доступ к методу или свойству, описанному как private. Теперь в ActionScript 3 private это полноценная приватность, т.е. доступ только в этом классе, в котором метод или переменная описана, для всех остальных, включая потомков, она не будет существовать. Это означает, что в классе-потомке можно объявить еще одну переменную или метод с таким же именем и при этом не возникнет никаких конфликтов.
Классы и конструкторы не могут быть определены как private.

internal: аналогично public, но ограничено пределами пакета (package). Только классы, описанные в этом же пакете, будут иметь доступ к internal переменным или методам.
Internal ставится по умолчанию для любого класса, переменным или методам класса, кроме конструкторов, которые всегда public.

Права доступа во вспомогательных классах немного отличаются. Поскольку они фактически описываются вне пакета (package), то переменные или методы, объявленные как internal будут доступны только классам, описанным в этом же файле.

В ActionScript 3 проверка прав доступа осуществляется не только на этапе компиляции, но и при выполнении, т.е. хаки использовавшиеся в ActionScript 2, чтобы получить доступ к скрытым методам больше не будут работать.

Рассмотрим все это на примерах.
Пример 1.

```
package {

import flash.display.Sprite;

// Класс AccessControl объявлен как public (по умолчанию был бы internal)
public class AccessControl extends Sprite {

// Конструктор всегда public
function AccessControl() {

// Только классы описанные в этом файле
// имеют доступ к вспомогательным классам
var helper:Helper = new Helper();

trace(helper.pubNum); // OK - это public
// trace(helper.protNum); // Error – нет доступа к protected
// trace(helper.privNum); // Error – нет доступа к private
trace(helper.interNum); // OK - это internal
}
}
}

// Класс Helper по умолчанию internal
class Helper {

// public – переменная видна отовсюду
public var pubNum:Number = 1;

// protected – доступ только из классов потомков
protected var protNum:Number = 2;

// private – доступно только в этом классе
private var privNum:Number = 3;

// internal – доступно только в этом же package,
// но для вспомогательного класса это означает,
// что только для классов описанных в этом файле
internal var interNum:Number = 4;

// Конструктор всегда public
function Helper() {
}
}

// класс SubHelper будет internal
// Потомок вспомогательного класса Helper
class SubHelper extends Helper {

// Конструктор всегда public
```

```
function SubHelper() {
trace(pubNum); // OK – это public
trace(protNum); // OK – т.к. мы являемся потомком
// trace(privNum); // Error – нет доступа к private
trace(interNum); // OK – в этом же файле
}
}
```

Пример 2.

```
package {

import flash.display.Sprite;
import containers.*; // описано ниже

// Класс AccessControl объявлен как public (по умолчанию был бы internal)
public class AccessControl extends Sprite {

// Конструктор всегда public
function AccessControl() {

// Есть доступ из другого пакета (packages)
// только если класс public
var bowl:Bowl = new Bowl(); // OK
// var basket:Basket = new Basket(); // Error – нет доступа к internal

trace(bowl.pubNum); // OK
// trace(bowl.protNum); // Error - нет доступа к protected
// trace(bowl.privNum); // Error - нет доступа к private
// trace(bowl.interNum); // Error - нет доступа к internal
}
}
}
package containers {

// Класс public доступен везде
public class Bowl {

// Переменная public доступна везде
public var pubNum:Number = 1;

// protected – доступ только у наших потомков
protected var protNum:Number = 2;

// private – доступ только в этом классе
private var privNum:Number = 3;

// internal – доступ только в этом пакете (package)
internal var interNum:Number = 4;

// Конструктор всегда public
function Bowl() {

// Есть доступ к inteneral т.к. в этом же пакете
var basket:Basket = new Basket();

trace(basket.pubNum); // OK
// trace(basket.protNum); // Error – нет доступа к protected
// trace(basket.privNum); // Error - нет доступа к private
trace(basket.interNum); // OK – в этом же пакете

// clone объявлен как public
var basketCopy:Basket = basket.clone();
}
}
}
package containers {

// interal – доступ только в этом пакете (package)
internal class Basket {

// public – доступ отовсюду
public var pubNum:Number = 1;

// protected – доступ только у потомков
protected var protNum:Number = 2;

// private – доступ только в этом классе
```

```
private var privNum:Number = 3;

// internal – доступ только в этом пакете (package)
internal var interNum:Number = 4;

// Конструктор всегда public
function Basket() {
}

// public – доступ отовсюду
public function clone():Basket {
var basket:Basket = new Basket();
basket.pubNum = pubNum; // OK
basket.protNum = protNum; // OK – этот же класс
basket.privNum = privNum; // OK - этот же класс
basket.interNum = interNum; // OK
return basket;
}
}
}
```

Огион 29-09-2006, 11:14

19. Абстрактные классы (http://www.delphimaster.ru/articles/as3tips.html#abstractclasses)
20. Ключевое слово override (http://www.delphimaster.ru/articles/as3tips.html#override)

Временно не буду иметь возможность продолжить, минимум до понедельника.

Nirth 30-09-2006, 21:38

В ActionScript 3.0, подход к управлению глубинами DibsplayObject'ов изменился. Разберем пример, когда мы хотим поставить мувики с наибольшей "y" координатой наверх, а наименьшей в низ, в AS 1-2, мы могли просто назначить им глубину, в ActionScript 3, мы работаем с массивом дочерних объектов, а значит никаких пустых глубин между двумя мувиками быть не может.

Один из способов сделать это, это отсортировать наши мувики в массиве. Сначала мы создадим массив с сылками на на мувики, после чего отсортируем его с помощью свойства sortOn.

```
var sortedItems:Array = new Array(mc1, mc2, mc3);
function arrange():void {
sortedItems.sortOn("y", Array.NUMERIC);
var i:int = sortedItems.length;
while(i--){
if (getChildAt(i) != sortedItems[i]) {
setChildIndex(sortedItems[i], i);
}
}
}
```

Данный способ не столь краток как AS 1-2 вариант, но делает свое дело.

Nirth: я писал статью-обзор о коллекциях (http://orangeflash.eu/?p=75) в AS3 и Flex Framework, с некоторыми другими коллекциями можно сотворить более качественную глубинную сортировку.

Nirth 30-09-2006, 22:09

С помощью класса ByteArray, мы можем копировать полностью объекты. Под полным копированием имеется ввиду, что мы можем копировать так же "дочерние" объекты копируемого объекта, и так далее. Например если у нас есть массив с объектами, и мы копируем его, то кроме самого массива скопируются и объекты.

Пример функции:

```
import flash.utils.ByteArray;

function clone(source:Object):* {
var copier:ByteArray = new ByteArray();
copier.writeObject(source);
copier.position = 0;
return(copier.readObject());
}
```

Использование:

newObjectCopy = clone(originalObject);

Nirth: Информация о классе не сохраняется, но мы можем воспользоватся кастингом:

```
package {
import flash.display.*;
import flash.utils.*;

dynamic public class Test extends Sprite
{
public function Test()
{
var user:UserInfo = new UserInfo();
user.name = "David";
user.email = "david@david.com";
user.page = new HomePage();
user.page.title = "David";
user.page.url = "david.com";

trace("user class name:",getQualifiedClassName(user));
trace("user.page class name:",getQualifiedClassName(user.page));

var user2:UserInfo = clone(user) as UserInfo;

trace("user2 class name:",getQualifiedClassName(user));
trace("user2.page class name:",getQualifiedClassName(user.page));

}

private function clone(source:Object):* {
var copier:ByteArray = new ByteArray();
copier.writeObject(source);
copier.position = 0;
return(copier.readObject());
}
}
}

class UserInfo
{
internal var name:String;
internal var email:String;
internal var page:HomePage;

public function UserInfo()
{
}
}
class HomePage
{
internal var url:String;
internal var title:String;

public function HomePage()
{
}
}
```

Данный метод хорошо подходит для чистых DTO/VO (http://ru.wikipedia.org/wiki/DTO) объектов.

Aziz Zaynutdinoff 30-09-2006, 22:37

Прототип объекта в ActionScript — это объект, который существует среди классов, чьи значения доступны для всех экземпляров класса, к которым он (объект) принадлежит. В ActionScript 1.0/2.0 prototype использовался для контроля наследования класса. Когда экземпляр подкласса обращается к переменной, сначала проверяется наличиие этой переменной в текущем экземпляре класса, потом (если не нашел) в прототипе данного класса, потом в прототипе базового класса, и так далее по цепочке наследственности (прототипов), до тех пор пока есть базовые классы.

В ActionScript 3.0 наследование в основном управляется через Класс Наследований и не зависит от прототипа объекта. Однако, прототип объекта всё же существует и обеспечивает большую часть функциональности, как это было и в AS1.0/2.0

Каждый класс и внутриклассовая функция (метод), созданные в ActionScript 3.0 имеют прототипный объект связанный с ними. Для классов прототип имеет имеет аттрибут «read-only», что значит, что вы не можете изменять его значение новым. Однако, это вовсе не значит, что вы не можете задать новую переменную и присвоить ей

значение внутри прототипа (иначе наличие прототипа было б бессмысленным ;)). Прототипы функций изменяемы, что позволяет создавать динамические классы, используя определение классов написанием функций и определением наследований через прототипы («old style» как это делалось раньше).

Пример:

ActionScript Code:

```
package {

import flash.display.Sprite;

public dynamic class MyClass extends Sprite {

public function MyClass(){

// prototype = new Object(); // ОШИБОЧНО, нельзя изменить прототип класса
prototype.newValue = 1; // OK, добавляем (или удаляем) переменные в прототип

trace(this.newValue); // 1
trace(prototype.toString); // function Function() {}
trace(prototype.addChild); // undefined
trace(addChild); // function Function() {}

// динамическое определение класса («old style»)
var TempClass:Function = function():void {
trace("Создаём TempClass");
}

TempClass.prototype = prototype; // OK, можно утсанавливать наследствования

var tempObject:* = new TempClass(); // «Создаём TempClass»

trace(tempObject.newValue); // 1
}
}
}
```

Помните: обязательно ставить префикс this, обращаясь к динамичным перемнным. Также следует учитывать, что методы класса Object динамичны и определены прототипом (поэтому не могу перекрываться при помощи override). также следует отметить, что tempObject имеет тип «*» вместо TempClass. Потому что TempClass распознаётся всего лишь как функция в AS3.0, не класс, хотя может использоваться и как класс. Все динамические классы (созданные как TempClass) всегда будут распознаваться как функции (хотя и существует тип Class), поэтому созданные таким путём классы всегда порождают собой экземпляры простых объектов.

Nirth 30-09-2006, 23:30

В ActionScript 1, мы могли дать свойству экземпляра и свойству класса одинаковое имя, в ActionScript 2 ввели ограничение на это. Теперь в ActionScript 3 мы опять можем это делать.

```
package {

import flash.display.Sprite;

public class MyApp extends Sprite {

private var variable:String;
private static var variable:String;

public function MyApp() {
this.variable = "foo";
MyApp.variable = "bar";

trace(this.variable); // "foo"
trace(MyApp.variable); // "bar"
trace(variable); // "foo"
}
}
}
```

Nirth 30-09-2006, 23:52

В ActionScript 3 используется класс flash.events.EventDispatcher для работы с событиями (рассылка, подписка обработчиков). Одна из реализаций данного класса была доступна в ActionScript 2, но это был внешний класс, для работы с событиями в mx.* пакетах. Теперьэто встроенный класс, а значит он работает быстрее.

Кажды раз когда вы хотите обработать событие в ActionScript 3 (например "enterFrame" или "click") вы должны использовать методы EventDispatcher.
Теперь у классов нету методов вроде onEnterFrame, onPress, onLoad.

У EventDispatcher'а есть следующие методы:

addEventListener(type:String, listener:Function, useCapture:Boolean = false, priority:int = 0,
useWeakReference:Boolean = false):void
Метод подписывает обработчик события.
type - Тип(имя) события
listener - ссылка на функцию - обработчика
useCapture - используется ли capture фаза или bubbles фаза.
priority - приоритет вызова обработчика
useWeakReference - использовать слабую проверку типов.

dispatchEvent(event:Event):Boolean
Метод рассылает события.
event - Экземпляр класса Event или подкласса, который будет разослан.

hasEventListener(type:String):Boolean
Проверяет есть ли обработчик события у события указанного в параметре type

removeEventListener(type:String, listener:Function, useCapture:Boolean = false):void
Удаляет обработчик событий, параметры аналогичны addEventListener

willTrigger(type:String):Boolean

Запомните, теперь в качестве слушателя выступают функции а не объекты как это было в случае с addListener.

Теперь о хорошем:
Нам больше не нужно использовать mx.utils.Delegate или подобные ему, теперь this указывает на то что надо.

У класса Event есть свойство target, которое ссылается на экземпляр который разослал событие, при использовании capture или bubble фаз, так же можно испоьзовать свойство currentTarget.

Простой пример

```
package {

import flash.display.Sprite;
import flash.events.Event;

public class MyDispatcher extends Sprite {

public function MyDispatcher() {
addEventListener("customEvent", handleEvent);
dispatchEvent(new Event("customEvent"));
}

private function handleEvent(event:Event):void {
trace(event.type); // "customEvent"
}
}
}
```

Путем расширения класса EventDispatcher или любого другого класса, который его расширяет (все DisplayObject'ы его расширяют), ваш класс получает возможность разсылать и принимать события, тоесть они могут использовать dispatchEvent и addEventListener.

Но иногда нам нужно наследовать классы, которые не расширяют EventDispatcher, в этом случае мы можем реализовать интерфейс IEventDispatcher, и создать в классе экземпляр EventDispatcher. Тоесть мы используем композицию вместо наследования.

Пример:

```
package {

import flash.display.Sprite;
import flash.events.Event;

public class MyDispatcher extends Sprite {

public function MyDispatcher() {
var dispatcher:CustomDispatcher = new CustomDispatcher();
dispatcher.addEventListener("customEvent", handleEvent);
dispatcher.dispatchEvent(new Event("customEvent"));
}
```

```
private function handleEvent(event:Event):void {
trace(event.type); // "customEvent"
}
}
}

import flash.events.Event;
import flash.events.EventDispatcher;
import flash.events.IEventDispatcher;

class CustomDispatcher implements IEventDispatcher {

private var eventDispatcher:EventDispatcher;

public function CustomDispatcher() {
eventDispatcher = new EventDispatcher(this);
}

public function addEventListener(type:String, listener:Function, useCapture:Boolean = false, priority:int = 0,
useWeakReference:Boolean = false):void {
eventDispatcher.addEventListener.apply(null, arguments);
}
public function dispatchEvent(event:Event):Boolean {
return eventDispatcher.dispatchEvent.apply(null, arguments);
}

public function hasEventListener(type:String):Boolean {
return eventDispatcher.hasEventListener.apply(null, arguments);
}

public function removeEventListener(type:String, listener:Function, useCapture:Boolean = false):void {
eventDispatcher.removeEventListener.apply(null, arguments);
}

public function willTrigger(type:String):Boolean {
return eventDispatcher.willTrigger.apply(null, arguments);
}
}
```

Класс CustomDispatcher не наследует EventDispatcher, вместо этого он реализует интерфейс IEventDispatcher и содержит в себе экземпляр EventDispatcher, который занимается работой с событиями.

Nirth: В FAQ на flasher.ru есть еще несколько статей о событиях:
Создание и рассылка собственных событий (http://flasher.ru/forum/showthread.php?t=79874)
Обработка событий (http://flasher.ru/forum/showthread.php?t=79873)

artcraft 01-10-2006, 16:34

Предыдущие версии Flash плеера не могли оборвать соединение после начала загрузки данных с сервера. Например, если вы начали загружать 50-ти мегабайтный swf в свой ролик, но хотите прекратить загрузку т.к. юзер решил не ждать загрузки а посмотреть что-то другое, у вас ничего не получится. Единственным способом избежать продолжения загрузки было бы закрыть плэйер (например перейти на другую html страницу)

Теперь, используя AS3, вы можете остановить соединения и оборвать загрузку запросов сделанных плеером. Рассмотрим класс Loader (flash.display.Loader). Это субкласс displayObjectContainer-а который загружает внешний контент в ваш ролик. Вы можете загружать содержимое в этот класс используя метод load. Что бы оборвать процесс загрузки, вы должны исползовать метод close.

Пример:
```
var loader:Loader = new Loader();
var request:URLRequest = new URLRequest("image.jpg");
loader.load(request);
addChild(loader);

// abort loading if not done in 3 seconds
var abortID:uint = setTimeout(abortLoader, 3000);

// abort the abort when loaded
loader.contentLoaderInfo.addEventListener(Event.CO MPLETE, abortAbort);

function abortLoader(){
try {
loader.close();
}catch(error:Error) {}
}
function abortAbort(event:Event){
clearTimeout(abortID);
```

}
Обратите внимания что метод close() был помещён внутрь блока try..catch. Это сделано потому что close будет выдавать ошибку (IOError) в том случае если в данный момент нету открытого соединения (хотя в данном случае этого никогда не произойдёт, потому что после окончания загрузки Timeout будет удалён - тем не менее это хорошая практика)

MerlinTwi 02-10-2006, 12:29

ActionScript 3 поддерживает регулярные выражения! Реализация аналогична что и в JavaScript. Для создания регулярного выражения можно использовать конструктор RegExp или строку заключенную в слеши (/), например:
var reCon:RegExp = new RegExp("\\w+", "i");
var reLit:RegExp = /\w+/i;
Методы RegExp:

RegExp.exec()
RegExp.test()

Методы строк (String), которые работают в регулярными выражениями:

String.match()
String.replace()
String.search()

MerlinTwi 02-10-2006, 13:14

В ActionScript 3 все события имеют свой класс. Базовые события расположены в классе Event (flash.events.Event (http://livedocs.macromedia.com/flex/2/langref/flash/events/Event.html)), события связанные с мышкой в классе MouseEvent (flash.events.MouseEvent (http://livedocs.macromedia.com/flex/2/langref/flash/events/MouseEvent.html)). Остальные классы событий расположены в пакете flash.events и все они потомки базового класса Event.
Вызывая EventDispatcher.dispatchEvent(), в качестве аргумента нужно передать экземпляр класса события Event (или любой потомок от Event). Например, для рассылки события "enterFrame" вызываем dispatchEvent, передавая ему экземпляр класса Event с типом (type) "enterFrame".
dispatchEvent(new Event("enterFrame"));
Когда вызывается функция обработчика события, ей в качестве аргумента передается этот экземпляр класса, из свойств которого можно узнать подробности о событии. Например, в type записан тип события.
addEventListener("enterFrame", eventHandler);
dispatchEvent(new Event("enterFrame"));
...
private function eventHandler(event:Event):void {
trace(event.type); // "enterFrame"
}
Тип события это строка (String), все возможные типы событий записаны в константах классов событий. Например, событие "enterFrame" лучше записывать так: Event.ENTER_FRAME. События от мышки расположены в классе MouseEvent, например, событие по клику мышки: MouseEvent.CLICK. Для работы с событиями лучше использовать эти константы, чем писать тип события строкой, это убережет вас от ошибок и опечаток. Пример выше правильнее записать так:
addEventListener(Event.ENTER_FRAME, eventHandler);
dispatchEvent(new Event(Event.ENTER_FRAME));
Для генерации своих событий, можно использовать класс Event, указывая свои типы событий или же, создать класс потомок от Event, прописав в нем константы новых типов событий.

