Бесплатная электронная книга

УЧУСЬ ActionScript 3

Free unaffiliated eBook created from
Stack Overflow contributors.

#actionscrip

t-3

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: actionscript-3

It is an unofficial and free ActionScript 3 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ActionScript 3.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с ActionScript 3

замечания

ActionScript 3 - это язык программирования для сред Adobe Flash Player и Adobe AIR. Это объектно-ориентированный язык на основе ECMAScript, используемый для разработки собственных приложений на настольных (Windows / Mac) и мобильных (iOS / Android) устройствах.

Ресурсы обучения Adobe: http://www.adobe.com/devnet/actionscript/learning.html

История и более подробная информация: https://en.wikipedia.org/wiki/ActionScript

Онлайн-документация по классам и ссылке: http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html

Версии

Существует одна версия ActionScript 3 под названием «ActionScript 3.0»,

Версия Flash	Кодовое имя	Изменения и улучшения	Дата выхода
Flash Player 9.x	Зафод	Первый выпуск	2006-06-22
Flash Player 10.0	астрономический	введен тип `Vector.<T>` , шейдер Adobe Pixel Bender фильтрует в классе `flash.filters.ShaderFilter` и его аппаратную поддержку на нескольких процессорах.	2008-10-15
Flash Player 10.1	Арго	представил класс `flash.events.TouchEvent` для работы с мультитач-устройствами и другую поддержку мобильных устройств, таких как акселерометр.	2010-06-10
Flash Player 10.2	Пряный	представил класс `flash.media.StageVideo` и общую структуру для работы с воспроизведением сцены в AS3.	2011-02-08

Версия Flash	Кодовое имя	Изменения и улучшения	Дата выхода
Flash Player 11	Serrano	добавляет поддержку H.264 для потоковой передачи видео по объектам `NetStream` в обоих направлениях. Также он добавляет поддержку SSL / TLS для соединения Flash с классом `SecureSocket` .	2011-10-04
Flash Player 11.4	Брэннэн	представил класс `flash.system.Worker` и возможность делегировать асинхронную работу другим потокам на клиенте.	2012-08-10
Flash Player 11.8	Харрисон	удаленная аппаратная поддержка (компиляция JIT) для фильтров шейдеров Adobe Pixel Bender, значительно снижая производительность любого исполнения фильтра шейдеров PB.	2013-05-09

Examples

Обзор установки

ActionScript 3 можно использовать, установив Adobe AIR SDK или Apache Flex SDK или как часть продукта Adobe Animate CC *(ранее известного как Flash Professional)* .

Adobe Animate CC - это профессиональное программное решение, которое можно использовать для создания проектов AS3 с использованием визуальных инструментов - после установки, никаких дальнейших шагов для создания проектов AS3 не требуется.

AIR SDK и Flex SDK можно использовать с инструментами командной строки или с различными сторонними IDE.

В дополнение к Adobe Animate CC есть четыре других популярных IDE, способных работать с AS3. Эти IDE имеют свои собственные инструкции о том, как начать работу.

- Flash Builder (от Adobe - на основе Eclipse)
- IntelliJ IDEA (от Jetbrains)
- FlashDevelop
- FDT (плагин Eclipse)

Привет, мир

Пример класса документа, который выводит «Hello, World» на консоль отладки при создании экземпляра.