MerlinTwi 02-10-2006, 13:17

В ActionScript 3 можно писать XML прямо в тексте скрипта, больше нет необходимости записывать XML в виде строки с последующим парсингом. Компилятор Flash проверяет синтаксис XML, что спасает от опечаток.
var myXml:XML =
<body>
<!-- comment -->
text1
<a>
text2

</body>;

MerlinTwi 04-10-2006, 11:45

В ActionScript 3 есть новая функция getQualifiedClassName (flash.utils.getQualifiedClassName (http://livedocs.macromedia.com/flex/2/langref/flash/utils/package.html#getQualifiedClassName())) с помощью которой можно запросто узнать имя класса:
var sprite:Sprite = new Sprite();
trace(getQualifiedClassName(sprite)); // "flash.display::Sprite"
Можно узнать и название предка (superclass)
trace(getQualifiedSuperclassName(sprite)); // "flash.display::DisplayObjectContainer"
Для обратного преобразования названия в класс можно использовать функцию getDefinitionByName

(flash.utils.getDefinitionByName
(http://livedocs.macromedia.com/flex/2/langref/flash/utils/package.html#getDefinitionByName()))
trace(getDefinitionByName("flash.display::Sprite")); // [class Sprite]
Для получения детальной информации о классе в виде XML используйте describeType() (flash.utils.describeType
(http://livedocs.macromedia.com/flex/2/langref/flash/utils/package.html#describeType()))
var sprite:Sprite = new Sprite();
var spriteDescription:XML = describeType(sprite);
trace (spriteDescription);

MerlinTwi 04-10-2006, 11:55

В ActionScript 2, вызвать super() можно было только в самом начале конструктора класса, иначе компилятор сообщит об ошибке «The superconstructor must be called first in the constructor body». В ActionScript 3, super можно вызывать в любом месте конструктора.
package {
public class SubClass extends SuperClass {
protected var value:int;
public function SubClass(value:int = 0) {
this.value = value;
super(20);
}
}
}

MerlinTwi 04-10-2006, 12:11

В класс MovieClip (flash.display.MovieClip
(http://livedocs.macromedia.com/flex/2/langref/flash/display/MovieClip.html)) добавлены новые методы для работы с метками фреймов: currentLabels и currentLabel.
currentLabels - это массив всех меток FrameLabel (flash.display.FrameLabel
(http://livedocs.macromedia.com/flex/2/langref/flash/display/FrameLabel.html)) которые есть на timeline в мувиклипе. Каждый FrameLabel содержит два свойства: name:String – название метки, и frame:int – номер фрейма.
currentLabel - возвращает название метки текущего фрейма.
var label:String = my_mc.currentLabel;

MerlinTwi 04-10-2006, 12:15

В ActionScript 3, функции trace можно передавать любое количество аргументов, все они будут выведены в Output panel.
trace(value1, value2, value3);
P.S. В AS1 и AS2 аналогичного эффекта можно добиться используя массив:
trace([value1, value2, value3]);

MerlinTwi 04-10-2006, 12:21

Обработчик события в ActionScript 3 должен получать один аргумент класса Event. Если же нужно самостоятельно вызывать функцию обработчика события, можно написать так:
myHandler(new Event(someEventType));
Что длинно и создает пустое ненужное событие, гораздо проще и удобнее в обработчике события присвоить дефолтное значение аргументу:
public function myHandler(event:Event = null):void {...}
...
// обычный вызов без события
myHandler();

MerlinTwi 04-10-2006, 12:32

Классы и функции работающие с URL в ActionScript 3 используют класс URLRequest (flash.net.URLRequest
(http://livedocs.macromedia.com/flex/2/langref/flash/net/URLRequest.html)), который кроме самой строки URL, содержит дополнительные свойства, более детально описывающие способ запроса:

contentType:String
data:Object
method:String
requestHeaders:Array
url:String

Например, для вызова navigateToURL() (flash.net.navigateToURL
(http://livedocs.macromedia.com/flex/2/langref/flash/net/package.html#navigateToURL())) (эта функция заменяет getURL() из предыдущих версий ActionScript) в качестве аргумента нужно передать экземпляр класса URLRequest:
var request:URLRequest = new URLRequest("http://www.adobe.com/");
navigateToURL(request);

__etc 04-10-2006, 14:12

name:int – название метки, и frame:String – номер фрейма.

Ошибся типом данных

MerlinTwi 05-10-2006, 19:46

В ActionScript 3 работа с XML приведена к стандарту E4X - ECMAScript's XML specification. Этот подход обеспечивает более удобные методы работы с узлами и атрибутами XML дерева. Все методы собраны в новом классе XML (top level XML (http://livedocs.macromedia.com/flex/2/langref/XML.html))
var myXML:XML = new XML(objectToConvert);
var myXML:XML = <node />;
Прежний XML класс, такой каким он был в ActionScript 1 и 2, так же остался, просто переименован в XMLDocument (flash.xml.XMLDocument (http://livedocs.macromedia.com/flex/2/langref/flash/xml/XMLDocument.html)). Если хотите работать с XML по старинке, то используйте XMLDocument вместо XML.
var myXML:XMLDocument = new XMLDocument("<node />");

MerlinTwi 05-10-2006, 20:02

В предыдущих версиях ActionScript для загрузки внешнего текста использовали LoadVars или XML. В ActionScript 3 все собрано в один класс URLLoader (flash.net.URLLoader
(http://livedocs.macromedia.com/flex/2/langref/flash/net/URLLoader.html)), который немного похож на прежний LoadVars. Из класса XML убрана функция по загрузки, теперь нужно загружать текст при помощи URLLoader, а затем передавать XML классу для преобразования.
Для начала загрузки нужно использовать метод load(), который получает один параметр URLRequest (это не просто строка URL!). Далее следует использовать события, для того, чтобы узнать об окончании загрузки или ошибке. Когда текст загружен, он сохраняется в свойстве data.
Пример:
var loader:URLLoader;
// ...
loader = new URLLoader();
loader.addEventListener(Event.COMPLETE, xmlLoaded);

var request:URLRequest = new URLRequest("file.xml");
loader.load(request);
//...
function xmlLoaded(event:Event):void {
var myXML:XML = new XML(loader.data);
//...
}

MerlinTwi 06-10-2006, 12:56

Дополнение к заметке про абстрактные классы. Для создания своего абстрактного класса, можно использовать такой код:

package {
public class Abstract {
public function Abstract() {
if(toString() == "[object Abstract]") throw(new Error("Abstract must be extended"));
}
}
}

MerlinTwi 09-10-2006, 15:37

В ActionScript 3 введен новый оператор “is” (is operator
(http://livedocs.macromedia.com/flex/2/langref/operators.html#is)) для проверки принадлежности экземпляра класса к типу класса или интерфейса.
var mySprite:Sprite = new Sprite();
trace(mySprite is Sprite); // true
trace(mySprite is DisplayObject); // true
trace(mySprite is IEventDispatcher); // true
Оператор is заменяет оператор instanceof (instanceof operator
(http://livedocs.macromedia.com/flex/2/langref/operators.html#instanceof)), использовавшийся в ActionScript 1 и 2. Оператор is проверяет цепочку наследования AS3 класса, в то время как instanceof проверяет цепочку прототипов (prototype). Отсюда несколько нюансов:
1. поскольку, AS3 классы воссоздают цепочку прототипов, то с помощью instanceof теоретически можно проверить принадлежность экземпляра класса к определенному типу AS3 класса, но не к интерфейсу:
var mySprite:Sprite = new Sprite();
trace(mySprite instanceof Sprite); // true
trace(mySprite instanceof DisplayObject); // true
trace(mySprite instanceof IEventDispatcher); // false – нет в цепочке прототипов
2. С помощью оператора is нельзя проверить тип динамического класса созданного не через наследование, а

через прототип (стиль AS1):
```
var AS1StyleClass:Function = function(){}
AS1StyleClass.prototype = new MovieClip();

var as1Instance:* = new AS1StyleClass();
trace(as1Instance instanceof AS1StyleClass); // true
trace(as1Instance instanceof MovieClip); // true
trace(as1Instance is AS1StyleClass); // true
trace(as1Instance is MovieClip); // false
```

В AS3 следует использовать только оператор is, прежний оператор instanceof хоть и поддерживается, но компилятор будет выдавать предупреждение:
The instanceof operator is deprecated, use the is operator instead.

MerlinTwi 09-10-2006, 15:57

Во Flash 9 вы можете ассоциировать timeline с классом, в том числе и рутовый. Timeline мувиклипа ассоциируется с классом так же как и в ActionScript 2 используя диалог linkage. Timeline рута может быть ассоциирован с классом в диалоге property inspector или в настройках публикации (publish settings - ActionScript settings).
Если вы написали код внутри фрейма в timeline, то он будет автоматически преобразован в класс. При этом переменные объявленные внутри какого-либо фрейма становятся свойствами класса, а функции методами класса. Поэтому вы не сможете объявить переменную или функцию с одинаковым названием в разных фреймах. Любой скрипт внутри фрейма ассоциируются с методом, который автоматически называется, когда происходит переход на этот фрейм.
Например, такой код написанный внутри фрейма:
```
var num:Number = 1;
function showNum():void {
trace(num);
}
showNum();
```
Будет преобразован в примерно такой класс:
```
package {
class TimelineClass extends MovieClip {

public var num:Number = 1;

public function showNum():void {
trace(this.num);
}
// метод автоматически вызываемый при переходе на первый фрейм
public function frame1():void {
showNum();
}
}
}
```

MerlinTwi 09-10-2006, 16:01

Пример, как можно использовать регулярное выражение (Top level RegExpx (http://livedocs.macromedia.com/flex/2/langref/RegExp.html)) для проверки корректности написания email:
```
function isValidEmail(email:String):Boolean {
var emailExpression:RegExp = /^[a-z][\w.-]+@\w[\w.-]+\.[\w.-]*[a-z][a-z]$/i;
return emailExpression.test(email);
}
//...
trace(isValidEmail("senocular@example.com")); // true
trace(isValidEmail("@example.com")); // false
trace(isValidEmail("senocular@example")); // false
trace(isValidEmail("seno\\cular@example.com")); // false
```

MerlinTwi 11-10-2006, 11:51

В ActionScript 3 для удобства работы с XML добавлен новый оператор “@”, предназначенный для доступа к атрибутам XML, и является аналогом метода attribute (Top level XML.attribute() (http://livedocs.macromedia.com/flex/2/langref/XML.html#attribute())).
```
var myXML:XML = <user name="senocular" id="2867" />;
trace(myXML.attribute("name")); // senocular
trace(myXML.attribute("id")); // 2867
trace(myXML.@name); // senocular
trace(myXML.@id); // 2867
```
В операторе @ можно использовать звездочку (*) для получения списка всех атрибутов узла XML в формате XMLList, это будет аналогом метода attributes() (Top level XML.attributes() (http://livedocs.macromedia.com/flex/2/langref/XML.html#attribute())). Пример:
```
var myXML:XML = <user name="senocular" id="2867" />;
```

```
var atts:XMLList;

atts = myXML.attributes();
trace(atts.toXMLString());
/* Output:
senocular
2867
*/
atts = myXML.@*;
trace(atts.toXMLString());
/* Output:
senocular
2867
*/
```

MerlinTwi 11-10-2006, 12:05

ActionScript 3 поддерживает распространение события (event propagation) – передача одного события нескольким объектам. К примеру, в ActionScript 1 и 2 события для кнопок (такие как onPress, onRelease, …) перехваченные мувиклипом никогда не дойдут до вложенных в него мувиклипов, даже если мышкой кликнули именно по вложенному мувиклипу. Например, кликаем мышкой по мувиклипу child_mc, который вложен в parent_mc:

```
// AS1 и AS2
parent_mc.onPress = function(){
trace("parent pressed");
}
parent_mc.child_mc.onPress = function(){
trace("child pressed");
}

/* trace при клике мышкой по child_mc:
parent pressed
*/
```

А в ActionScript 3 событие будет передано обоим мувиклипам:

```
// AS3
parent_mc.addEventListener(MouseEvent.CLICK, parentClick);
parent_mc.child_mc.addEventListener(MouseEvent.CLI CK, childClick);

function parentClick(event:MouseEvent):void {
trace("parent pressed");
}
function childClick(event:MouseEvent):void {
trace("child pressed");
}

/* trace при клике мышкой по child_mc:
child pressed
parent pressed
*/
```

MerlinTwi 12-10-2006, 16:00

Используя ActionScript 3 можно получить информацию о спектре проигрываемого flash-плеером в данный момент звука/музыки. Для этого можно вызывать статичный метод computeSpectrum класса SoundMixer (flash.media.SoundMixer (http://livedocs.macromedia.com/flex/2/langref/flash/media/SoundMixer.html)), и в качестве параметра передать массив ByteArray, куда и будет записана информация о спектре: 256 чисел с плавающей запятой (floating-point в диапазоне -1.0 … 1.0) для левого канала и следом 256 чисел для правого канала.

```
// запустить проигрывание музыки
// ...
var spectrumInfo:ByteArray = new ByteArray();
SoundMixer.computeSpectrum(spectrumInfo);
// spectrumInfo теперь содержит информацию о спектре
```

Эту информацию можно использовать для создания визуализации звука, как в популярных медиа-плеерах.

MerlinTwi 12-10-2006, 16:10

В ActionScript 1 и 2 преобразовывая строку в число, используя Number(), если строка начиналась с нуля “0”, то число интерпретировалось как восьмеричное. Если строка начиналась с “0x”, то число интерпретировалось как шестнадцатеричное:

```
// ActionScript 1 и 2
trace(Number("010")); // 8
```

Это могло создать определенные проблемы, если вы ожидали получить обычное десятичное число. Теперь ActionScript 3 интерпретирует строку, начинающуюся с нуля как десятичное число.

```
// ActionScript 3
trace(Number("010")); // 10
```

Если же нужно интерпретировать строку как восьмеричное число (или в любую другу систему счисления), то нужно использовать функцию parseInt:
trace(parseInt("010", 8)); // 8

MerlinTwi 13-10-2006, 10:48

Сборщик мусора (Garbage Collection - GC) автоматический процесс во Flash-плеере, который удаляет данные из памяти, когда они более не нужны. GC оценивает необходимость удаления по двум критериям: счетчик ссылок (reference counting) и очистка по меткам (mark and sweep)

1. Счетчик ссылок - это процесс, который следит за количеством переменных, ссылающихся на объекты в памяти.
var a:Object = new Object(); //создается новый объект и новая переменная "a", которая ссылается на этот объект, счетчик ссылок = 1
var b:Object = a; // еще одна ссылка на тот же объект, счетчик ссылок = 2
Когда не останется ни одной ссылки на объект, то он будет удален из памяти.
a=null; // счетчик ссылок стал равен 1
b=null; // счетчик ссылок стал равен 0, и объект будет удален из памяти
Есть случаи, когда счетчик ссылок не сработает, например:
var a:Object = new Object(); // счетчик ссылок на первый объект = 1
var b:Object = new Object(); // счетчик ссылок на второй объект = 1
a.b = b; // счетчик ссылок на второй объект = 2
b.a = a; // счетчик ссылок на первый объект = 2
a=null; // счетчик ссылок на первый объект = 1
b=null; // счетчик ссылок на второй объект = 1
Обе переменные "a" и "b" более не ссылаются на объекты, но первый объект доступен из второго и второй доступен из первого. В данном случае счетчик ссылок не сработает, пора приниматься за дело процессу очистке по меткам.

2. Очистка по меткам – это процесс, который сканирует все объекты от базового класса (root или stage) и помечает каждого, кого нашел. Все не найденные объекты недоступны и потому будут удалены. Из предыдущего примера про "a" и "b", поскольку "a" и "b" более недоступны из root они не будут помечены и будут удалены.
[root] <- сканирование...
[objectRef (помечен)] <- сканирование...
[objectRef (помечен)] <- сканирование...
[objectRef (помечен)] <- сканирование...
[objectRef (помечен)] <- сканирование...
[objectRef (помечен)] <- сканирование...
...
[удалить все непомеченное]
Очистка по меткам очень ресурсоемкий процесс, поэтому выполняется редко, решение о том, когда запустить процессы удаления данных из памяти принимает flash-плеер сам, и программно инициировать эти процессы мы не можем.

MerlinTwi: Хочу обратить ваше внимание на то, что добавление обработчиков событий к объекту (или setInterval) не позволит сборщику мусора (GC) удалить этот объект, т.к. на него остаются ссылки из процессов рассылающих события. Например:
init();

```
function init():void {
var test:Sprite = new Sprite();
test.addEventListener(Event.ENTER_FRAME, handle);
}

function handle(e:Event):void {
trace("enterFrame");
}
```

Локальная переменная test, ссылающаяся на созданный спрайт удалена, сам спрайт недоступен из root или stage, но он не будет удален из памяти т.к. спрайт слушает событие enterFrame, и функция handle будет продолжать срабатывать. Для того чтобы объект был удален, нужно освобождать слушаемые им события, когда они более не нужны:
test.removeEventListener(Event.ENTER_FRAME, handle);

MerlinTwi 13-10-2006, 11:29

В ActionScript 3 есть две возможности поставить мягкую ссылку (Weak Reference) на объект, которая не будет учитываться счетчиком ссылок.
1. При использовании класса Dictionary его конструктору можно передать параметр true, чтобы использовались мягкие ссылки на объекты.
var dict:Dictionary = new Dictionary(true); // использовать мягкие ссылки
В этом случае Dictionary в качестве ключей будет использовать мягкие ссылки на объекты, которые не учитываются счетчиком ссылок и не мешают удалению объектов из памяти.
var obj:Object = new Object();
dict[obj] = true;
delete obj; // сборщик мусора удалит объект obj, поскольку ссылка на него из dict мягкая
2. Можно добавить слушатель события с мягкой ссылкой. У addEventListener последний параметр указывает, использовать мягкую ссылку или обычную (по умолчанию). При использовании мягкой ссылки, не удаленный

обработчик события не помешает удалению объекта из памяти.
// addEventListener(type:String, listener:Function, useCapture:Boolean = false, priority:int = 0, useWeakReference:Boolean = false):void
addEventListener(MouseEvent.CLICK, clickHandler, false, 0 true); // использовать мягкую ссылку

MerlinTwi: Для понимания мягкой ссылки в addEventListener, рассмотрим случай из предыдущего примера (http://flasher.ru/forum/showpost.php?p=577862&postcount=39). Используем мягкую ссылку при создании слушателя события enterFrame:
```
init();

function init():void {
var test:Sprite = new Sprite();
test.addEventListener(Event.ENTER_FRAME, handle, false,0,true);
}

function handle(e:Event):void {
trace("enterFrame");
}
```

Теоретически сборщик мусора должен удалить спрайт, т.к. на него не остается ссылок из переменных, из root он недоступен, и обработчик события enterFrame стоит мягкий. Вот только когда сработает сборщик мусора, решает flash-плеер сам… через секунду, через час, день? И до тех пор, пока сборщик мусора не удалит спрайт из памяти, функция handle будет исправно срабатывать каждый фрейм.
Экспериментально был найден способ заставить сборщик мусора сработать немедленно, нужно создать много объектов для которых потребуется выделение большого куска памяти. Подтолкнем таким способом сборщик мусора сработать по клику мышки:
```
init();
addEventListener(MouseEvent.CLICK, immediatelyGC);

function init():void {
var test:Sprite = new Sprite();
test.addEventListener(Event.ENTER_FRAME, handle, false,0,true);
}

function handle(e:Event):void {
trace("enterFrame");
}

function immediatelyGC(e:Event):void {
trace("run GC");
var ar:Array = new Array();
for (var i:int=0; i<100000; i++) {
ar.push( new Object() );
}
}
```

В трейсе будет:
```
. . .
enterFrame
enterFrame
enterFrame
run GC
```

После клика мышки и запуска функции immediatelyGC, сборщик мусора удалил спрайт из памяти и более функция handle не вызывается.

MerlinTwi 16-10-2006, 11:51

В AS3 добавлено новое событие Event.RENDER (flash.events.Event.RENDER (http://livedocs.macromedia.com/flex/2/langref/flash/events/Event.html#RENDER)), которое рассылается после enterFrame, но перед тем, как экран флеш-плеера обновится. Это событие не генерится автоматически, его нужно активировать самостоятельно, вызвав метод stage.invalidate(); (flash.display.Stage.invalidate() (http://livedocs.macromedia.com/flex/2/langref/flash/display/Stage.html#invalidate())), после чего, перед обновлением экрана все объекты приаттаченные на stage (и только они) получат событие Event.RENDER. Если флэш плеер не занимается рендерингом (например окно минимизировано в таск бар), то Event.RENDER не рассылается.

Пример:
```
var sprite:Sprite = new Sprite();
stage.addChild(sprite);

sprite.addEventListener(Event.ENTER_FRAME, enterFrame);
sprite.addEventListener(Event.RENDER, render);
stage.addEventListener(MouseEvent.CLICK, click);

function enterFrame(event:Event):void {
trace("enter frame");
```

```
}
function render(event:Event):void {
trace("render");
}
function click(event:MouseEvent):void {
trace("click");
stage.invalidate();
}
```
Output:
```
enter frame
enter frame
enter frame
click
enter frame
render
enter frame
enter frame
enter frame
enter frame
...
```

MerlinTwi 16-10-2006, 11:52

Еще во Flash 8 была добавлена возможность загрузить картинку из библиотеки (library) используя linkage ID и метод BitmapData.loadBitmap(). В ActionScript 3 это делается несколько иначе, т.к. все мувиклипы, фреймы и объекты в библиотеке ассоциированы с классами. Классы для картинок в библиотеке являются потомками BitmapData (flash.display.BitmapData (http://livedocs.macromedia.com/flex/2/langref/flash/display/BitmapData.html)) и перед добавлением картинки к какому-либо визуальному объекту нужно ее связать с классом Bitmap (flash.display.Bitmap (http://livedocs.macromedia.com/flex/2/langref/flash/display/Bitmap.html)), поскольку непосредственно BitmapData приаттачить нельзя.
В качестве примера, добавьте в библиотеку картинку, в диалоге linkage поставьте галочку «Export for ActionScript» и укажите название класса «RomeImage» (этот класс будет сгенерирован автоматически при публикации SWF).
Далее нужно написать код для добавления этой картинки на экран:
```
// создаем класс RomeImage с содержимым картинки
var romeImageData:RomeImage = new RomeImage();

// Создаем Bitmap на основе нашей картинки
var romeImageBitmap:Bitmap = new Bitmap(romeImageData);

// и добавляем этот Bitmap на экран
addChild(romeImageBitmap);
```

MerlinTwi 16-10-2006, 12:08

Оператор typeof позволяет узнать базовый тип любого объекта/переменной. Это не информация о классе, а просто тип переменной связанной с объектом. Для детальной информации о классе следует использовать instanceof, getQualifiedClassName, describeType.
В ActionScript 1 и 2, typeof возвращал такие значения:

```
boolean
function
movieclip
null
number
object
string
undefined
```

BActionScript 3 typeof возвращает:

```
boolean
function
number
object
string
xml
undefined
```

Убрали MovieClip и null, теперь это тоже самое, что object. Добавили xml. Новые типы переменных int и uint распознаются как number. И простейшие типы переменных boolean, number, string, созданные через конструктор распознаются правильно, а не как object, как было в AS1/AS2.
```
// AS1 & AS2
trace(typeof new XML()); // object

trace(typeof my_mc); // movieclip
```

```
trace(typeof null); // null

trace(typeof true); // boolean
trace(typeof 1); // number
trace(typeof ""); // string

trace(typeof new Boolean()); // object
trace(typeof new Number()); // object
trace(typeof new String()); // object

// AS3
trace(typeof new XML()); // xml

trace(typeof my_mc); // object

trace(typeof null); // object

trace(typeof true); // boolean
trace(typeof 1); // number
trace(typeof ""); // string

trace(typeof new Boolean()); // boolean
trace(typeof new Number()); // number
trace(typeof new String()); // string

trace(typeof int(1)); // number
trace(typeof uint(1)); // number
```

MerlinTwi 17-10-2006, 11:06

Как и в ActionScript 1 и 2, в ActionScript 3 есть метод getBounds() (flash.display.DisplayObject.getBounds() (http://livedocs.macromedia.com/flex/2/langref/flash/display/DisplayObject.html#getBounds())), который вычисляет границу мувиклипа в системе координат любого объекта на timeline. Но в ActionScript 3 метод getBounds возвращает Rectangle (flash.geom.Rectangle (http://livedocs.macromedia.com/flex/2/langref/flash/geom/Rectangle.html)) вместо Object со свойствами xMin, xMax, yMin, yMax, как это было раньше.
В ActionScript 3 еще добавлен новый метод getRect() (flash.display.DisplayObject.getRect() (http://livedocs.macromedia.com/flex/2/langref/flash/display/DisplayObject.html#getRect())), который выполняет аналогичную задачу, но в отличие от getBounds не учитывает толщину линий (strokes on shapes).
В качестве наглядного примера, красный прямоугольник высчитывается по getBounds, а синий по getRect:

```
var sprite:Sprite = new Sprite();
sprite.graphics.beginFill(0x999999);
sprite.graphics.lineStyle(10, 0x333);
sprite.graphics.drawCircle(100, 100, 50);
sprite.graphics.endFill();
addChild(sprite);

addChild(createRectShape(sprite.getRect(this), 0x0000FF));
addChild(createRectShape(sprite.getBounds(this), 0xFF0000));

function createRectShape(rect:Rectangle, color:uint):Shape {
var rectShape:Shape = new Shape();
rectShape.graphics.lineStyle(0, color);
rectShape.graphics.drawRect(rect.left, rect.top, rect.width, rect.height);
return rectShape;
}
```

MerlinTwi 17-10-2006, 11:19

В ActionScript 3 добавлен новый вид цикла for each..in (http://livedocs.macromedia.com/flex/2/langref/statements.html#for_each..in) который работает аналогично прежнему for..in (http://livedocs.macromedia.com/flex/2/langref/statements.html#for..in), но перебираются значения объекта, а не ключи. Пример:

```
var object:Object = new Object();
object.name = "senocular";
object.id = 2867;
object.isModerator = true;
for each (var value:* in object){
trace(value);
}
/* Output:
true
2867
senocular
*/
```

```
// А прежний for..in:
for (var key:String in object){
trace(key + ": " + object[key]); // object[key] is value
}
/* Output:
isModerator: true
id: 2867
name: senocular
*/
```

Учтите, в цикле for each..in нельзя получить значение ключа.
Так же обратите внимание, что ActionScript 3 в циклах (for..in и for each..in) сохраняет порядок элементов в массиве, а ActionScript 1 и 2 перебирал в порядке обратном добавлению элементов. См. пример:

```
var array:Array = new Array();
array[1] = 1;
array[0] = 2;
array[2] = 3;
for (var key:String in array){
trace("array[" + key + "] = "+ array[key]);
}
```

Output AS 1 & AS 2:

```
array[2] = 3
array[0] = 2
array[1] = 1
```

Output AS 3:

```
array[0] = 2
array[1] = 1
array[2] = 3
```

MerlinTwi 17-10-2006, 11:25

ActionSript 3 дает возможность присвоить значения по умолчанию для параметров функций. Такие параметры становятся необязательными при вызове функции.

```
function method(required:String, optional:String = "default"):void {
trace(required +" "+optional);
}
method("Hello"); // "Hello default"
```

Необязательные параметры со значениями по умолчанию должны стоять после обязательных параметров.

```
// Так неправильно:
function method(required:String = "default", optional:String):void { ... }
// [Compiler] Error #1138: Required parameters are not permitted after optional parameters.
```

MerlinTwi 18-10-2006, 11:18

Для того, чтобы в ActionSript 3 функция могла принимать любое количество аргументов необходимо использовать специальный параметр три точки
«... название_переменной» (Keyword: ...(rest) (http://livedocs.macromedia.com/flex/2/langref/statements.html#..._(rest)_parameter)), который всегда располагается последним, после перечисления обязательных аргументов.
Переданные аргументы записываются в указанную переменную в виде массива:

```
usingRest(1, 2, 3, 4);
function usingRest(required:Number, ... optionalArgs):void {
trace(required); // 1
trace(optionalArgs); // [2, 3, 4]
}
```

MerlinTwi 18-10-2006, 11:25

В ActionScript 3, как и в ActionScript 1 и 2, любая функция содержит arguments (Top level arguments (http://livedocs.macromedia.com/flex/2/langref/arguments.html)), куда в виде массива записываются все переданные аргументы. Но в ActionScript 3 убрали свойство arguments.caller, для получения ссылки на вызвавшую функцию советуют передавать эту ссылку самостоятельно в виде аргумента функции.

```
args("a", 1);
function args(str:String, num:Number):void {
trace(arguments); // ["a", 1]
}
```

MerlinTwi 18-10-2006, 12:06

ActionScript 3 поддерживает пространство имен (namespaces) в классах по аналогии с namespaces в XML.
Пространство имен можно сравнить с пакетами (packages), поскольку пакеты позволяют создать несколько различных классов с одинаковым названием, но описанных в разных пакетах, так и пространство имен дает возможность определить несколько разных методов и свойств с одним названием внутри одного класса.
Для использования пространства имен необходимо этому пространству присвоить уникальное название с помощью ключевого слова namespace (namespace Keyword

(http://livedocs.macromedia.com/flex/2/langref/statements.html#namespace)). После объявления пространства имен его название можно использовать аналогично public, private…
Пример:
```
package {

public class UsingNameSpaces {

public namespace company;
public namespace individual;

company var value:int = 10;
individual var value:int = 2;

public function UsingNameSpaces(){
}

company function showValue() {
}

individual function showValue() {
}
}
}
```
Здесь было объявлено два пространства имен «company» и «individual», которые использовались для разделения свойств «value» и методов «showValue», которые хоть они и имеют одинаковые названия, конфликта не возникает, поскольку расположены в разных пространствах имен.
При объявлении пространства имен можно дополнительно указать URI.
```
package {

public class UsingNameSpaces {

public namespace company = "http://www.example.com/company";
public namespace individual = "http://www.example.com/individual";

company var value:int = 10;
individual var value:int = 2;

public function UsingNameSpaces(){
}

company function showValue() {
}

individual function showValue() {
}
}
}
```

MerlinTwi 18-10-2006, 12:18

При вызове метода класса, или обращении к свойству, описанному внутри пространства имен, необходимо уточнить из какого именно пространства имен нужно вызвать метод. Один из способов сделать это - использовать оператор :: (name qualifier operator (http://livedocs.macromedia.com/flex/2/langref/operators.html#name_qualifier)). Указываете название пространства имен, два двоеточия и название метода (namespace::member).
```
package {

public class UsingNameSpaces {

public namespace company = "http://www.example.com/company";
public namespace individual = "http://www.example.com/individual";

company var value:int = 10;
individual var value:int = 2;

public function UsingNameSpaces(){
company::showValue(); // traces 10
individual::showValue(); // traces 2
}

company function showValue() {
trace(company::value);
}

individual function showValue() {
trace(individual::value);
}
}
```

}
Несмотря на то, что метод showValue расположен внутри такого же пространства имен, что и выводимое свойство value, необходимо уточнять пространство имен.

MerlinTwi 19-10-2006, 12:01

Как и в ActionScript 2 в ActionScript 3 есть возможность создавать динамические классы, для этого используется ключевое слово dynamic (dynamic keyword (http://livedocs.macromedia.com/flex/2/langref/statements.html#dynamic)). В динамические классы можно добавлять новые свойства в процессе выполнения кода.
В AS2, когда создаете новый класс, наследованный от динамического класса, он по умолчанию так же становился динамическим.

```
// superclass.as
dynamic class superclass {
}

// subclass.as
class subclass extends superclass {
}

// main movie
var instance:subclass = new subclass();
trace(instance.anything); // Ошибки не будет, т.к. subclass унаследовал динамичность
```

В AS3 это не так, любой класс не является динамическим, если это явно не указано ключевым словом dynamic.

```
// superclass.as
package {
dynamic class superclass {
}
}

// subclass.as
package {
class subclass extends superclass {
}
}

// main movie
var instance:subclass = new subclass();
trace(instance.anything); // Ошибка, свойство не определено, т.к. класс не динамический
```

MerlinTwi 19-10-2006, 12:23

В Director Lingo есть удобное событие mouseWithin, которое генерится каждый фрейм, если мышь находится в пределах мувиклипа. Это событие довольно просто реализовать на ActionScript 3. (Конечно можно и на ActionScript 1 или 2, но придется заюзать методы onRollOver и onEnterFrame, а они могут понадобиться и для других целей. В AS3 такой проблемы нет, т.к. вы можете на одно событие навесить сколько угодно слушателей.)
Итак, для реализации события mouseWithin нужно, по событию входа мышки на мувиклип подключиться к событию enterFrame и рассылать каждый фрейм mouseWithin, пока мышь не покинет пределы мувиклипа.

```
// Реализация события mouseWithin
private function addMouseWithin(event:MouseEvent):void {
addEventListener(Event.ENTER_FRAME, mouseWithin);
}
private function removeMouseWithin(event:MouseEvent):void {
removeEventListener(Event.ENTER_FRAME, mouseWithin);
}
private function mouseWithin(event:Event):void {
dispatchEvent(new MouseEvent("mouseWithin"));
}

// В конструкторе
public function MySpriteClass() {
addEventListener(MouseEvent.MOUSE_OVER, addMouseWithin);
addEventListener(MouseEvent.MOUSE_OUT, removeMouseWithin);
}
```

Для обработки такого события просто добавьте слушателя:

```
addEventListener("mouseWithin", mouseWithinHandler);
```

MerlinTwi 19-10-2006, 12:38

Ключевое слово final (Toplevel final keyword (http://livedocs.macromedia.com/flex/2/langref/statements.html#final)) можно использовать для предотвращения переопределения метода класса или запрещения наследования от класса.
Если метод помечен как final, классы потомки не смогут переопределить его (override).

```
final public function methodName() { … }
```

Если пометить весь класс как final, то от него нельзя будет создать потомков. Например:

```
// superclass.as
package {
final public class finalclass {
}
}

// subclass.as
package {
class subclass extends finalclass {
}
}
```

Компилятор выдаст ошибку: Error #1016: Base class is final.
Примечание: нет смысла делать final методы в final классе, т.к. от этого класса все равно не может быть наследников, а значит, и нет возможности переопределить методы.

MerlinTwi 20-10-2006, 10:51

Если вы используете MXMLC (компилятор из Flex 2 SDK (http://www.adobe.com/products/flex/sdk/)) для компиляции ActionScript 3 SWFs, вы можете использовать SWF metadata tag для назначения свойств флешки. Всего можно установить 4 свойства:

width
height
frameRate
backgroundColor

Пример:

```
package {
[SWF(width="500", height="450", frameRate="24", backgroundColor="#FFFFFF")]
public class MyApp extends Sprite {

}
}
```

После компиляции будет создан SWF размером 500x450 с белым фоном и частотой кадров 24.