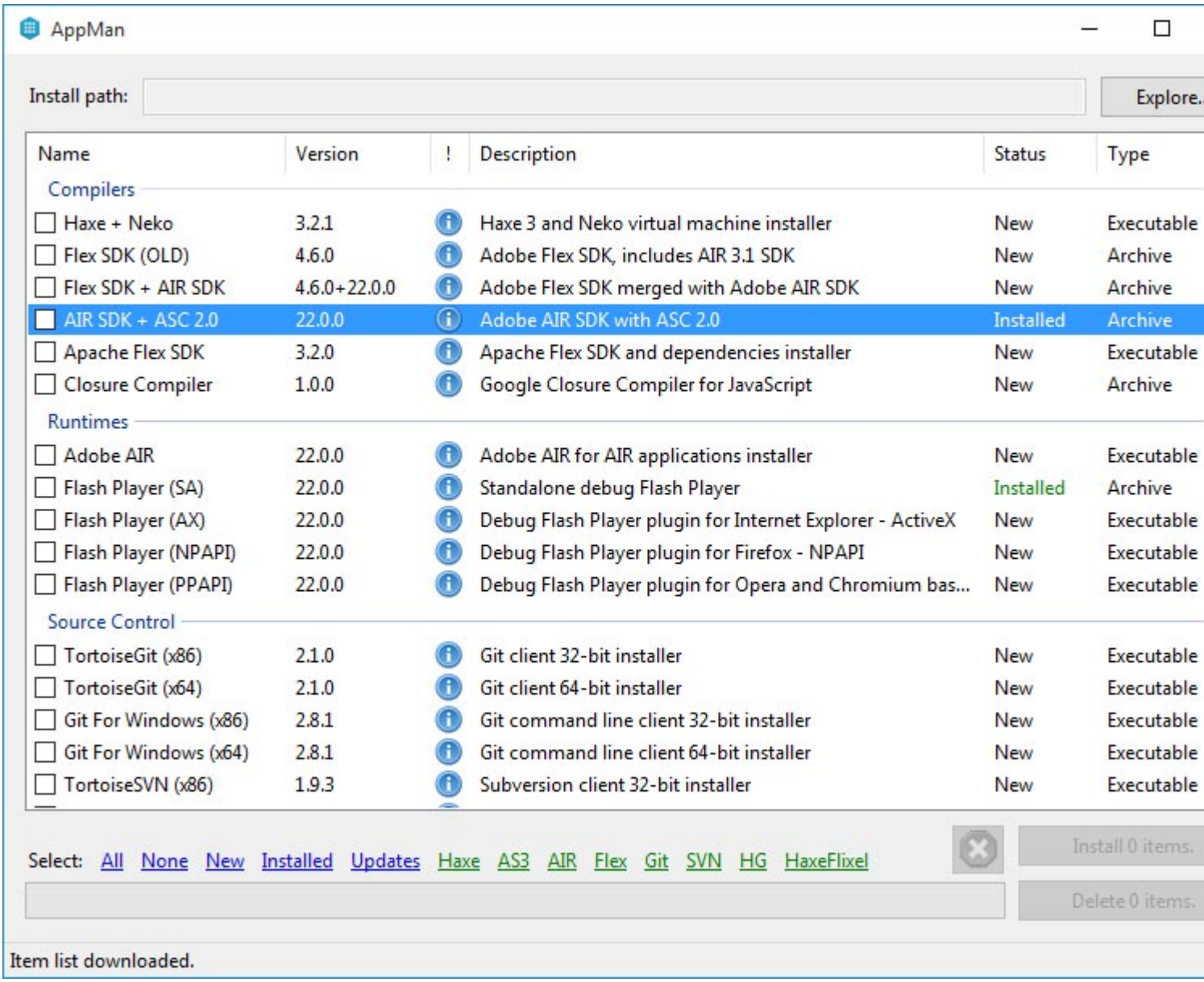
```
import flash.display.Sprite;

public class Main extends Sprite {

    public function Main() {
        super();

        trace("Hello, World");
    }

}
```

Разработка Flash-разработки

FlashDevelop - это многоплатформенная среда с открытым исходным кодом, созданная в 2005 году для разработчиков Flash. Без каких-либо затрат это очень популярный способ начать работу с AS3.

Чтобы установить FlashDevelop:

1. Загрузите установочный файл и запустите программу установки
2. По завершении установки запустите FlashDevelop. При первом запуске должно появиться окно `App Man` предлагающее вам выбрать, какие SDK и инструменты для установки.



Если AppMan не открывается автоматически или вы хотите добавить что-то позже, откройте его, выбрав «Установить программное обеспечение» в меню «Инструменты».