MerlinTwi 20-10-2006, 11:35

В ActionScript 3 добавлен новый класс Proxy (flash.utils.Proxy (http://livedocs.macromedia.com/flex/2/langref/flash/utils/Proxy.html)) для замены методов addProperty и __resolve из ActionScript 1 и 2.

addProperty(prop:String, getFunc:Function, setFunc:Function) : Boolean
использовался для динамического добавления getter/setter свойств в объект или класс. Параметры:
prop - имя создаваемого свойства объекта.
getFunc - функция, которая вызывается для возвращения значения свойства;
setFunc - функция, которая вызывается для установки значения свойства;
__resolve - ссылка на определяемую пользователем функцию, которая вызывается, если ActionScript код ссылается на неопределенное свойство или метод.

В ActionScript 3 вы можете создать класс наследник от класса Proxy, который предоставляет следующие возможности:

Перехватить чтение свойства
Перехватить запись свойства
Перехватить проверку свойства (на наличие)
Перехватить удаление свойства
Перехватить вызов метода
Перехватить установку атрибутов
Использовать [] для доступа к свойствам

Хоть возможности Proxy и превосходят то, что было доступно в AS1 и AS2, здесь есть и свои недостатки. Вы обязаны наследоваться от класса Proxy, нельзя унаследоваться от произвольного класса, например от Display objects, и получить возможности класса Proxy.
Proxy классы обычно используются для создания набора переменных с гибкими возможностями (например, Flex классы ArrayCollection, ListCollectionView, XMLListCollection).

Proxy класс более детально будет рассмотрен в следующих советах

MerlinTwi 20-10-2006, 11:44

В ActionScript 3 добавлен новый оператор in (in operator
(http://livedocs.macromedia.com/flex/2/langref/operators.html#in)), который используется для проверки наличия

свойства в объекте. Это аналогично hasOwnProperty, но работает и для унаследованных свойств.
```
trace("PI" in Math); // true
trace("myProperty" in Math); // false
```
Так же оператор in можно использовать для проверки правильности номера индекса в массиве:
```
var myArray:Array = ["zero", "one", "two"];
trace(0 in myArray); // true
trace(1 in myArray); // true
trace(2 in myArray); // true
trace(3 in myArray); // false
```

MerlinTwi 23-10-2006, 12:18

Методы getProperty и setProperty используются для управления необъявленными свойствами класса. Если какое-то свойство явно не объявлено в классе, то при попытке прочитать его значение, будет вызван метод getProperty. Аналогично при записи вызывается метод setProperty.
Как и все методы класса Proxy, getProperty и setProperty объявлены в пространстве имен «flash_proxy» (flash.utils.flash_proxy), чтобы не возникало конфликтов с публичными методами. Для переопределения этих методов в своем классе обязательно нужно указывать пространство имен flash_proxy.
Пример: класс CustomObject наследуется от Proxy и переопределяет методы getProperty и setProperty для управления динамическими свойствами.
```
package {

import flash.utils.Proxy;
import flash.utils.flash_proxy;

dynamic public class CustomObject extends Proxy {

public var classProperty:String = "classProperty"; // объявленное свойство
private var customProperties:Object = new Object(); // хранилище динамических свойств

public function CustomObject() {
}

// Вызывается при чтении динамического свойства
override flash_proxy function getProperty(name:*):* {
if (name in customProperties) {
return customProperties[name];
}
return "Property does not exist";
}
// Вызывается при записи в динамическое свойство
override flash_proxy function setProperty(name:*, value:*):void {
customProperties[name] = "Property "+name+": "+value;
}
}
}
// Пример использования
var myObj:CustomObject = new CustomObject();
trace(myObj.foo); // Property does not exist
myObj.foo = "bar";
trace(myObj.foo); // Property foo: bar

trace(myObj.classProperty); // classProperty
myObj.classProperty = "bar";
trace(myObj.classProperty); // bar
```

Обратите внимание, что методы getProperty и setProperty вызываются при попытке доступа к необъявленным свойствам класса. В примере, при доступе к явно объявленному свойству «classProperty» эти методы не вызывались.

MerlinTwi 23-10-2006, 12:43

Во Flash 9, когда вы размещаете MovieClip на сцене (timeline) с указанием имени (instance name), автоматически происходят сразу два действия:
1. Вы присваиваете методу «name» этого мувиклипа строку эквивалентную instance name.
2. Вы создаете переменную в timeline с именем аналогичным instance name мувиклипа со ссылкой на этот мувиклип.
Флеш делает все это самостоятельно при компиляции SWF файла, чтобы помочь вам управлять мувиклипами на сцене. Важно понимать, что при создании мувиклипами средствами ActionScript, эти действия не выполняются автоматически.
К примеру:
```
// Здесь «my_mc» это instance name мувиклипа на сцене
trace(my_mc); // [object MovieClip]
trace(my_mc.name); // my_mc

// Создаем мувиклип с помощью AS и добавляем его к my_mc
```

```
var another_mc:MovieClip = new MovieClip();
another_mc.name = "child_mc";
my_mc.addChild(another_mc);

// Переменная для доступа к созданному мувиклипу
// автоматически не создается в my_mc
trace(another_mc); // [object MovieClip]
trace(my_mc.child_mc); // undefined
trace(my_mc.another_mc); // undefined
```

Если нужно получить ссылку на мувиклип (или любой DisplayObject) по его имени (свойству name), нужно использовать метод getChildByName();

```
trace(my_mc.getChildByName("child_mc")); // [object MovieClip];
```

MerlinTwi: В дополнение к примеру стоит отметить, что поскольку любой мувиклип размещенный на timeline является динамическим, то можно просто создать в нем новое свойство ссылающееся на созданный мувиклип.

```
// Создаем мувиклип с помощью AS и добавляем его к my_mc
var another_mc:MovieClip = new MovieClip();
another_mc.name = "child_mc";
my_mc.addChild(another_mc);
// создаем в my_mc новое свойство «another_mc» для ссылки на новый мувиклип
my_mc.another_mc = another_mc;
// проверяем
trace(my_mc.another_mc); // [object MovieClip]
```

MerlinTwi 23-10-2006, 12:57

ActionScript 3 позволяет определять константы, для этого используется ключевое слово const (const keyword (http://livedocs.macromedia.com/flex/2/langref/statements.html#const)), которое записывается вместо var (var keyword (http://livedocs.macromedia.com/flex/2/langref/statements.html#var)). Обычно константы записывают большими буквами, разделяя слова подчеркиванием, например, Event.ENTER_FRAME это константа.
Пример объявления константы:

```
package {

import flash.display.Sprite;

public class MyClass extends Sprite {

public const MY_CONSTANT:String = "constant";

public function MyClass () {
}
}
}
```

Константу нельзя переопределить. Хотя, если константа является ссылкой на объект, например массивом, то в такой массив можно добавлять новые элементы, или удалять старые, но нельзя присвоить константе ссылку на другой массив.

```
package {

import flash.display.Sprite;

public class MyClass extends Sprite {

public const MY_CONSTANT:Array = new Array(1,2,3);

public function MyClass () {
MY_CONSTANT.push(4); // ok
MY_CONSTANT = new Array(5,6,7); // error
}
}
}
```

MerlinTwi 24-10-2006, 11:58

В ActionScript 3 больше нет метода duplicateMovieClip, но его можно частично воспроизвести. Например, такая функция:

```
package com.senocular.display {

import flash.display.DisplayObject;
import flash.geom.Rectangle;

/**
* duplicateDisplayObject
* Создает дубликат DisplayObject,
* аналогично методу duplicateMovieClip из AVM1
* @param target Дублируемый объект
```

```
* @param autoAdd Если true, то созданный дубликат будет добавлен
* на сцену туда же, где и оригинал
* @return возвращает ссылку на дубликат
*/
public function duplicateDisplayObject(target:DisplayObject, autoAdd:Boolean = false):DisplayObject {
// Создаем дубликат (constructor в кавычках, чтобы компилятор не ругался в strict mode)
var targetClass:Class = target["constructor"];
var duplicate:DisplayObject = new targetClass();

// Дублируем свойства
duplicate.transform = target.transform;
duplicate.filters = target.filters;
duplicate.cacheAsBitmap = target.cacheAsBitmap;
duplicate.opaqueBackground = target.opaqueBackground;
if (target.scale9Grid) {
var rect:Rectangle = target.scale9Grid;
// Баг, Flash 9 возвращает scale9Grid в 20 раз больше чем на самом деле
rect.x /= 20, rect.y /= 20, rect.width /= 20, rect.height /= 20;
duplicate.scale9Grid = rect;
}

// Добавить дубликат к target.parent
// если autoAdd установлен в true
if (autoAdd && target.parent) {
target.parent.addChild(duplicate);
}
return duplicate;
}
}
```

Как видите, все довольно просто, создается новый объект из того же класса, что и оригинал и дублируются свойства: transform, фильтры, кэширование и т.п.
Замечание: на данный момент во Flash Player 9 есть баг, некорректно возвращается свойство scale9Grid, эта функция исправляет ошибку, но когда баг в плеере исправят, строчку:
rect.x /= 20, rect.y /= 20, rect.width /= 20, rect.height /= 20; нужно будет удалить.
Пример использования функции:

```
import com.senocular.display.duplicateDisplayObject;

// Создаем дубликат мувиклипа myOldSprite
// автоматически добавляя его на экран (autoAdd=true)
var newInstance: DisplayObject = duplicateDisplayObject(myOldSprite, true);
newInstance.x += 100; // сдвинуть правее, чтобы увидеть дубликат
```

Учтите, эта функция не является полным аналогом duplicateMovieClip, т.к. не может продублировать что-то динамически нарисованное, только объекты, имеющие свой класс.

MerlinTwi 24-10-2006, 12:15

Если в классе наследнике от Proxy попытаться вызвать несуществующий метод, то будет вызван метод callProperty, которому передается название метода с массивом аргументов. Это можно использовать для создания динамических методов.

Как и все методы класса Proxy, callProperty объявлен в пространстве имен «flash_proxy» (flash.utils.flash_proxy), чтобы не возникало конфликтов с публичными методами. Для переопределения метода в своем классе обязательно нужно указывать пространство имен flash_proxy.

Пример: класс CustomObject наследуется от Proxy и переопределяет метод callProperty для управления динамическими методами.

```
package {

import flash.utils.Proxy;
import flash.utils.flash_proxy;

dynamic public class CustomObject extends Proxy {

private var variables:Object = new Object(); // хранилище переменных

public function CustomObject() {
}

// Вызывает при обращении к динамическому методу
override flash_proxy function callProperty(name:*, ... args):* {

// Преобразуем name в строку
name = String(name);

var callType:String = name.slice(0,3); // get или set
var callVariable:String = name.slice(3); // название переменной
```

```
switch(callType) {

case 'get':
// Если «get», то возвращает значение переменной, если она существует
if (callVariable in variables) {
return variables[callVariable];
}
return null;

case 'set':
// Если «set» запомним новую переменную
variables[callVariable] = args[0];
}
}
}
}
```
В этом примере, с использованием callProperty, реализованы динамические методы для чтения и записи «переменных».
```
// пример использования
var myObj:CustomObject = new CustomObject();
trace(myObj.getMyVar()); // null
myObj.setMyVar("foo");
trace(myObj.getMyVar()); // foo
myObj.setMyVar("bar");
trace(myObj.getMyVar()); // bar
trace(myObj.MyVar); // не существует (get property error)
```

В начале getMyVar возвращает null, поскольку переменная «MyVar», еще не определена. После записи при помощи setMyVar, getMyVar возвращает записанное значение.

MerlinTwi 24-10-2006, 12:42

В ActionScript 3 нет возможности непосредственно скопировать графику созданную в Graphics. Но можно использовать класс Proxy для записи всех вызванных методов, чтобы потом их воспроизвести при копировании. Например, такой вариант:
```
package com.senocular.display {

import flash.display.Graphics;
import flash.utils.flash_proxy;
import flash.utils.Proxy;

/**
* Класс для расширения graphics с возможностью создания копии рисунка
*/
public class GraphicsCopy extends Proxy {

private var _graphics:Graphics;
private var history:Array = new Array();

/**
* Ссылка на graphics
*/
public function get graphics():Graphics {
return _graphics;
}
public function set graphics(g:Graphics):void {
_graphics = g;

// Создаем копию рисунка на новом Graphics
copy(this);
}

/**
* Конструктор
* @param graphics Ссылка на Graphics рисование по которому запоминаем
*/
public function GraphicsCopy(graphics:Graphics = null) {
_graphics = graphics;
}

/**
* Копирует сюда рисунок из другого класса GraphicsCopy
*/
public function copy(graphicsCopy:GraphicsCopy):void {
var hist:Array = graphicsCopy.history;
history = hist.slice();
```

```
if (_graphics) {
var i:int;
var n:int = hist.length;
_graphics.clear();
for (i=0; i<n; i += 2) {
_graphics[hist[i]].apply(_graphics, hist[i + 1]);
}
}
}

// Перехват вызова несуществующего метода, с помощью этого
// метода PROXY и реализована запись процесса рисования.
override flash_proxy function callProperty(methodName:*, ... args):* {
methodName = String(methodName);
switch(methodName) {
case "clear":
history.length = 0;
break;
default:
history.push(methodName, args);
}
if (_graphics && methodName in _graphics) {
return _graphics[methodName].apply(_graphics, args);
}
}
}
}
```

Теперь достаточно при рисовании вместо свойства graphics использовать экземпляр класса GraphicsCopy, который запоминает историю вызванных при рисовании методов и может ее воспроизвести, при создании копии.
Пример:

```
// MyClass.as Класс наследник от Shape с использованием GraphicsCopy
package {

import flash.display.Shape;
import com.senocular.display.GraphicsCopy;

class MyShape extends Shape {

// свойство graphicsCopy
private var _graphicsCopy:GraphicsCopy;
public function get graphicsCopy():GraphicsCopy {
return _graphicsCopy;
}

// Конструктор
function MyShape(){
_graphicsCopy = new GraphicsCopy(graphics);
}
}
}


// Пример использования
// Что-то рисуем на shape 1
var shape1:MyShape = new MyShape();
shape1.graphicsCopy.beginFill(0xFF80);
shape1.graphicsCopy.lineStyle(2, 0);
shape1.graphicsCopy.drawRect(0, 0, 50, 50);

// Делаем копию рисунка на shape2
var shape2:MyShape = new MyShape();
shape2.graphicsCopy.copy(shape1.graphicsCopy);

// Добавляем на экран
addChild(shape1);
addChild(shape2);
shape2.x += 100;
```

В результате вы увидите два зеленых квадрата, первый нарисован нами (метод drawRect), а вторая копия воспроизведена автоматически.

MerlinTwi 24-10-2006, 12:46

В класс TextField (flash.text.TextField (http://livedocs.macromedia.com/flex/2/langref/flash/text/TextField.html)) добавлен новый метод appendText (flash.text.TextField.appendText() (http://livedocs.macromedia.com/flex/2/langref/flash/text/TextField.html#appendText())), который добавляет новый текст в конец существующего в текстовом поле (обычный текст, не html). Этот метод аналог TextField.text +=

newText, но работает гораздо быстрее.
```
var my_tf = new TextField();
my_tf.text = "Hello";
my_tf.appendText(" world!"); // my_tf.text == "Hello world!"
```

MerlinTwi 25-10-2006, 10:46

В ActionScript 3 оставлена директива include для вставки кода из другого файла, только изменился синтаксис, теперь не нужно писать символ #
```
include "scripts/myscript.as"
```
Храните подключаемые внешние файлы вне папки class path чтобы они не интерпретировались как классы.

MerlinTwi 25-10-2006, 11:00

В ActionScript 1 и 2 можно было объявить одну переменную дважды в одной области видимости и это не вызывало ошибки. Например:
```
// ActionScript 1 и 2
var i:Number = 1;
var i:String = 2;
```
В ActionScript 3 так нельзя. Вы можете только один раз объявить переменную в одной области видимости. Так же нельзя изменить тип переменной.
```
// ActionScript 3
var i:Number = 1;
var i:String = 2; // ERROR: duplicate definition or undefined property
```
В ситуации, где переменная используется в разных блоках for или if, вы должны объявить ее до блока:
```
// Это НЕВЕРНО для ActionScript 3
// (но работает в AS1 и AS2)
if (my_btn.enabled == true) {
var returnValue:Number = 1;
}else{
var returnValue:Number = 0;
}
if (provideAsString == true) {
var returnValue:String = "1";
}else{
var returnValue:String = "0";
}
```
Правильно для AS3 в данной ситуации будет так:
```
// ПРАВИЛЬНО для ActionScript 3
var returnValueNum:Number;
var returnValueStr:String;
if (my_btn.enabled == true) {
returnValueNum = 1;
}else{
returnValueNum = 0;
}
if (provideAsString == true) {
returnValueStr = "1";
}else{
returnValueStr = "0";
}
```
Если же неприемлемо использование двух переменных и обязательно должна быть одна, которая должна принимать как числовое значение так и строку, нужно использовать универсальный тип переменных *
```
// Также ПРАВИЛЬНО для ActionScript 3
var returnValue:*;
if (my_btn.enabled == true) {
returnValue = 1;
}else{
returnValue = 0;
}
if (provideAsString == true) {
returnValue = "1";
}else{
returnValue = "0";
}
```

MerlinTwi 25-10-2006, 11:39

В ActionScript 1 и 2, в случае если несколько мувиклипов, обрабатывающих события от мышки, частично или полностью перекрываются, то события от мышки (onPress, onRollOver …) получал только верхний мувиклип. Для того чтобы событие от мышки мог получить нижестоящий мувиклип, нужно или удалить верхний или удалить у него все обработчики событий мышки, тогда мувиклип станет «прозрачен» для событий от мышки.
В ActionScript 3 ситуация примерно аналогичная, но проверка на то, прозрачен мувиклип для событий от мышки или нет, лежит не на наличие обработчиков этих событий, а на свойстве mouseEnabled (flash.display.InteractiveObject.mouseEnabled

(http://livedocs.macromedia.com/flex/2/langref/flash/display/InteractiveObject.html#mouseEnabled)).
Если mouseEnabled=true (так по умолчанию), то мувиклип перехватывает и обрабатывает события от мышки. Если же mouseEnabled=false, то он становится прозрачен для мышки и события смогут получать и мувиклипы лежащие под ним.