Проверьте элемент **AIR SDK + ACS 2.0** (в разделе «Компилятор») и элемент **Flash Player (SA)** в разделе «Runtimes» (плюс все, что вы хотели бы установить). Нажмите кнопку установки.

3. Как только SDK будет установлен, давайте протестируем это, создав проект hello world. Начните с создания нового проекта (из меню « *Проект*»)

4. Выберите **AIR AS3 Projector** из списка и укажите ему имя / местоположение.

5. В панели диспетчера проектов (выберите «Менеджер проектов» из меню просмотра, если он еще не виден), разверните папку **src** и откройте файл `Main.as`

6. В файле `Main.as` теперь вы можете создать первую `Main.as` программу, например Hello World

7. Запустите проект, щелкнув значок воспроизведения или нажав `F5` или `Ctrl+Enter` . Проект будет компилироваться, и по завершении должно появиться пустое окно (это ваше приложение). В окне вывода FlashDevelop вы должны увидеть слова **Hello World** .

Теперь вы готовы начать разработку приложений AS3 с помощью FlashDevelop!

Установка Apache Flex

от http://flex.apache.org/doc-getstarted.html

1. Загрузите программу установки SDK
2. Запустите программу установки SDK. Первый вопрос, который вам задан, - это каталог установки.

   - на Mac, используйте `/Applications/Adobe Flash Builder 4.7/sdks/4.14.0/`
   - на ПК используйте `C:\Program Files(x86)\Adobe Flash Builder 4.7\sdks\4.14.0`

   Вам нужно будет создать папки 4.14.0. Нажмите "Далее. Принять лицензии и установить SDK.

Специфические инструкции IDE для установки Apache Flex:

- Flash Builder
- IntelliJ IDEA
- FlashDevelop
- FDT

Создание Flex или Flash-проектов в командной строке с использованием mxmlc

Компилятор Flex (`mxmlc`) является одной из наиболее важных частей Flex SDK. Вы можете редактировать код AS3 в любом текстовом редакторе, который вам нравится. Создайте файл основного класса, который простирается от `DisplayObject` .

Вы можете запускать сборки в командной строке следующим образом:

```
mxmlc -source-path="." -default-size [width in pixels] [height in pixels] -default-frame-rate
[fps] -o "outputPath.swf" "mainClass.as"
```

Если вам нужно скомпилировать Flash-проект (в отличие от Flex), вы можете добавить ссылку на библиотеку Flash следующим образом (вам необходимо установить Adobe Animate IDE):

```
mxmlc -source-path="." -library-path+="/Applications/Adobe Animate CC 2015.2/Adobe Animate CC
2015.2.app/Contents/Common/Configuration/ActionScript 3.0/libs" -static-link-runtime-shared-
```

```
libraries=true -default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o
"outputPath.swf" "mainClass.as"
```

Или в Windows:

```
mxmlc -source-path="." -library-path+="C:\Program Files\Adobe\Adobe Animate CC
2015.2\Common\Configuration\ActionScript 3.0\libs" -static-link-runtime-shared-libraries=true
-default-size [width in pixels] [height in pixels] -default-frame-rate [fps] -o
"outputPath.swf" "mainClass.as"
```

Отображаемый пример «Hello World»

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

    public class TextHello extends Sprite {
        public function TextHello() {
            var tf:TextField = new TextField();
            tf.text = "Hello World!"
            tf.x = 50;
            tf.y = 40;
            addChild(tf);
        }
    }
}
```

Этот класс использует класс `TextField` для отображения текста.

Прочитайте Начало работы с ActionScript 3 онлайн: https://riptutorial.com/ru/actionscript-3/topic/1065/начало-работы-с-actionscript-3

глава 2: Генерация случайных величин

Examples

Случайное число от 0 до 1

```
Math.random();
```

создает равномерно распределенное случайное число между 0 (включительно) и 1 (эксклюзивным)

Пример вывода:

- +0,22282187035307288
- 0,3948539895936847
- 0,9987191134132445

Случайное число между значениями min и max

```
function randomMinMax(min:Number, max:Number):Number {
    return (min + (Math.random() * Math.abs(max - min)));
}
```

Эта функция вызывается путем передачи диапазона минимальных и максимальных значений.

Пример:

```
randomMinMax(1, 10);
```

Примеры выходов:

- +1,661770915146917
- 2,5521070677787066
- +9,436270965728909

Случайный угол, в градусах

```
function randomAngle():Number {
    return (Math.random() * 360);
}
```

Примеры выходов:

- +31,554428357630968

- +230,4078639484942
- +312,7964010089636

Случайное значение из массива

Предполагая, что у нас есть массив `myArray` :

```
var value:* = myArray[int(Math.random() * myArray.length)];
```

Обратите внимание, что мы используем `int` для передачи результата `Math.random()` в int, поскольку значения, такие как `2.4539543` , не будут действительным индексом массива.

Случайная точка внутри круга

Сначала определите радиус окружности и его центр:

```
var radius:Number = 100;
var center:Point = new Point(35, 70);
```

Затем генерируйте случайный угол в *радианах* из центра:

```
var angle:Number = Math.random() * Math.PI * 2;
```

Затем создайте эффективный радиус возвращенной точки, поэтому он будет находиться внутри заданного `radius` . Простой `Math.random()*radius` не будет выполняться, потому что при этом распределении точки создания будут `Math.random()*radius` в внутренний круг половины радиуса половины времени, но квадрат этого круга составляет четверть оригинала. Чтобы создать правильное распределение, функция должна быть такой:

```
var rad:Number=(Math.random()+Math.random())*radius; // yes, two separate calls to random
if (rad>radius) { rad=2*radius-rad; }
```

Эта функция дает значение, которое имеет свою вероятностную функцию, линейно увеличивающуюся от 0 до нуля до максимума при `radius` . Это происходит потому, что сумма случайных значений имеет функцию плотности вероятности, равную свертке всех индивидуальных функций плотности случайных величин. Это некоторые расширенные математические вычисления для человека среднего класса, но представлен вид GIF для рисования графика функции свертки двух функций плотности распределения в форме, которые объясняются как « боковые сигналы ». Оператор `if` сгибает результирующую функцию по ее максимуму, оставляя только пилообразный график.

Эта функция выбрана потому, что квадрат круглой полосы, расположенной между `radius=r` и `radius=r+dr` линейно возрастает с ростом `r` и очень малой постоянной `dr` так, что `dr*dr<<r` . Поэтому количество точек, сгенерированных вблизи центра, меньше, чем количество точек, созданных на краю круга, на тот же край, что и радиус центральной области меньше

радиуса всей окружности. Таким образом, точки равномерно распределены по всему кругу.

Теперь найдите свою случайную позицию:

```
var result:Point = new Point(
    center.x + Math.cos(angle) * rad,
    center.y + Math.sin(angle) * rad
);
```

Чтобы получить случайную точку на круге (на краю круга заданного радиуса), используйте `radius` вместо `rad` .

PS: Пример оказался перегруженным объяснением математики.

Случайный угол, в радианах

```
function randomAngleRadians():Number
{
    return Math.random() * Math.PI * 2;
}
```

Примеры выходов:

- +5,490068569213088
- 3,1984284719180205
- +4,581117863808207

Определение успеха операции «процентный шанс»

Если вам нужно катиться для `true` или `false` в ситуации «x% шанс», используйте:

```
function roll(chance:Number):Boolean {
    return Math.random() >= chance;
}
```

Используется как:

```
var success:Boolean = roll(0.5); // True 50% of the time.
var again:Boolean = roll(0.25); // True 25% of the time.
```

Создайте случайный цвет

Для того, чтобы получить *какой - либо* случайный цвет:

```
function randomColor():uint
{
    return Math.random() * 0xFFFFFF;
}
```

Если вам нужно больше контролировать красный, зеленый и синий каналы:

```
var r:uint = Math.random() * 0xFF;
var g:uint = Math.random() * 0xFF;
var b:uint = Math.random() * 0xFF;

var color:uint = r << 16 | g << 8 | b;
```

Здесь вы можете указать свой собственный диапазон для `r` , `g` и `b` (этот пример от 0 до 255).

Случайно петля через алфавит