В примере ниже, рисуются два частично перекрывающихся круга. Внизу "circle 1" и над ним "circle 2". При клике мышкой по любому кругу вызывается обработчик события "click", который показывает по какому кругу кликнули и переключается свойство mouseEnabled для верхнего "circle 2". Таким образом, если кликать мышкой в область, где круги пересекаются, то сначала событие перехватывает верхний "circle 2", а на второй клик реагирует уже нижний "circle 1", т.к. у вернего круга mouseEnabled=false;

```
function createCircle(name:String = ""):Sprite {
var circle:Sprite = new Sprite();
circle.name = name;
circle.graphics.lineStyle(0);
circle.graphics.beginFill(0xFF8080);
circle.graphics.drawCircle(50, 50, 50);
return circle;
}

var circle1:Sprite = createCircle("circle 1");
var circle2:Sprite = createCircle("circle 2");
circle2.x += 25;

addChild(circle1);
addChild(circle2);

circle1.addEventListener(MouseEvent.MOUSE_DOWN, click);
circle2.addEventListener(MouseEvent.MOUSE_DOWN, click);

function click(e) {
trace(e.target.name + " clicked."); // trace name

// Выключаем/включаем реакцию circle2 на события от мышки
circle2.mouseEnabled = !circle2.mouseEnabled;
}
```

MerlinTwi 26-10-2006, 11:53

Новое свойство mouseChildren (flash.display.DisplayObjectContainer.mouseChildre n (http://livedocs.macromedia.com/flex/2/langref/flash/display/DisplayObjectContainer.html#mouseChildren)) позволяет включать и выключать возможность реагировать на события от мышки для всех вложенных мувиклипов.
Присвоить mouseChildren=false аналогично, если присвоить mouseEnabled=false для всех вложенных мувиклипов, после этого никто из вложенных мувиклипов не будет реагировать на события от мышки.
Это может быть полезно, если вы создаете свою кнопку наследуясь от MovieClip или Sprite со вложенными мувиклипами. По умолчанию в обработчике события такой кнопки Event.target будет не сама кнопка, а вложенный в нее мувиклип. Пример:

```
// Сама кнопка
var spriteButton:Sprite = new Sprite();
spriteButton.name = "spriteButton";
spriteButton.mouseChildren = true;

// Рисунок внутри кнопки
var spriteGraphics:Sprite = new Sprite();
spriteGraphics.name = "spriteGraphics";
spriteGraphics.graphics.beginFill(0x4080A0);
spriteGraphics.graphics.drawCircle(50, 50, 25);

// добавляем на экран
spriteButton.addChild(spriteGraphics);
addChild(spriteButton);

// слушаем событие click
spriteButton.addEventListener(MouseEvent.CLICK, click);
function click(evt:MouseEvent):void {
trace(evt.target.name);
}
Если mouseChildren = true; (по умолчанию)
//output
spriteButton

Если mouseChildren = false;
// output
spriteGraphics
```

MerlinTwi 26-10-2006, 12:25

Любой интерактивный объект (flash.display.InteractiveObject (http://livedocs.macromedia.com/flex/2/langref/flash/display/InteractiveObject.html)) имеет две пары очень похожих событий rollOver/rollOut и mouseOver/mouseOut. Оба они реагируют на то когда мышка входит в область объекта или покидает ее. Различия в поведении этих событий возникают только тогда, когда есть вложенные мувиклипы. События rollOver и rollOut не обращают внимание на наличие вложенных мувиклипов, когда мышка войдет в область объекта сработает rollOver, и только когда мышка покинет объект сработает rollOut. События mouseOver и mouseOut будут срабатывать на перемещение мышки над вложенными мувиклипами. mouseOver сработает когда мышка войдет в область объекта, если при дальнейшем движении курсор мышки войдет в область вложенного мувиклипа сработает mouseOut, когда мышка покинет вложенный объект вновь сработает mouseOver, хотя при этих перемещениях курсор мыши фактически не покидал области объекта. События mouseOver и mouseOut будут вести себя аналогично rollOver и rollOut если установить mouseChildren=false;
Лучше всего посмотреть пример:

```
// Основная кнопка
var spriteButton:Sprite = new Sprite();
spriteButton.name = "spriteButton";
spriteButton.graphics.beginFill(0xFF0000);
spriteButton.graphics.drawCircle(50,50, 50);
//spriteButton.mouseChildren=false;

// Вложенный в кнопку Sprite
var spriteGraphics:Sprite = new Sprite();
spriteGraphics.name = "childrenSprite";
spriteGraphics.graphics.beginFill(0x00FF00);
spriteGraphics.graphics.drawCircle(50,50, 15);

// Добавляем на экран
spriteButton.addChild(spriteGraphics);
addChild(spriteButton);

// События
spriteButton.addEventListener(MouseEvent.ROLL_OVER , evnt);
spriteButton.addEventListener(MouseEvent.ROLL_OUT, evnt);
spriteButton.addEventListener(MouseEvent.MOUSE_OVE R, evnt);
spriteButton.addEventListener(MouseEvent.MOUSE_OUT , evnt);

function evnt(e:MouseEvent):void {
trace(e.type, "on", e.target.name);
}
```

При однократном проведении мышки через всю кнопку:

```
// output
rollOver on spriteButton
mouseOver on spriteButton
mouseOut on spriteButton
mouseOver on childrenSprite
mouseOut on childrenSprite
mouseOver on spriteButton
mouseOut on spriteButton
rollOut on spriteButton
```

Если убрать комментарий со строки spriteButton.mouseChildren=false; то различий в поведении событий roll и mouse не будет.

MerlinTwi 26-10-2006, 12:36

Метод contains (flash.display.DisplayObjectContainer.contains() (http://livedocs.macromedia.com/flex/2/langref/flash/display/DisplayObjectContainer.html#contains())) позволяет определить является ли указанный мувиклип вложенным в этот мувиклип (причем не важно в какой степени вложенности).
Пример:

```
var king:Sprite = new Sprite();
var queen:Sprite = new Sprite();
var jack:Sprite = new Sprite();
var joker:Sprite = new Sprite();

queen.addChild(jack);
king.addChild(queen);
addChild(king);
addChild(joker);

// В итоге получилась такая вложенность
// king
// queen
// jack
// joker
// проверяем:
```

```
trace(king.contains(queen)); // true
trace(king.contains(jack)); // true
trace(king.contains(joker)); // false

Если нужно проверить только первый уровень вложенности, то можно просто сравнить свойство parent.
trace(queen.parent == king); // true
trace(jack.parent == king); // false
```

MerlinTwi 27-10-2006, 11:32

Когда имеете дело с отображаемыми объектами (display objects) в ActionScript 3 важно помнить, что они могут существовать даже если не добавлены на экран (display list) и при этом продолжают получать и обрабатывать события на которые подписаны (очевидно, кроме событий от мышки). К примеру, событие enterFrame будет продолжать вызываться, и после удаления объекта с экрана. Это существенное отличие от ActionScript 1 и 2, т.к. там удаление мувиклипа с экрана означало и удаление всех событий.
Если вы хотите, чтобы событие типа enterFrame не вызывалось, когда объект удален с экрана, вы должны самостоятельно удалять подобные события. Сделать это просто, достаточно прописать обработчики событий Event.ADDED (вызывается когда объект добавлен к другому отображаемому объекту) и Event.REMOVED (вызывается когда объект удаляется от родителя).

```
var sprite:Sprite = new Sprite();
sprite.addEventListener(Event.ADDED, addEnterFrame);
sprite.addEventListener(Event.REMOVED, removeEnterFrame);

// Добавляем/удаляем обработчик события enter frame
function addEnterFrame(evt:Event):void {
trace("added");
sprite.addEventListener(Event.ENTER_FRAME, enterFrame);
}
function removeEnterFrame(evt:Event):void {
trace("removed");
sprite.removeEventListener(Event.ENTER_FRAME, enterFrame);
}
function enterFrame(evt:Event):void {
trace("Time: " + getTimer());
}

// Добавляем/удаляем объект по клику мышки
stage.addEventListener(MouseEvent.CLICK, addRemove);
function addRemove(evt:Event):void {
if (this.contains(sprite)) {
this.removeChild(sprite);
}else{
this.addChild(sprite);
}
}

// output
added
Time: 813
Time: 927
removed
added
Time: 2509
Time: 2597
removed
```

MerlinTwi 27-10-2006, 11:54

В предыдущем примере показано как определить добавление и удаление объекта от другого отображаемого объекта. Но если с экрана будет удален наш родительский объект, то события ADDED/ REMOVED не сработают. Как же отследить реальное добавление или удаление с экрана? Можно воспользоваться классом StageDetection (com.senocular.events.StageDetection (http://www.senocular.com/flash/actionscript.php?file=ActionScript_3.0/com/senocular/events/StageDetection.as))

```
package com.senocular.events {

import flash.display.DisplayObject;
import flash.events.Event;
import flash.events.EventDispatcher;
import flash.utils.Dictionary;

/**
* StageDetection
* Позволяет DisplayObject реагировать на добавление и удаление со stage
*
* Использовать в конструкторе
* var stageDetect:StageDetection = new StageDetection(this);
```

```
* stageDetect.addEventListener(StageDetection.ADDED_ TO_STAGE, addedToStage);
* stageDetect.addEventListener(StageDetection.REMOVE D_FROM_STAGE, removedFromStage);
*/
public class StageDetection extends EventDispatcher {

protected var target:DisplayObject;
protected var parents:Dictionary;
protected var detect:String;

public static const ADDED_TO_STAGE:String = "addedToStage";
public static const REMOVED_FROM_STAGE:String = "removedFromStage";

/**
* Конструктор
*/
public function StageDetection(targetObject:DisplayObject) {
target = targetObject;

// Определяем за чем нужно следить, за добавлением или удалением
detect = (target.stage == null) ? Event.ADDED : Event.REMOVED;

// Обновляем слушателей у родителей
updateListeners();
}

/**
* Обновляет слушателей событий ADDED и REMOVED
*/
protected function updateListeners(newDetect:String = null):void {

// Очищаем старые listeners у родителей
if (parents) {
for (var key:Object in parents) {
key.removeEventListener(detect, stageCheck, false);
}
}

// Запомним на какие события подписываемся
if (newDetect) detect = newDetect;

// Прописывает слушателей этого события у родителей
parents = new Dictionary(true);
var parent:DisplayObject = target;
while (parent) {
parent.addEventListener(detect, stageCheck, false, 0, true);
parents[parent] = true;
parent = parent.parent;
}
}


/**
* В обработчике ADDED и REMOVED проверяем наличие stage
*/
protected function stageCheck(evt:Event):void {

// Реагируем только на оригинальный источник события
if (evt.target != evt.currentTarget) return;

// evt.type или ADDED или REMOVED
switch(evt.type) {

// Кого-то из родителей добавили к другому объекту
case Event.ADDED:

// stage доступен, значит нас добавили на экран
if (target.stage != null) {

// Рассылаем событие и обновляем listeners
dispatchEvent(new Event(ADDED_TO_STAGE));
updateListeners(Event.REMOVED);

// stage недоступен, значит просто добавили к другому объекту
} else {

// Список родителей изменился, значит обновим listeners
updateListeners();
```

```
}
break;

// Родителя удалили со stage
case Event.REMOVED:

// Рассылаем событие и обновляем listeners
dispatchEvent(new Event(REMOVED_FROM_STAGE));
updateListeners(Event.ADDED);
break;
}
}
}
}
```

MerlinTwi: На данный момент этот класс стоит рассматривать только в ознакомительных целях, как можно красиво реализовать решение поставленной задачи. Практической пользы нет, т.к. с очередным обновлением Flash плеера были добавлены события ADDED_TO_STAGE и REMOVED_FROM_STAGE позволяющие DisplayObject 'у проверять доступность stage. Событие ADDED_TO_STAGE отправляется DisplayObject 'у когда он один или дерево объектов с ним добавляются на сцену, REMOVED_FROM_STAGE при удалении.

MerlinTwi 27-10-2006, 12:21

В ActionScript 3 для отображаемых объектов добавлены три фазы распространения событий:

Захват, capturing
Обработка, targeting
Подъем, bubbling

1. захват (capturing)
Событие начинается со stage и спускается вниз, вызывая обработчики связанные с этим событием у всех объектов (если же обработчика не существует, ничего не происходит).

```
event
+---------|----+
| parent | |
| +------V-+ |
| |child | |
| | | |
| +--------+ |
+--------------+
```

2. обработка (targeting)
На стадии обработки вызывается обработчик события объекта которому собственно оно и предназначалось.

```
+--------------+
| parent |
| +------ -+ |
| |child | |
| |event X | |
| +--------+ |
+--------------+
```

3. подъем (bubbling)
Перебираются все объекты, начиная с "виновника" события и до stage.

```
+---------^----+
| parent | |
| +------|-+ |
| |child | | |
| | event| | |
| +--------+ |
+--------------+
```

По умолчанию addEventListener подписывает обработчик события только на две фазы 2 (обработка) и 3 (подъем). Это позволяет вашему обработчику события реагировать на событие у себя а так же на это событие и у детей. Если вы хотите реагировать на стадию захвата (1), необходимо в addEventListener указать третьим параметром true.

```
target.addEventListener("event", listener, true);
```

При этом обработчик события будет реагировать только на стадию захвата. Если вы хотите подписать обработчик события на все стадии нужно вызвать addEventListener дважды:

```
target.addEventListener("event", listener, true); // захват
target.addEventListener("event", listener, false); // обработка и подъем
```

MerlinTwi 30-10-2006, 11:31

Обработчик события в ActionScript 3 может определить фазу события (см. предыдущий совет о фазах событий) по

свойству eventPhase, которое может принимать три значения:

EventPhase.CAPTURING_PHASE
EventPhase.AT_TARGET
EventPhase.BUBBLING_PHASE

Пример:

```
var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x4080A0);
circle.graphics.drawCircle(50, 50, 25);
addChild(circle);

circle.addEventListener(MouseEvent.CLICK, clickCircle);
circle.addEventListener(MouseEvent.CLICK, clickCircle, true);
stage.addEventListener(MouseEvent.CLICK, clickStage);
stage.addEventListener(MouseEvent.CLICK, clickStage, true);

function clickCircle(evt:MouseEvent):void {
trace("clickCircle: " + getPhaseName(evt.eventPhase));
}
function clickStage(evt:MouseEvent):void {
trace("clickStage: " + getPhaseName(evt.eventPhase));
}

function getPhaseName(phase:int):String {
switch(phase) {
case EventPhase.CAPTURING_PHASE:
return "Capturing Phase";
case EventPhase.AT_TARGET:
return "At Target Phase";
case EventPhase.BUBBLING_PHASE:
return "Bubbling Phase";
}
return "Error: No Phase Detected";
}
```

Кликнув по кругу:

```
//output
clickStage: Capturing Phase
clickCircle: At Target Phase
clickStage: Bubbling Phase
```

Кликнув вне круга:

```
//output
clickStage: At Target Phase
```

MerlinTwi 30-10-2006, 11:46

Если нужно пресечь дальнейшее распространение события можно использовать методы stopPropagation() (flash.events.Event.stopPropagation() (http://livedocs.macromedia.com/flex/2/langref/flash/events/Event.html#stopPropagation())) или stopImmediatePropagation() (flash.events.Event.stopImmediatePropagation() (http://livedocs.macromedia.com/flex/2/langref/flash/events/Event.html#stopImmediatePropagation())).
stopPropagation – предотвращает дальнейшее распространение события в стадии захвата (capturing) или подъема (bubbling)
stopImmediatePropagation – блокирует вообще все, в том числе и фазу обработка (targeting).
Смотрим пример:

```
var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x4080A0);
circle.graphics.drawCircle(50, 50, 25);
addChild(circle);

circle.addEventListener(MouseEvent.CLICK, clickCircle1);
circle.addEventListener(MouseEvent.CLICK, clickCircle2);
stage.addEventListener(MouseEvent.CLICK, clickStage);

function clickCircle1(evt:MouseEvent):void {
// evt.stopPropagation();
// evt.stopImmediatePropagation();
trace("clickCircle1");
}
function clickCircle2(evt:MouseEvent):void {
trace("clickCircle2");
}
function clickStage(evt:MouseEvent):void {
trace("clickStage");
}
```

При клике мышкой по кругу (circle), без использования прерывания распространения события:
//output
clickCircle1
clickCircle2
clickStage

Срабатывают два обработчика событий на клик мышки "clickCircle1","clickCircle2" фаза обработка (targeting) и обработчик "clickStage" на фазе подъема (bubbling).

Если убрать комментарий с evt.stopPropagation();
//output с использованием stopPropagation
clickCircle1
clickCircle2

В обработчике "clickCircle1" вызывается stopPropagation, которое предотвращает дальнейшее распространение события по фазам захвата (capturing) и подъема (bubbling), поэтому далее нормально вызывается обработчик "clickCircle2" в фазе обработка (targeting), но не вызывается "clickStage", поскольку он уже на фазе подъема (bubbling).

Если убрать комментарий с evt.stopImmediatePropagation();
// output с использованием stopImmediatePropagation
clickCircle1

В этом случае не будет вызван даже обработчик "clickCircle2" в фазе обработка (targeting).