```
var alphabet:Vector.<String> = new <String>[ "A", "B", "C", "D", "E", "F", "G",
                                             "H", "I", "J", "K", "L", "M", "N",
                                             "O", "P", "Q", "R", "S", "T", "U",
                                             "V", "W", "X", "Y", "Z" ];

while (alphabet.length > 0)
{
    var letter:String = alphabet.splice(int(Math.random() *
                                            alphabet.length), 1)[0];
    trace(letter);
}
```

Пример вывода:

V, M, F, E, D, U, S, L, X, K, Q, H, A, I, W, N, P, Y, J, C, T, O, R, G, Z

Рандомизировать массив

```
var alphabet:Array = [ "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N",
"O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z" ];

for (var i:int=alphabet.length-1;i>0;i--) {
    var j:int=Math.floor(Math.random()*(i+1));
    var swap=alphabet[j];
    alphabet[j]=alphabet[i];
    alphabet[i]=swap;
}
trace(alphabet);
```

Пример вывода

B, Z, D, R, U, N, O, M, I, L, C, J, P, H, W, S, Q, E, K, T, F, V, X, Y, G,

Этот метод известен как Shuffle массива Fisher-Yates .

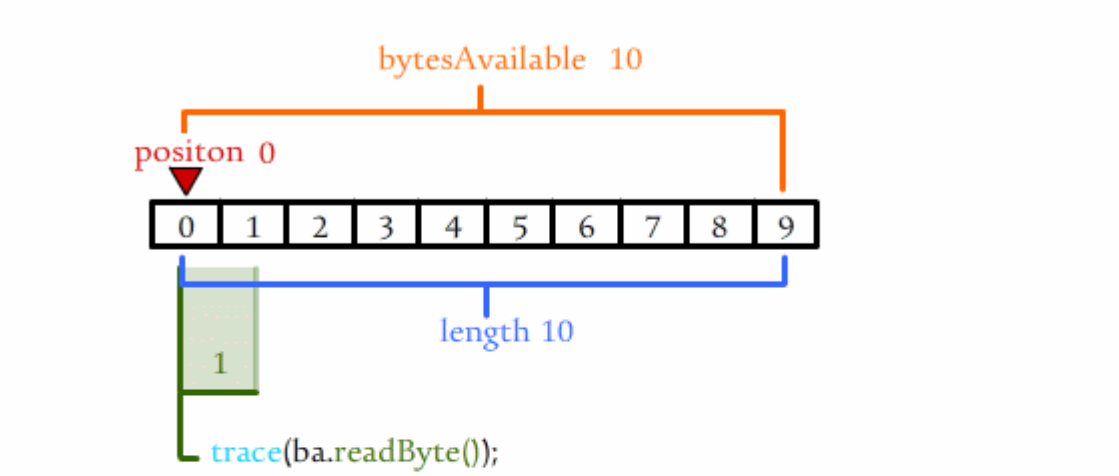
Прочитайте Генерация случайных величин онлайн: https://riptutorial.com/ru/actionscript-3/topic/1441/генерация-случайных-величин

глава 3: Двоичные данные

Examples

Чтение формы ByteArray через интерфейс IDataInput.

Анимация ниже показывает, что происходит, когда вы используете методы интерфейса IDataInput для доступа к форме данных ByteArray и другим классам, реализующим этот интерфейс.



Прочитайте Двоичные данные онлайн: https://riptutorial.com/ru/actionscript-3/topic/9464/двоичные-данные

глава 4: Жизненный цикл списка отображения

замечания

Рамочная анимация в Flash и AIR реализует следующий жизненный цикл:

- `Event.ENTER_FRAME` отправлено
- Выполняется код конструктора дочерних экранных объектов
- `Event.FRAME_CONSTRUCTED` отправляется
- Выполняются действия кадра в символе `MovieClip`
- Действия кадров в детских `MovieClip` символов выполняются
- `Event.EXIT_FRAME` отправлено
- `Event.RENDER` отправляется

Examples

Добавлено и удалено из жизненного цикла этапа

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

public class Viewport extends Sprite {

    /** Constructor */
    public function Viewport() {
        super();

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
    }

    /** Added to stage handler */
    protected function addedToStageHandler(event:Event):void {
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Listen for removed from stage event
        addEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

    /** Removed from stage handler */
    protected function removedFromStageHandler(event:Event):void {
        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);

        // Listen for added to stage event
        addEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);
```

```
    }

    /** Dispose */
    public function dispose():void {
        // Remove added to stage event listener
        removeEventListener(Event.ADDED_TO_STAGE, addedToStageHandler);

        // Remove removed from stage event listener
        removeEventListener(Event.REMOVED_FROM_STAGE, removedFromStageHandler);
    }

}
}
```

Прочитайте Жизненный цикл списка отображения онлайн:
https://riptutorial.com/ru/actionscript-3/topic/1877/жизненный-цикл-списка-отображения

глава 5: Загрузка внешних файлов

замечания

В некоторых случаях приложение не может управлять содержимым активов, загружаемых с внешнего ресурса (например, преобразовывать изображения). Я не могу точно помнить, что это такое, но я уверен, что это связано с загрузкой содержимого междоменного контента.

Examples

Загрузка внешних изображений / SWF с загрузчиком

1. Создайте объект Loader:

```
var loader:Loader = new Loader();   //import
```

2. Добавьте слушателей в загрузчик. Стандартные - полные и io / ошибки безопасности

```
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, loadComplete); //when the
loader is done loading, call this function
loader.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, loadIOError); //if the
file isn't found
loader.contentLoaderInfo.addEventListener(SecurityErrorEvent.SECURITY_ERROR,
loadSecurityError); //if the file isn't allowed to be loaded
```

3. Загрузите нужный файл:

```
 loader.load(new URLRequest("image.png"));
```

4. Создайте обработчики событий:

```
 function loadComplete(e:Event):void {
     //load complete
     //the loader is actually a display object itself, so you can just add it to the
display list
     addChild(loader)
     //or addChild(loader.content) to add the root content of what was loaded;
 }

 function loadIOError(e:IOErrorEvent):void {
     //the file failed to load,
 }

 function loadSecurityError(e:SecurityError):void {
     //the file wasn't allowed to load
 }
```

Загрузка с классом Loader является асинхронной. Это означает, что после вызова `loader.load` приложение будет продолжать работать при загрузке файла. Содержимое вашего загрузчика недоступно до тех пор, пока загрузчик не отправит событие `Event.COMPLETE` .

требуется импорт:

```
import flash.display.Loader;
import flash.events.Event;
import flash.events.IOErrorEvent;
import flash.events.SecurityErrorEvent;
import flash.net.URLRequest;
```

Загрузка текстового файла с помощью FileStream (только для среды выполнения AIR)

Простой пример того, как читать текстовый файл UTF синхронно.

```
import flash.filesystem.File;
import flash.filesystem.FileMode;
import flash.filesystem.FileStream;
```

```
//First, get a reference to the file you want to load
var myFile:File = File.documentsDirectory.resolvePath("lifestory.txt");

//Create a FileStream object
fileStream = new FileStream();

//open the file
fileStream.open(myFile, FileMode.READ);

//read the data and assign it to a local variable
var fileText:String = fileStream.readUTF();

//close the current filestream
fileStream.close();
```

Прочитайте Загрузка внешних файлов онлайн: https://riptutorial.com/ru/actionscript-3/topic/1694/загрузка-внешних-файлов

глава 6: Использование прокси-класса

Вступление

Во-первых, я должен сказать. Существует причина этого прокси , что, несмотря на кажущуюся полезность, не выделяется достаточно в Интернете.

Вы **не можете** использовать его для просмотра случайного свойства случайного класса / экземпляра. Вы можете использовать этот метод только путем подкласса класса **Proxy** .