MerlinTwi 30-10-2006, 12:21

В ActionScript 1 и 2 имелись события onMouseDown и onMouseUp, которые вызывались во всех объектах не зависимо от того, где находилась мышка. В ActionScript 3 такие события распространяются только на те объекты, над которыми находится мышка в момент клика. К примеру, если ваш мувиклип обрабатывает события MOUSE_DOWN и MOUSE_UP, а пользователь нажал кнопку мыши над мувиклипом, отвел мышку в сторону и отпустил кнопку, то событие MOUSE_UP ваш мувиклип не получит. Для обработки события глобально в ActionScript 3 нужно добавлять обработчик события к Stage. Поскольку именно к stage добавляются все отображаемые объекты, то через него проходят все события.
Одна маленькая, но важная деталь. Что если вы добавили обработчик события к stage, а какой-то мувиклип использовал предотвращение распространения события (см. предыдущий совет), в этом случае ваш обработчик события вызван не будет. Значит нужно добавлять обработчик события к stage на фазу захвата (capture). Но если пользователь кликнет по пустому месту, где нет никаких объектов, то целью события в этом случае будет stage и ваш обработчик события на фазу захвата (capture) вызван не будет, поскольку stage в данном случае участвует только в фазе обработка (targeting). Получается, для того чтобы поймать событие глобально в любом случае, нужно добавить два обработчика события к stage на фазе захвата (capture) и на фазе обработка (targeting). Поскольку нельзя поставить обработчик события только на фазу обработка (targeting), то нужно будет фазу подъема (bubbling) игнорировать в обработчике.
Пример:
```
var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x4080A0);
circle.graphics.drawCircle(50, 50, 25);
addChild(circle);

// Используем stage для глобальной обработки mouse up
stage.addEventListener(MouseEvent.MOUSE_UP, mouseUp);
stage.addEventListener(MouseEvent.MOUSE_UP, mouseUp, true);
function mouseUp(evt:MouseEvent):void {
// обрабатываем только фазы capturing и target
if (evt.eventPhase == EventPhase.BUBBLING_PHASE) return;

trace("global mouseUp");
}

// пусть мувиклип circle предотвращает распространение события mouse up
circle.addEventListener(MouseEvent.MOUSE_UP, mouseUpCircle);
function mouseUpCircle(evt:MouseEvent):void {
trace("mouseUpCircle");
evt.stopPropagation();
}
```

Даже не смотря на то, что circle вызывает stopPropagation для предотвращения распространения события, в этом случае наш глобальный обработчик будет вызван на фазе захвата (capturing)
//клик вне мувиклипа
global mouseUp

// клик по мувиклипу circle
global mouseUp
mouseUpCircle

MerlinTwi 31-10-2006, 14:44

В ActionScript 3 нет события onReleaseOutside, но его можно реализовать самостоятельно. Поскольку события mouseDown и mouseUp привязаны к объекту, то узнать о том, что кнопка мышки была отпущена за пределами объекта можно только слушая событие mouseUp глобально (см. предыдущий совет). Единственное, нужно добавить проверку отпустили кнопку мыши над тем же объектом (тогда это будет просто mouseUp) или над другим (тогда это будет onReleaseOutside).
Пример:

```
// в этой переменной запоминаем ссылку на объект по которому кликнули
var clicked:DisplayObject;

var circle:Sprite = new Sprite();
circle.graphics.beginFill(0x4080A0);
circle.graphics.drawCircle(50, 50, 25);
addChild(circle);

function mouseDown(evt:MouseEvent):void {
trace("mouseDown");
clicked = circle;
}
function mouseUp(evt:MouseEvent):void {
trace("mouseUp");
}
function mouseUpOutside(evt:MouseEvent):void {
trace("mouseUpOutside");
}

// обработчик события на клик мышкой по объекту circle
circle.addEventListener(MouseEvent.MOUSE_DOWN, mouseDown);
// обработчики событий для глобального mouseUp (подробнее см. предыдущий совет)
stage.addEventListener(MouseEvent.MOUSE_UP, captureMouseUp);
stage.addEventListener(MouseEvent.MOUSE_UP, captureMouseUp, true);

function captureMouseUp(evt:MouseEvent):void {
if (evt.eventPhase == EventPhase.BUBBLING_PHASE) return;

// убедимся, что кликнули по нужному нам объекту
if (clicked == circle) {
clicked = null; // очищаем ссылку для следующих кликов

var target:Sprite = evt.target as Sprite;
// отпустили кнопку над circle
if (target == circle) {
mouseUp(evt);
}else{
// отпустили кнопку мыши вне circle
mouseUpOutside(evt);
}
}
}
```

В этом примере мы реализовали аналог событий onPress, onRelease, и onReleaseOutside из AS1 и AS2.
Нажать и отпустить кнопку мыши над кругом (объект circle):

```
//output
mouseDown
mouseUp
```

Нажать кнопку мыши над кругом, а отпустить в другом месте:

```
//output
mouseDown
mouseUpOutside
```

Примечание: вместо того чтобы использовать переменную clicked, можно было бы в обработчике клика мышки mouseDown прописать слушателей на глобальный mouseUp, а в обработчике captureMouseUp удалить их.

MerlinTwi 01-11-2006, 12:48

В ActionScript 3 можно назначить свой класс (Document class) для главного timeline, который фактически будет являться root для всех отображаемых объектов. Назначить Document class можно в панели Property Inspector для документа (когда ничего не выделено) или же в диалоге ActionScript 3 Settings который расположен: File > Publish Settings > Flash [закладка] > Settings... [кнопка для ActionScript 3]. Просто впишите в поле "Document class" название вашего класса.
Document Class обязательно должен наследоваться от Sprite (flash.display.Sprite (http://livedocs.macromedia.com/flex/2/langref/flash/display/Sprite.html)) или его потомков. Если будете использовать главную timeline, то document class лучше наследовать от MovieClip (flash.display.MovieClip (http://livedocs.macromedia.com/flex/2/langref/flash/display/MovieClip.html)) т.к. MovieClip поддерживает фреймы.
Document Class обязательно должен быть публичным (public).
Пример:

```
package {

import flash.events.Event;
import flash.display.MovieClip;

public class CustomDocument extends MovieClip {

public function CustomDocument() {
addEventListener(Event.ADDED, checkChildren);
checkChildren(new Event("initialize"));
}

private function checkChildren(evt:Event):void {
// Позволяем добавить только один объект на root
if (numChildren > 1) {
throw (new Error("This movie can have only one child instance"));
}
}
}
}
```
Этот класс запрещает разместить на root более одного объекта. Обратите внимание, что метод checkChildren нужно отдельно вызвать в конструкторе, т.к. объекты могли быть добавлены на timeline до того, как создан обработчик события Event.ADDED (размещение в IDE на timeline до публикации).

MerlinTwi 01-11-2006, 12:56

Stage (сцена) – это фактически первый контейнер в который загружаются все остальные отображаемые объекты, включая root. Stage может быть только один, а вот root может быть сколько угодно, по одному на каждую загруженную внешнюю флешку.
Все отображаемые объекты (flash.display.DisplayObject (http://livedocs.macromedia.com/flex/2/langref/flash/display/DisplayObject.html)) имеют свойства stage и root. До тех пор, пока объект не добавлен на сцену (stage) эти свойства равны null. Свойство stage, когда доступно, всегда ссылается на объект stage. А вот свойство root в разный случаях содержит разные ссылки:

Для stage, свойство root всегда ссылается на stage
Для всех объектов, расположенных на главной timeline, свойство root ссылается на главную timeline
Если объект добавлен прямо на сцену (stage) из любой timeline, свойство root для него и его детей будет ссылаться на stage
Для объектов расположенных в подгруженном внешнем SWF-файле, свойство root ссылается на главную timeline этого SWF-файла
Для загруженных картинок, свойство root ссылается на экземпляр класса Bitmap, куда была загружена картинка

Помните, что только после добавления на сцену, или к любому другому объекту уже добавленному на сцену, есть доступ к свойствам stage и root. Это серьезное отличие от ActionScript 1 и 2 и на то есть две причины:

В AS1 и 2 мувиклипы в момент своего создания уже были добавлены на сцену. В AS3 это не так. Фактически только document class (см. предыдущий совет) изначально расположен на stage (во Flash 9 еще и объекты, расположенные на timeline в IDE до публикации).
В AS1 и 2 все скрипты и классы были определены внутри timeline или мувиклипов, как результат отовсюду был доступ к свойству _root.В AS3 классы не привязаны к timeline и могут не иметь доступа к root или stage.

Это сильно ограничивает наш доступ к stage или root в скриптах. К примеру, не отображаемый класс в AS3 вообще не может получить доступ к stage или root, только если получит ссылку извне.
```
package {

import flash.display.Stage;

public class CustomObject {

private var stage:Stage;

public function CustomObject(stageRef:Stage) {

// Ссылку на stage нужно передавать как параметр конструктора
stage = stageRef;
}
}
}
```
Отображаемые объекты в момент создания в конструкторе не имеют доступа к stage и root (если только они не были расположены на timeline до компиляции)
```
package {

import flash.display.Sprite;
import flash.events.Event;

public class CustomDisplayObject extends Sprite {
```

```
public function CustomObject() {

// stage недоступен до тех пор,
// пока объект не будет добавлен на сцену
trace(stage); // null
}
}
}
```
Это создает определенные трудности. Нужно добавлять дополнительный обработчик события добавления на сцену (ADDED_TO_STAGE), из которого уже вызывать функцию init для совершения необходимых действий.
Другой вариант решения создать статичное свойство в каком-либо классе, в которое прописать ссылку на root или stage и уже к этому свойству обращаться из любого места. Лучше всего для этого подойдет document class, т.к. он изначально имеет доступ к stage и сам является root.
Пример:
```
package {

import flash.display.DisplayObject;
import flash.display.MovieClip;
import flash.display.Stage;

public class TopLevel extends MovieClip {

public static var stage:Stage;
public static var root:DisplayObject;

public function TopLevel() {
TopLevel.stage = this.stage;
TopLevel.root = this;
}
}
}
```
Доступ к stage из неотображаемого класса:
```
package {

public class RandomClass {

public function RandomClass() {
trace(TopLevel.stage); // [object Stage]
}
}
}
```
В идеале стоит писать код так, чтобы не было необходимости в создании классов подобных TopLevel и жесткой привязке к нему. Особенно если в разработке участвуют несколько программистов. Определять доступность stage лучше по событию.

MerlinTwi 01-11-2006, 13:00

Для уточнения используемого пространства имен (см. совет 54) вместе с оператором «::» (name qualifier operator (http://livedocs.macromedia.com/flex/2/langref/operators.html#name_qualifier)), можно использовать use namespace (use namespace directive (http://livedocs.macromedia.com/flex/2/langref/statements.html#use_namespace)). use namespace определяет пространство имен на весь блок кода (package, class, method). use namespace нельзя дважды использовать в одном блоке, но можно обратиться к другому пространству имен через оператор «::»
Пример:
```
package {

public namespace company = "http://www.example.com/company";
public namespace individual = "http://www.example.com/individual";

public class UsingNameSpaces {

use namespace individual;

company var value:int = 10;
individual var value:int = 2;

public function UsingNameSpaces(){
showValue(); // traces individual::2
company::showValue(); // traces company::2;
}

company function showValue() {
trace("company::" + value);
}

individual function showValue() {
trace("individual::" + value);
}
```

}
}
В этом легко запутаться, хоть все и логично. Здесь директивой use namespace было назначено по умолчанию для всего класса использовать пространство имен individual. Соответственно первый вызов метода showValue без указания пространства имен это аналог individual::showValue. Во втором вызове явно указано пространство имен company, но поскольку в самом методе showValue используется свойство value без указания пространства имен, то по умолчанию берется то, что назначено use namespace, т.е. individual и метод company::showValue выводит 2 (individual::value) а не 10 (company::value).

MerlinTwi 02-11-2006, 17:30

Flash Player 9 с ActionScript 3 имеет встроенные средства для отображения и предупреждения об ошибках в коде во время выполнения. Если у вас установлена отладочная версия плеера (debug versions player), то появляется диалог с подробным описанием возникшей ошибки.
Возможные типы ошибок можно найти здесь (Top Level (http://livedocs.macromedia.com/flex/2/langref/package-detail.html)) или в пакете (flash.errors (http://livedocs.macromedia.com/flex/2/langref/flash/errors/package-detail.html)). Вы можете создавать свои типы ошибок, расширяя класс Error, и создавать ее с помощью ключевого слова throw (throw statement (http://livedocs.macromedia.com/flex/2/langref/statements.html#throw)). Эта возможность была и в ActionScript 1 и 2, но раньше плеер на ошибку никак не реагировал.
Пример ошибки, которая возникает во время выполнения:
var value:* = new Array();
trace(value.getChildAt(0));
Попробуйте запустить SWF с этим кодом в браузере. Здесь генерится ошибка TypeError (Top Level TypeError (http://livedocs.macromedia.com/flex/2/langref/TypeError.html)), поскольку value типа Array и не имеет метода getChildAt.
Дополнительная информация об ошибках выполнения.
(http://livedocs.macromedia.com/flex/2/langref/runtimeErrors.html)

MerlinTwi 02-11-2006, 17:32

В ActionScript 3 больше нет класса Color, взамен можно использовать класс ColorTransform (flash.geom.ColorTransform (http://livedocs.macromedia.com/flex/2/langref/flash/geom/ColorTransform.html)). По сути это тот же класс ColorTransform что был и в Flash Player 8, только свойство rgb было переименовано в color.

```
// создаем красный квадрат
// рисуем черный
var square:Shape = new Shape();
square.graphics.beginFill(0x000000);
square.graphics.drawRect(0, 0, 100, 100);
// и делаем цветокоррекцию с помощью ColorTransform
var colorTransform:ColorTransform = square.transform.colorTransform;
colorTransform.color = 0xFF0000;
square.transform.colorTransform = colorTransform;

addChild(square);
```

MerlinTwi 02-11-2006, 17:45

Для перехвата ошибок вы можете использовать (try..catch..finally statement (http://livedocs.macromedia.com/flex/2/langref/statements.html#try..catch..finally)).

```
try {
// statements
} catch (error:Error) {
// statements
} finally {
// statements
}
```

Первый блок внутри try – это место где ожидается возникновение ошибки. Далее идет блок catch который будет выполнится если в блоке try произойдет ошибка. Блоков catch может быть несколько с указанием на какой тип ошибки реагировать. И в конце идет блок finally, который выполняется после try и catch и выполняется всегда, даже если ошибка не была поймана в catch и дальнейшее выполнение кода будет прервано, или же было вызвано return, блок finally все равно выполняется.
Пример:

```
try {
var value:* = new Array();
trace(value.getChildAt(0));

} catch(error:IOErrorEvent) {
trace("IOErrorEvent catch: " + error);

} catch(error:TypeError) {
trace("TypeError catch: " + error);

} catch(error:Error) {
```

```
trace("Error catch: " + error);

} finally {
trace("Finally!");
}

trace("Continuing with script...");
```

Результат выполнения:
TypeError catch: TypeError: Error #1006: getChildAt is not a function.
Finally!
Continuing with script...

В соответствии с типом ошибки выполнился блок catch(error:TypeError), и выполнение скрипта продолжилось.
Что произойдет, если не будет блока catch соответствующего типу возникшей ошибки:

```
try {
var value:* = new Array();
trace(value.getChildAt(0));

} catch(error:IOErrorEvent) {
trace("IOErrorEvent catch: " + error);

} finally {
trace("Finally!");
}

trace("Continuing with script...");
```

Результат выполнения:
Finally!
TypeError: Error #1006: getChildAt is not a function.
at Timeline0_a24dfc5f2aa9864b9d4de27c9fa097db/::frame1()

Ошибка не была поймана, выполнился только блок finally и дальнейшее выполнение скрипта было прервано.

MerlinTwi 02-11-2006, 18:03

Иногда ошибка может произойти намного позже того, как была вызвана функция ее спровоцировавшая. Например, загрузка внешнего контента с помощью (flash.net.URLLoader (http://livedocs.macromedia.com/flex/2/langref/flash/net/URLLoader.html)). После вызова URLLoader.load с неправильным URL, плеер продолжит выполнение дальнейшего кода, а непосредственно процесс загрузки начнется позже, и только тогда возникнет ошибка. В таких случаях использование блока try..catch..finally не поможет, т.к. в самом коде в момент выполнения ошибки нет.
Пример:

```
var loader:URLLoader = new URLLoader();
loader.load(new URLRequest("Invalid XML URL"));
trace("Continuing with script...");
```

Результат выполнения:
Continuing with script...
Error #2044: Unhandled ioError:. text=Error #2032: Stream Error. URL: file:///C|/DOCUME%7E1/senocular/LOCALS%7E1/Temp/Invalid XML URL
at Timeline0_5650dafd68b564789d938267b772078/::frame1()

Ошибка возникла после выполнения trace("Continuing with script...").
В подобных случаях можно просто использовать обработчики событий.

```
var loader:URLLoader = new URLLoader();

loader.addEventListener(IOErrorEvent.IO_ERROR, catchIOError);
function catchIOError(event:IOErrorEvent){
trace("Error caught: "+event.type);
}

loader.load(new URLRequest("Invalid XML URL"));
trace("Continuing with script...");
```

Результат:
Continuing with script...
Error caught: ioError

MerlinTwi 02-11-2006, 18:58

ActionScript 3 поддерживает широкий диапазон типов переменных, включая несколько отсутствующих в предыдущих версиях ActionScript. Основные типы для AS3:

Простые:

Boolean

int
null
Number
String
uint
undefined

Сложные:

Object
Array
Date
Error
Function
RegExp
XML
XMLList

Существуют также дополнительные типы, которые относятся к их классам; такие как : Matrix (flash.geom.Matrix), Shape (flash.display.Shape), URLRequest (flash.net.URLRequest), и т.д.

Примечание:

Специальный тип Void изменился в AS3 на слово в нижнем регистре (void, а не Void)
Новый тип * используется для обозначения любого типа данных. Рекомендуется использовать вместо опущенной информации о типе для ваших переменных.
var anything:*;
Тип XML - не то же самое, что и тип XML в ActionScript 1 и 2. Старый XML тип (object) теперь определен как XMLObject. XML теперь обозначает новый E4X-based XML объект.
int и uint - новые простые типы данных для целых (числа без дробной части) и unsigned integer (неотрицательные числа без дробной части). Могут быть полезны для значений, не подразумевающих дробные значения, как, например, итераторы цикла. В большинстве случаев использование типа данных int вместо Number обеспечит небольшое увеличение производительности, а uint лучше использовать только когда это необходимо, например для значений цвета.