Некоторые операции не требуют исключений, поэтому вам нужно полностью понять, что вы делаете, и почему вы это делаете, а ваш код **должен** быть абсолютно чистым и без ошибок.

Examples

Реализация

Другое дело о классе Proxy, и почему оно не так популярно, заключается в том, что довольно сложно понять проблему, которая нуждается в динамическом классе с контролируемым доступом к его динамическим свойствам и методам как наиболее подходящее решение. Каждый раз, когда я пытался использовать Proxy, я в конечном итоге прибегал к чему-то другому, более простому и контролируемому.

Однако не будем обескураживать. Мне нравится идея обращения к последним элементам массива с помощью индексов [-1], [-2] и т. Д. В **Python** . Возможно, это не большой подвиг, но приятно использовать этот, а не длинный и неуклюжий **someArray [someArray.length - 1]** . Давайте посмотрим, что мы можем с этим поделать.

```
package
{
    import flash.utils.Proxy;
    import flash.utils.flash_proxy;

    /**
     * Pyaray the Tentacled Whisperer of Impossible Secrets.
     */

    dynamic public class PyArray extends Proxy
    {
        private var data:Array;

        public function PyArray(...args:Array)
        {
            if (args.length == 0)
            {
                data = new Array;
            }
```

```
        else if ((args.length == 1) && (args[0] is Array))
        {
            data = args[0];
        }
        else
        {
            data = args;
        }
    }

    // This is a getter proxy to all the available Array
    // elements and properties and, sometimes, methods.
    override flash_proxy function getProperty(name:*):*
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return null;
            if (anIndex < 0) return null;

            return data[anIndex];
        }

        // Handle the existing public Array properties.
        if (data.hasOwnProperty(name)) return data[name];

        // Handle the Array methods addressed via ["member"] access.
        try
        {
            if (data[name] is Function) return data[name];
            else throw new Error;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to resolve property \"" + name + "\".");
        }

        return null;
    }

    // This will set either elements, or settable properties.
    override flash_proxy function setProperty(name:*, value:*):void
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            // In case the element index is out of range,
            // the PyArray will extend its data Array.
            // if (anIndex >= data.length) return;
            if (anIndex < 0) return;
```

```
            data[anIndex] = value;

            return;
        }

        // Handle the existing (or dynamic) public Array properties.
        try
        {
            data[name] = value;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to set property \"" + name + "\".");
        }

        return;
    }

    // This allows to delete PyArray elements with "delete" operator.
    override flash_proxy function deleteProperty(name:*):Boolean
    {
        var anIndex:int = name;

        // Handle the int indices of the Array elements.
        if (anIndex == name)
        {
            // Handle the -1, -2, etc indexing.
            if (anIndex < 0) anIndex += data.length;

            if (anIndex >= data.length) return false;
            if (anIndex < 0) return false;

            data.splice(anIndex, 1);

            return true;
        }

        // Handle the dynamic public Array properties.
        try
        {
            delete data[name];
            return true;
        }
        catch (fail:Error)
        {
            trace("[PyArray] is unable to delete the \"" + name + "\" property.");
        }

        return false;
    }

    // This proxies any attempt to call a method on PyArray directly to data Array, thus
    // all Array methods (including "toString" method called through trace) are available.
    override flash_proxy function callProperty(name:*, ...rest):*
    {
        try
        {
            return (data[name] as Function).apply(data, rest);
        }
        catch (fail:Error)
        {
```

```
                trace("[PyArray] is unable to resolve method \"" + name + "\".");
                return null;
            }
        }

        // This allows PyArray to handle for..in and for..each..in loops.
        // The initial call starts with zero, so we need to do this +1 -1 magic
        // in order for enumeration to work correctly. I'm not happy with this either.
        override flash_proxy function nextNameIndex(index:int):int
        {
            if (index >= data.length) return 0;
            else return index + 1;
        }

        // This method handles the for..in loop.
        override flash_proxy function nextName(index:int):String
        {
            return (index - 1).toString();
        }

        // This method handles the for..each..in loop.
        override flash_proxy function nextValue(index:int):*
        {
            return data[index - 1];
        }
    }
}
```