MerlinTwi 02-11-2006, 19:05

ActionScript 3 пополнилась коллекция "отображаемых объектов", которые могут быть видимы на экране и добавлены в "display list".
Отображаемые объекты AS3 :

AVM1Movie
Bitmap
Loader
MorphShape*
MovieClip
Shape
SimpleButton
Sprite
StaticText*
TextField
Video

*Для ссылки на объекты существующие в timeline; вы не можете создавать их с помощью AS.

AVM1Movie представляет собой flash-ролик, созданный в ActionScript 1 или 2. Эти ролики используют ActionScript Virtual Maching 1, тогда как AS3 ролики используют AVM2. AVM2 ролики могут проигрывать ролики AVM1, но не могут взаимодействовать с ними (с их ActionScript) используя AS3.

Bitmaps - объекты bitmap. Вы можете задавать им изображение с помощью BitmapData объектов или это могут быть bitmap-картинки из файлов.

Объекты Loader - отображаемые объекты, которые загружают в себя внешнее содержимое. Это могут быть изображения или другие SWF-приложения.

MorphShapes - это shape tweens (анимации фигур), созданные на timeline. Хотя вы не можете создать их в ActionScript, вы можете получить доступ к уже существующим на timeline, используя ActionScript и они будут иметь тип MorphShape.

MovieClips - это ролики, которые вы знаете и любите.

Shapes это разобранные ролики, которые по существу содержат только graphics object для рисования в нем с использованием vector drawing API. Использование Shapes вместо MovieClips или Sprites может помочь сэкономить память

Sprite объекты - по существу ролики без timelines. Это наиболее часто используемый тип в AS3, и обычно он расширяется при создании вашего собственного подкласса отображаемых объектов.

StaticText, как и MorphShapes, не могут быть созданы в ActionScript, ссылаясь вместо этого на объекты static text, созданные на Flash timeline.

Объекты TextField включают dynamic и input text.

Объекты Video представляют Flash video.

MerlinTwi 02-11-2006, 19:07

Директива import в ActionScript 3 немного отличается от import в ActionScript 2. В AS2 import использовался только для сокращения имен классов, определенных в пакетах; она не была обязательной для использования класса. Вместо использования import, вы могли просто использовать полное имя класса, например:
```
// ActionScript 2
var myPoint:flash.geom.Point = new flash.geom.Point(0,0);
```
В ActionScript 3, директива import обязательна для доступа к классам в их пакетах, даже если вы ссылаетесь на класс по его полному имени. Вы все еще можете использовать полное имя (или только проимпортированное имя класса), но использование import необходимо
```
// ActionScript 3
import flash.geom.Point;
var myPoint:flash.geom.Point = new flash.geom.Point(0,0);
```

Как и в AS2, в AS3 вы можете также использовать символ шаблона (*) для импорта всех классов в пакете.
```
import flash.geom.*;
```

MerlinTwi 02-11-2006, 19:10

ActionScript позволяет заменить назначенный объекту тип на другой совместимый при необходимости. Это называется приведением (casting). Оба ActionScript 2 и 3 поддерживают приведение с использованием синтаксиса type(object). Например, если объект вашего класса был назначен переменной как object, вы можете переопределить типизацию для этого объекта приведением его к типу вашего класса, дав таким образом возможность Flash знать какие именно методы и свойства доступны у этого объекта
```
var obj:Object = getMyCustomObject();
var customObj:MyClass = MyClass(obj);
```
ActionScript 3 вводит новый оператор для приведения типов - оператор as. Оператор as заменяет приведение типа с использованием type(object) синтаксиса в ActionScript 2 на синтаксис object as type.
```
var obj:Object = getMyCustomObject();
var customObj:MyClass = obj as MyClass;
```
Оператор as работает весьма схоже с приведением в ActionScript 2. Если преобразование не может быть выполнено, его результат null. В противном случае приводимый объект возвращается и ему назначается указанный тип.

ActionScript 3 все еще поддерживает type(object) приведение, но его поведение имеет теперь некоторые отличия. Вместо возврата null при неправильном приведении, генерируется TypeError. Сбой происходит когда вы пытаетесь привести объект к несовместимому типу, например при попытке привести объект к не связанному (not associated) или унаследованному типу.

Примечание: ActionScript также имеет глобальные функции преобразования для этих целей в стиле Class(object), имеющие приоритет над приведением type(object). Это такие функции, как String(), Number(), Array() и т.д. Они не выполняют приведение настолько же как фактическое преобразование одного объекта в другой (когда это возможно). Т.к. эти функции приоритетнее type(object), предпочтительнее использовать оператор as для приведения объектов к другому типу данных.

MerlinTwi 02-11-2006, 19:15

Ключевое слово delete во Flash используется для удаления определений переменных. Оно не удаляет объекты из памяти (это происходит за кулисами с помощью так называемого "сборщика мусора" /"Garbage Collector"/), а просто берет созданную вами переменную и избавляется от нее, делая ее недоступной и невидимой для итераторов (for..in циклы, и т.п.).

Внутренние механизмы Garbage Collector (GC для краткости), знают когда физически удалять объекты из памяти - когда больше нет переменных, ссылающихся на них. Так, например, если у вас есть две переменные A и B и они обе ссылаются на ObjectX, удаление переменной A не приведет к удалению сборщиком мусора ObjectX из памяти. Однако, если вы удалите обе переменные A и B, больше не будет ссылок на ObjectX и GC будет знать, что объект нуждается в удалении из памяти
```
var a:Object = new Object();
var b:Object = a; // reference same new Object();
delete a;
trace(b); // [object Object] - still exists in memory
delete b;
// GC will mark object for deletion from memory
```

Этот механизм работает практически одинаково для Flash 8 и Flash 9 (ActionScript 1, 2, и 3), однако в 8 были

сделаны некоторые изменения для оптимизации GC. (Примечание: GC очищает память не сразу.)

Хотя GC и новая виртуальная машина, управляющая им, на самом деле не сильно изменились в ActionScript 3, что изменилось, так это поведение ключевого слова delete. Теперь ключевое слово delete работает только для динамических свойств экземпляров класса, но не для объявленных членов класса (переменных и методов). В ActionScript 1 и 2 delete можно было использовать для всего. ActionScript 3 позволит только удалить динамические переменные и заблокирует прочее.

```
// ActionScript 2
class DeleteVarClass {

public var myVar:Number;

function DeleteVarClass() {
myVar = 1;
trace(myVar); // 1
delete myVar;
trace(myVar); // undefined
}
}
// ActionScript 3
package {
public class DeleteVarClass {

public var myVar:Number;

public function DeleteVarClass() {
myVar = 1;
trace(myVar); // 1
delete myVar;
trace(myVar); // 1
}
}
}
```

Поскольку myVar в примере выше был объявлен как свойство класса, он не может быть удален в ActionScript 3.

Поскольку вы не можете удалять члены класса в ActionScript 3, если вы хотите, чтобы переменная больше не ссылалась на объект или значение в памяти, вы можете установить значение переменной в null вместо ее удаления.

```
myVar = null;
```

MerlinTwi 02-11-2006, 19:20

Класс Dictionary (flash.utils.Dictionary (http://livedocs.macromedia.com/flex/2/langref/flash/utils/Dictionary.html)) - новое дополнение в ActionScript. Он представляет собой то же самое, что и базовые объекты Object, за исключением одной особенности: в объектах Dictionary в качестве ключей могут быть использованы не только строки, но и любые другие значения.

Object использует только строковые ключи. Если в качестве для ключа используется не строковое значение, оно интерпретируется в строку.
Пример:

```
var obj:Object = new Object();
obj["name"] = 1; // string key "name"
obj[1] = 2; // key 1 (converted to "1")
obj[new Object()] = 3; // key new Object() converted to "[object Object]"

for (var prop:String in obj) {
trace(prop); // traces: [object Object], 1, name
trace(obj[prop]); // traces: 3, 2, 1
}
```

Если вы попытаетесь использовать разные объекты в качестве ключей для Object, которые одинаково преобразуются в строку, в контейнере это будет один и тот же ключ, ссылающийся на единственное значение.

```
var a:Object = new Object();
var b:Object = new Object();

var obj:Object = new Object();
obj[a] = 1; // obj["[object Object]"] = 1;
obj[b] = 2; // obj["[object Object]"] = 2;

for (var prop:String in obj) {
trace(prop); // traces: [object Object]
trace(obj[prop]); // traces: 2
}
```

Класс Dictionary подобного ограничения не имеет. В качестве ключа вы можете использовать любой тип. Таким образом, если в примере выше использовать Dictionary, вы получите два различных ключа, по одному для каждого объекта.

```
import flash.utils.Dictionary;
```

```
var a:Object = new Object();
var b:Object = new Object();

var dict:Dictionary = new Dictionary();
dict[a] = 1; // dict[a] = 1;
dict[b] = 2; // dict[b] = 2;

for (var prop:* in dict) {
trace(prop); // traces: [object Object], [object Object]
trace(dict[prop]); // traces: 1, 2
}
```
И несмотря на то, что вы все же получили [object Object] при трассировке, это просто результат строкового преобразования в команде trace; это уникальный объектный ключ в экземляре Dictionary.

Обратите внимание, что prop здесь определена как *. Это важно, т.к. ключи в объекте dict могут быть любого типа. Если вы используете String как тип prop, объекты a и b при их выборке в цикле будут приводиться к строкам, помещая в prop "[object Object]" вместо действительных ссылок на a и b, которыми они должны быть для корректного доступа к значениям 1 и 2 из dict.

MerlinTwi 02-11-2006, 19:23

В ActionScript 3 введены метки (url=http://livedocs.macromedia.com/flex/2/langref/statements.html#label]labels [/url]), новые идентификаторы, которые можно привязывать к блокам циклов. Зачем вам может понадобиться идентифицировать блок цикла? Потому, что этот идентификатор можно использовать в командах break и continue. Представьте себе два цикла, один вложен в другой. Если в какой-то момент вы захотите прервать оба цикла, находясь во внутреннем, вы этого сделать не сможете. Команда break прервет только текущий блок. Обычно для этих целей используют переменную-флаг, которую устанавливают во внутреннем цикле, чтобы иметь возможность проверить во внешнем и, при необходимости, выйти также и из него. Пример:
```
var i:Number;
var j:Number;
var exit:Boolean = false;
for (i=0; i<10; i++) {
for (j=0; j<10; j++) {
if (i > 3 && j > 3) {
exit = true;
break;
}
}
if (exit) {
break;
}
}
```
Когда i больше 3 и j больше 3, используется break для выхода из текущего цикла, но это выход только из цикла по j. Для того, чтобы выйти и из цикла i, была использована переменная exit с условным оператором if в цикле по i.

Метки позволяют вам идентифицировать и прерывать указанный цикл (а также все вложенные в него). Формат метки label: statement
Пример:
```
var i:Number;
var j:Number;
mainLoop: for (i=0; i<10; i++) {
for (j=0; j<10; j++) {
if (i > 3 && j > 3) {
break mainLoop;
}
}
}
```
Задав для первого цикла метку mainLoop, мы получили возможность легко прервать вложенный цикл, используя break mainLoop; Это позволяет писать более понятный код и избавляет от необходимости использовать дополнительные переменные.

MerlinTwi 02-11-2006, 19:31

В ActionScript 3 можно определить что мышка ушла за пределы флеш-ролика, используя событие mouseLeave для stage. Это событие происходит каждый раз, когда мышка покидает пределы ролика. Нет события mouseEnter, но можно использовать mouseMove для определения того, что мышка вернулась к нам.
Простой пример использования нарисованного квадрата в качестве своего курсора.
```
package {
import flash.display.Sprite;
import flash.events.Event;
import flash.events.MouseEvent;
import flash.ui.Mouse;

public class Test extends Sprite {

private var cursor:Sprite = new Sprite();
```

```
public function Test() {
cursor.graphics.beginFill(0xFF);
cursor.graphics.drawRect(0, 0, 25, 25);
addChild(cursor);

stage.addEventListener(Event.MOUSE_LEAVE, cursorHide);
stage.addEventListener(MouseEvent.MOUSE_MOVE, cursorFollow);
Mouse.hide();
}

public function cursorHide(evt:Event):void {
cursor.visible = false;
}

public function cursorFollow(evt:MouseEvent):void {
if (!cursor.visible) cursor.visible = true;
cursor.x = stage.mouseX;
cursor.y = stage.mouseY;
evt.updateAfterEvent();
}
}
}
```
Когда мышка уходит за пределы флеш-ролика, наш курсор прячется. Когда мышка возвращается по событию mouseMove курсор снова отображается. Сделать такое в предыдущих версиях AS было нельзя.

MerlinTwi 02-11-2006, 19:37

При определении массива в ActionScript 3 с использованием квадратных скобок, теперь можно оставить запятую после последнего элемента и это не вызовет ошибки. Например:
```
var myList:Array = [
"The",
"quick",
"brown",
"fox",
];
```
Наличие запятой после "fox" в ActionScript 1 и 2 вызвало бы ошибку.
Это не работает с Array() или new Array(), только с [].
Это конечно мелочь, но при отладке может сэкономить время, если вам, к примеру, нужно временно убрать последний элемент массива, просто закомментировать его. Раньше еще приходилось и убирать запятую перед ним, теперь все проще.

MerlinTwi 02-11-2006, 19:48

Пакеты в ActionScript 3 немного отличаются от пакетов в ActionScript 2. Теперь пакет это не часть имени класса, а блок определяемый ключевым словом package, который содержит в себе определение класса. Структура такова:
```
package my.package.path {
class MyClass {
}
}
```
В AS2 подобная запись выглядела бы так:
```
// ActionScript 2:
class my.package.path.MyClass {
}
```

Фактически в AS3 все классы должны быть внутри пакета, если не нужно давать имя пакету, то следует записать так:
```
package {
class NotInAPackageClass {
}
}
```
Каждый пакет с определением класса или функции необходимо сохранить в файле имя которого совпадает с именем класса или функции, а расширение ".as". К примеру:
```
package com.kirupa.utils {
function StripString(str:String):void {
// ...
}
}
```
Нужно сохранить в файле: StripString.as в папке com/kirupa/utils

MerlinTwi 02-11-2006, 19:51

К сожалению, ActionScript 3 НЕ поддерживает абстрактные классы (экземпляры которых нельзя создать, а можно только унаследовать). Т.е. вы не можете создавать ваши собственные абстрактные классы. Однако, обратите внимание, что некоторые из внутренних классов ActionScript сами по себе являются абстрактными. Эти классы

включают:

DisplayObject
InteractiveObject
DisplayObjectContainer
Graphics

Как и с абстрактными классами вы не можете создавать их экземпляры с помощью ключевого слова new.
var myObj:InteractiveObject = new InteractiveObject(); // ERROR
Однако, в дополнение к этому, в ActionScript вы также не можете непосредственно расширять эти классы и создавать экземпляры таких подклассов

```
package {
import flash.display.DisplayObject;
public class MyDisplay extends DisplayObject{
public function MyDisplay (){
// ERROR
}
}
}
```

Если вы попытаетесь унаследовать один из них и создать экземпляр своего наследника, вы получите тот же Argument Error, что и при попытке создать экземпляр одного из этих классов непосредственно.

Вместо этого вам необходимо расширять те внутренние классы, которые уже являются наследниками этих классов. К примеру, если вы хотите расширить DisplayObject, вы можете вместо этого расширить Shape, легкий встроенный класс, унаследованный от DisplayObject.

MerlinTwi 02-11-2006, 19:54

Перекрытием (overriding) называется переопределение метода в классе, который в противном случае был бы унаследован. Новый метод будет использоваться вместо унаследованного (хотя унаследованный метод остается доступен с использованием super).

В ActionScript 3, когда вы перекрываете метод или свойство родительского класса, вы должны использовать аттрибут override. Он указывает на то, что создаваемый вами член класса будет перекрывать тот, который иначе был бы унаследован. Если вы не укажете override для метода, который уже существует в родительском классе, возникнет ошибка компиляции.

Пример:

```
package {
import flash.display.*;
class MySprite extends Sprite {

private var children:Array = new Array();

public function MySprite() {
}

public override function addChild(child:DisplayObject):DisplayObject {
children.push(child);
super.addChild(child);
return child;
}
}
}
```

Т.к. addChild уже существует в родительском классе Sprite, необходимо использовать override для успешного определения нового метода addChild, который добавляет кроме прочего переданный child в массив children.

Обратите внимание, что описание метода должно соответствовать перекрываемому

Перекрытие работает как с обычными методами класса, так и с getter/setter методами (свойствами), однако оно не будет работать ни с чем из нижеперечисленного:

Переменные
Константы
Статические методы
Методы, которые не унаследованы
Методы, реализующие интерфейс
Унаследованные методы, отмеченные как финальные (final) в родительском классе

Обратите также внимание, что перекрытие не требуется для методов, которые наследуются непосредственно от класса Object. Это:

hasOwnProperty
isPrototypeOf

propertyIsEnumerable
setPropertyIsEnumerable
toString
valueOf

Эти методы добавляются динамически и не являются частью действительного определения класса. Ключевое слово override используется только для методов, которые присутствуют в исходном описании класса.

Однако, если расширяется класс, в котором приведенные выше методы являются частью его описания, ключевое слово override необходимо. Например, если вы расширяете класс Object, вам не нужно использовать override для метода toString. Но если вы расширяете класс Sprite, вам потребуется перекрывать toString, т.к. класс Sprite имеет в своем описании собственный toString.

MerlinTwi 02-11-2006, 20:05

Для доступа к вложенным объектам (детям) в XML вы можете использовать оператор точка (.) (XML dot operator (http://livedocs.macromedia.com/flex/2/langref/operators.html#dot_(XML))). Пример:
var myXML:XML =
<foo>
<bar />
<bar />
<bar />
</foo>;

trace(myXML.bar.toXMLString());

Результат:
<bar/>
<bar/>
<bar/>

Это аналогично использованию метода elements (XML.elements() (http://livedocs.macromedia.com/flex/2/langref/XML.html#elements())).
trace(myXML.elements("bar").toXMLString());
Однако, есть еще похожий оператор две точки (..) (descendant accessor operator (http://livedocs.macromedia.com/flex/2/langref/operators.html#descendant_accessor)). Который работает аналогично, но возвращает и объекты более глубокого уровня вложенности. Пример:
var myXML:XML =
<note>
<replying-to>
<note>
<author>Julie</author>
<title>Reminder</title>
<body>Take out the trash</body>
</note>
</replying-to>
<author>Kevin</author>
<title>Re: Reminder</title>
<body>I will.</body>
</note>;

trace("Children:");
trace(myXML.author.toXMLString());
trace("Decendants:");
trace(myXML..author.toXMLString());
Результат:
Children:
<author>Kevin</author>
Decendants:
<author>Julie</author>
<author>Kevin</author>

Есть и специальный метод, который выполняет ту же задачу decendants (XML.descendants() (http://livedocs.macromedia.com/flex/2/langref/XML.html#descendants())).
trace(myXML.descendants("author").toXMLString());

MerlinTwi 02-11-2006, 20:09

В классе Array (Top level Array (http://livedocs.macromedia.com/flex/2/langref/Array.html)) добавлены новые методы indexOf и lastIndexOf.
AS3 function indexOf(searchElement:*, fromIndex:int = 0):int
AS3 function lastIndexOf(searchElement:*, fromIndex:int = 0x7fffffff):int

По сути они работают аналогично String.indexOf и String.lastIndexOf, возвращают позицию искомого элемента в массиве. Если объект не найден возвращается -1
var sprite:Sprite = new Sprite();

```
var object:Object = new Object();
var boolean:Boolean = true;
var number:Number = 10;

var array:Array = new Array(sprite, object, number);
trace(array.indexOf(sprite)); // 0
trace(array.indexOf(number)); // 2
trace(array.indexOf(boolean)); // -1
```

MerlinTwi 02-11-2006, 20:17

Для создания ссылок в текстовом поле, клик по которым можно программно отловить, теперь нужно использовать слово event (flash.text.TextField.event:link (http://livedocs.macromedia.com/flex/2/langref/flash/text/TextField.html#event:link)), а не asfunction как это было в ActionScript 1 и 2. Кроме того, event теперь не вызывает указанную функцию а создает событие TextEvent (flash.events.TextEvent (http://livedocs.macromedia.com/flex/2/langref/flash/events/TextEvent.html)) с типом TextEvent.LINK, а в свойство text записывается текст указанный в ссылке после слова event.
Пример:

```
var linkText:TextField = new TextField();
linkText.htmlText = 'Link: <a href="event:Link Clicked">Click</a>';
addChild(linkText);

linkText.addEventListener(TextEvent.LINK, linkEvent);

function linkEvent(event:TextEvent):void {
trace(event.text); // Link Clicked
}
```

MerlinTwi 08-11-2006, 15:32

Помимо создания динамических свойств и методов в Proxy (flash.utils.Proxy (http://livedocs.macromedia.com/flex/2/langref/flash/utils/Proxy.html)) есть возможность управлять порядком перечисления свойств в циклах for..in и for each..in. Для этого используются методы класса:

```
nextName(index:int):String
nextValue(index:int):*
nextNameIndex(index:int):int
```

Метод nextName возвращает название свойства по порядковому номеру для цикла for..in (аналогично nextValue для цикла for each..in). nextNameIndex вызывается перед началом каждой итерации в цикле и отвечает за две вещи:
1) предоставляет порядковый номер для nextName и nextValue (если возвращаемое значение >0), или
2) прерывает цикл (если возвращаемое значение равно нулю).
При начале цикла nextNameIndex получает порядковый номер равный нулю. Каждое последующее значение возвращаемое nextNameIndex отлично от нуля и передается методам nextName и nextValue. Если nextNameIndex возвращает ноль, то цикл заканчивается.
К примеру, proxy имеет 3 свойства (x,y,visible), попробуем их перечислить в цикле for..in

```
for (var prop:String in proxy) {
trace(prop);
}
```

Здесь будут вызываться методы nextNameIndex и nextName:

```
for (var prop:String in proxy) {
[ proxy.nextNameIndex(0) -> return 1 ]
[ proxy.nextName(1) -> return "x" ]
[ prop = "x" ]
trace(prop); // x
(end for block, repeat)
[ proxy.nextNameIndex(1) -> return 2 ]
[ proxy.nextName(2) -> return "y" ]
[ prop = "y" ]
trace(prop); // y
(end for block, repeat)
[ proxy.nextNameIndex(2) -> return 3 ]
[ proxy.nextName(3) -> return "visible" ]
[ prop = "visible" ]
trace(prop); // visible
[ proxy.nextNameIndex(3) -> return 0 ]
(0 index, break from for block)
}
```

Если использовать цикл for each..in, то nextValue будет вызываться вместо nextName.
Поскольку в циклах используется порядковый номер, то удобно хранить динамические свойства в массиве и в методах nextName или nextValue просто вернуть элемент массива index-1 (поскольку nextName и nextValue никогда не получают порядкового номера 0).
Создадим прокси-класс, который реализует пример выше. Помните, что методы proxy определены в пространстве

имен flash_proxy.

```
package {

import flash.utils.Proxy;
import flash.utils.flash_proxy;

public class ProxyEnum extends Proxy {

private var props:Array = ["x", "y", "visible"]; // массив свойств

// nextNameIndex вызывается при начале итерации в цикле
override flash_proxy function nextNameIndex (index:int):int {
if (index < props.length) {
// первый вызов 0, возвращаем 1 + index
// т.е. порядковый номер будет 1, затем 2, 3
return index + 1;
} else {
// все свойства перечислены для прерывания цикла
// возвращаем 0
return 0;
}
}

// nextName вызывается после nextNameIndex и порядковый номер начинается с 1
override flash_proxy function nextName(index:int):String {
// возвращаем элемент массива index – 1
return props[index - 1];
}
}
}
```

Пример использования:

```
var proxy:ProxyEnum = new ProxyEnum();
for (var prop in proxy) {
trace(prop);
}
/* output:
x
y
visible
*/
```

Для полной картины необходимо еще переопределить методы nextValue и getProperty. Но для них нужны возвращаемые значения, пусть класс принимает в конструкторе ссылку на какой-то отображаемый объект, его свойства и будем возвращать.

```
package {

import flash.display.DisplayObject;
import flash.utils.Proxy;
import flash.utils.flash_proxy;

public class ProxyEnum extends Proxy {

private var props:Array = ["x", "y", "visible"]; // массив свойств
private var _target:DisplayObject;

function ProxyEnum(target:DisplayObject) {
_target = target;
}

// nextNameIndex вызывается при начале итерации в цикле
override flash_proxy function nextNameIndex (index:int):int {
if (index < props.length) {
// первый вызов 0, возвращаем 1 + index
// т.е. порядковый номер будет 1, затем 2, 3
return index + 1;
} else {
// все свойства перечислены для прерывания цикла
// возвращаем 0
return 0;
}
}

// nextName вызывается после nextNameIndex и порядковый номер начинается с 1
override flash_proxy function nextName(index:int):String {
// возвращаем элемент массива index – 1
return props[index - 1];
}
```

```
// nextValue вызывается после nextNameIndex в циклах for each..in
override flash_proxy function nextValue(index:int):* {
// получаем название свойства из массива
var prop:String = props[index - 1];
// и возвращаем его значение
return _target[prop];
}

// возвращает значение свойства по его имени
override flash_proxy function getProperty(name:*):* {
return _target[name];
}
}
}
```

Пример использования:

```
var proxy:ProxyEnum = new ProxyEnum(my_mc);

for (var prop:String in proxy) { // nextName/nextNameIndex
trace(prop);
trace(proxy[prop]); // getProperty
}
/* output:
x
34
y
76
visible
true
*/

for each(var value:* in proxy) { // nextValue/nextNameIndex
trace(value);
}
/* output:
34
76
true
*/
```

MerlinTwi 09-11-2006, 11:36

MerlinTwi: Поскольку этот совет (Event Capturing and mouseEnabled (http://www.kirupa.com/forum/showthread.php?p=1961593#post1961593)) в оригинале практически дублирует (72. mouseEnabled и блокирование событий (http://flasher.ru/forum/showpost.php?p=581297&postcount=66)), видимо senocular где-то ошибся, то я сам расскажу еще про mouseEnabled.
К примеру, создаем класс текстовой кнопки, есть фоновый мувиклип, который растягивается под ширину текста и сверху на него накладывается текст (TextField). Если обработчики событий от мышки (MOUSE_OVER, MOUSE_OUT, CLICK) добавить к фоновому мувиклипу, то события будут срабатывать только на краях кнопки, где нет текста. Это происходит потому, что все интерактивные отображаемые объекты создаются по умолчанию с mouseEnabled=true, т.е. реагируют на события от мышки, а как было показано в предыдущих советах событие от мышки получает только самый верхний объект на сцене. Поэтому в таких случаях нужно не забывать запрещать текстовому полю перехватывать события от мышки:
mouseEnabled=false;
А вообще, при создании кнопки лучше сделать:
buttonMode=true;

MerlinTwi 09-11-2006, 11:51

Во Flash 9 добавлен новый пункт настройки ActionScript 3, называется "strict mode" (расположен: Edit > Preferences > ActionScript > ActionScript 3.0 settings... > Strict mode). Если strict mode включен, то компилятор более строго проверяет код, особенно соответствие типов, и не создает SWF даже при незначительных ошибках. К примеру, свойство "constructor" класса Object не определено в самом классе, а только в прототипе. Поэтому, если попытаться обратиться к этому свойству в не динамическом классе в strict mode, компилятор выдаст ошибку.
Пробуем:

```
var mySprite:Sprite = new Sprite();
trace(mySprite.constructor); Компиляция:
**Error** Scene 1, Layer 'Layer 1', Frame 1 : Line 2, Column 16 : [Compiler] Error #1119: Access of possibly undefined property constructor through a reference with static type flash.display:Sprite.
trace(mySprite.constructor);
ReferenceError: Error #1065: Variable Timeline0_9f132e9d986cc749b16415211316a5f0 is not defined.
```

В данном случае стоит сделать приведение типа к Object, который является динамическим, поэтому ошибки не будет:

```
var mySprite:Sprite = new Sprite();
trace(Object(mySprite).constructor); // [class Sprite]
```

MerlinTwi 09-11-2006, 11:55

В классе System (flash.system.System (http://livedocs.macromedia.com/flex/2/langref/flash/system/System.html)) есть новое интересное свойство totalMemory (flash.system.System.totalMemory (http://livedocs.macromedia.com/flex/2/langref/flash/system/System.html#totalMemory)), которое показывает сколько байт памяти использует Flash-плеер в данный момент. Например:
var o:Object = new Object();
trace(System.totalMemory); // 4960256
Или более «объемный» класс:
var o:MovieClip = new MovieClip();
trace(System.totalMemory); // 4964352

MerlinTwi 09-11-2006, 20:17

Работая с XML в ActionScript 3 вы в основном имеете дело с двумя объектами: XML (Top level XML (http://livedocs.macromedia.com/flex/2/langref/XML.html)) и XMLList (Top level XMLList (http://livedocs.macromedia.com/flex/2/langref/XMLList.html)). Существенная разница между ними в том, что XML – это один элемент XML-дерева, например узел (который может содержать произвольное количество вложенных узлов), а XMLList массив элементов (1 или более).

```
// XML:
<foo>
<foo />
<foo />
<foo />
</foo>


// XMLList:
<foo />
<foo />
<foo />
```

В первом примере есть один корневой узел и это будет представлено как XML объект. Во втором примере список узлов, это уже будет XMLList.
В отличие от старого XML объекта из предыдущих версий ActionScript (который теперь XMLDocument), новые XML и XMLList могут представлять из себя не только узлы XML дерева (nodes), но и другие значения, например атрибуты:

```
var myXML:XML = <foo>
<foo bar="bar1" />
<foo bar="bar2" />
<foo bar="bar3" />
</foo>;

trace(myXML.foo.@bar.toXMLString());
/* Output:
bar1
bar2
bar3
*/
trace(myXML.foo.@bar is XMLList); // true
```

Заметьте, что атрибуты всегда возвращаются как XMLList, даже если в итоге получается только один атрибут:
trace(myXML.foo[0].@bar is XMLList); // true
Для того чтобы получить XML объект, нужно из возвращаемого массива взять первый элемент:
trace(myXML.foo[0].@bar[0] is XML); // true
Аналогичная ситуация и с вложенными узлами XML:

```
var myXML:XML = <foo>
<foo />
</foo>;
trace(myXML.foo is XMLList); // true
```

MerlinTwi 09-11-2006, 20:35

В ActionScript 3 введен новый класс Timer (flash.utils.Timer (http://livedocs.macromedia.com/flex/2/langref/flash/utils/Timer.html)), который заменяет устаревшую функцию setInterval (flash.utils.setInterval() (http://livedocs.macromedia.com/flex/2/langref/flash/utils/package.html#setInterval())), хоть setInterval пока и оставлена, использовать ее не рекомендуется. В отличие от setInterval класс Timer рассылает события (flash.events.TimerEvent (http://livedocs.macromedia.com/flex/2/langref/flash/events/TimerEvent.html)), а значит на один таймер можно навесить сколько угодно обработчиков. Кроме того, Timer дает возможность контролировать сколько раз должен сработать таймер.
Пример:

```
var timer:Timer = new Timer(500, 10);
```

```
timer.addEventListener(TimerEvent.TIMER, notifier);
timer.addEventListener(TimerEvent.TIMER, stopper);
stage.addEventListener(MouseEvent.CLICK, continuer);

function notifier(event:TimerEvent):void {
trace(timer.currentCount);
}
function stopper(event:TimerEvent):void {
switch (timer.currentCount) {
case 5:
timer.stop();
break;
case timer.repeatCount:
timer.reset();
break;
}
}
function continuer(event:MouseEvent):void {
timer.start();
}

timer.start();
```

Здесь timer 10 раз подряд посылает событие TimerEvent.TIMER каждые 500 миллисекунд. На событие добавлены два обработчика, один выводит номер текущей итерации, другими словами, сколько раз уже сработал таймер (свойство currentCount), а другой останавливает таймер после 5-го срабатывания (метод stop), или сбрасывает счетчик итераций (метод reset), когда количество итераций достигает максимально заданного (свойство repeatCount). Кликом мышки по сцене можно заново запустить остановленный таймер (метод start).
Результат выполнения:

```
1
2
3
4
5
(пауза; клик для продолжения)
6
7
8
9
10
(пауза; клик для продолжения)
1
2
3
4
5
...
```

MerlinTwi 09-11-2006, 21:10

Код ActionScript выполняется во flash-плеере в специальной «виртуальной машине», для ActionScript 1 и 2 это была AVM1, для ActionScript 3 была разработана новая AVM2. Нет возможности напрямую взаимодействовать между программами выполняющимся в разных виртуальных машинах, поскольку они несовместимы. К примеру, если в мувик созданный на ActionScript 3 загрузить внешний мувик из ActionScript 1 или 2, то не удастся напрямую управлять им: запустить, остановить, вызвать функцию, считать значение свойства и т.п.
Но можно создать специальный канал для взаимодействия при помощи LocalConnection, используя:

LocalConnection AS2 (http://livedocs.macromedia.com/flash/8/main/00002338.html)
LocalConnection AS3 (flash.net.LocalConnection)
(http://livedocs.macromedia.com/flex/2/langref/flash/net/LocalConnection.html)

Пример:

```
// файл на ActionScript 2: AS2animation.fla
// на timeline размещен один мувиклип названный animation_mc

//Устанавливаем local connection для получения команд извне
var AVM_lc:LocalConnection = new LocalConnection();

// обработчик события stopAnimation
AVM_lc.stopAnimation = function(){
animation_mc.stop();
}

// слушаем события по каналу "AVM2toAVM1"
AVM_lc.connect("AVM2toAVM1");
```

```
// Файл на ActionScript 3: AS3Loader.fla

// local connection для взаимодействия с мувиклипом в AVM1
var AVM_lc:LocalConnection = new LocalConnection();

// загружаем внешнюю SWF AVM1
var loader:Loader = new Loader();
loader.load(new URLRequest("AS2animation.swf"));
addChild(loader);

// при клике мышкой по загруженному мувиклипу вызывается stopPlayback
loader.addEventListener(MouseEvent.CLICK, stopPlayback);

function stopPlayback(event:MouseEvent):void {
// вызывается stopAnimation по каналу "AVM2toAVM1"
AVM_lc.send("AVM2toAVM1", "stopAnimation");
}
```

Здесь AS3-мувик подгружает к себе AS2-мувик и размещает на сцене. Когда AS2-мувик подгружен и проигрывается пользователь может кликнуть по нему для прекращения проигрывания, при этом вызывается stopPlayback, который отсылает событие "stopAnimation" по local connection каналу "AVM2toAVM1". AS2-мувик получает событие и вызывается обработчик stopAnimation, который приказывает мувиклипу animation_mc остановиться.

MerlinTwi 09-11-2006, 21:36

В ActionScript 3 добавлена возможность работать с бинарными данными, используя класс ByteArray (flash.utils.ByteArray (http://livedocs.macromedia.com/flex/2/langref/flash/utils/ByteArray.html)). Класс ByteArray наследуется от массива Array (Top level Array (http://livedocs.macromedia.com/flex/2/langref/Array.html)) и может содержать в себе байты любой бинарной информации, аналогично тому, как она размещена в памяти компьютера. Работать с классом довольно просто, поэтому лучше посмотреть примеры, что можно с его помощью сделать:

ByteArray.org Эксперименты с ByteArray (http://www.bytearray.org/?cat=4)
Как проиграть pcm wave в ActionScript 3 (http://www.flashcodersbrighton.org/wordpress/?p=9)
PNG кодировщик на AS3 (http://www.kaourantin.net/2005/10/png-encoder-in-as3.html)
JPEG кодировщик на AS3 (http://www.kaourantin.net/2005/10/more-fun-with-image-formats-in-as3.html)
Клонирование объекта (http://www.richapps.de/?p=34)
Бесплатный ID3Stream и ID3Reader компонент для Flex 2 (http://blog.benstucki.net/?id=24)