использование

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

    /**
     * Daemonette of Slaanesh.
     *
     * It is minor female demon, vaguely human-like, but with crab-like pincers instead of
hands.
     * She wears a rather indecent skimpy leather bikini, moves quickly and casts deadly
spells!
     */

    public class Slaanesh extends Sprite
    {
        public function Slaanesh()
        {
            // Lets initialize the PyArray.
            var PA:PyArray = new PyArray(1,2,3,4,5,4,3,2,1,"Foo");

            // Basic check: get the last element.
            trace(PA[-1]);
            // output:
            // Foo

            // This will map to the 0-based third element.
            trace(PA[2.0]);
            // output:
            // 3
```

```
// This should not get us anywhere.
trace(PA[2.1]);
// output:
// [PyArray] is unable to resolve property "2.1".
// null

// This should return the length of the data Array.
trace(PA["length"]);
// output:
// 10

// This should return the length of the data Array.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.length);
// output:
// 10

// This will map to indexOf method of data Array via getProperty method.
trace(PA["indexOf"]);
// output:
// function Function() {}

// This will map to indexOf method of data Array via getProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.indexOf);
// output:
// function Function() {}

// This is a try to access a non-existent property.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124);
// output:
// [PyArray] is unable to resolve property "P124".
// null

// This is a try to call a non-existent method via callProperty method.
// This will not compile unless PyArray class is marked "dynamic".
trace(PA.P124());
// output:
// [PyArray] is unable to resolve method "P124".
// null

// Basic check: calling a proxied method via callProperty method.
trace(PA.indexOf(5));
// output:
// 4

// An attempt to replace an Array method with a random value.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOf = 123;
// output:
// [PyArray] is unable to set property "indexOf".

// An attempt to assign a random value to a random property.
// It will succees because Array, as a dynamic class, allows so.
// This will not compile unless PyArray class is marked "dynamic".
PA.indexOfz = 123;
trace(PA.indexOfz);
// output:
// 123
```

```
            // An attempt to assign an Array element via negative indexing.
            // This trace works fine because toString method is also proxied.
            PA[-3] = "Hello";
            trace(PA);
            // output:
            // 1,2,3,4,5,4,3,Hello,1,Foo

            // An attempt to delete Array elements via normal and negative indexing.
            // This operation is mapped via deleteProperty method.
            delete PA[-4]; // deletes "3" before "Hello"
            delete PA[0];  // deletes "1" at the start.
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo

            // An attempt to delete a non-dynamic method reference.
            // There's no error output, AS3 must be handling this internally.
            delete PA.indexOf;
            trace(PA.indexOf);
            // output:
            // function Function() {}

            // An attempt to set an element out of index range.
            PA[10] = "123abc";
            trace(PA);
            // output:
            // 2,3,4,5,4,Hello,1,Foo,,,123abc

            var aText:String;

            // This is a test of for..in loop, Array elements are
            // enumerated via nextName and nextNameIndex methods.
            aText = ""

            for (var aKey:String in PA)
                aText += aKey + ":" + PA[aKey] + " ";

            trace(aText);
            // output:
            // 0:2 1:3 2:4 3:5 4:4 5:Hello 6:1 7:Foo 8:undefined 9:undefined 10:123abc

            // This is a test of for..each..in loop, Array elements are
            // enumerated via nextValue and nextNameIndex methods.
            aText = "";

            for each (var aValue:* in PA)
                aText += aValue + " ";

            trace(aText);
            // output:
            // 2 3 4 5 4 Hello 1 Foo undefined undefined 123abc
        }
    }
}
```

Прочитайте Использование прокси-класса онлайн: https://riptutorial.com/ru/actionscript-3/topic/10631/использование-прокси-класса

глава 7: Манипуляция и фильтрация растровых изображений

Вступление

В этом разделе вы можете немного узнать о том, как манипулировать **bitmapdata** и визуальной обработкой, работать с пикселями и начинать с фильтров эффектов.

Examples

Порог (монохромный) эффект

требуется:

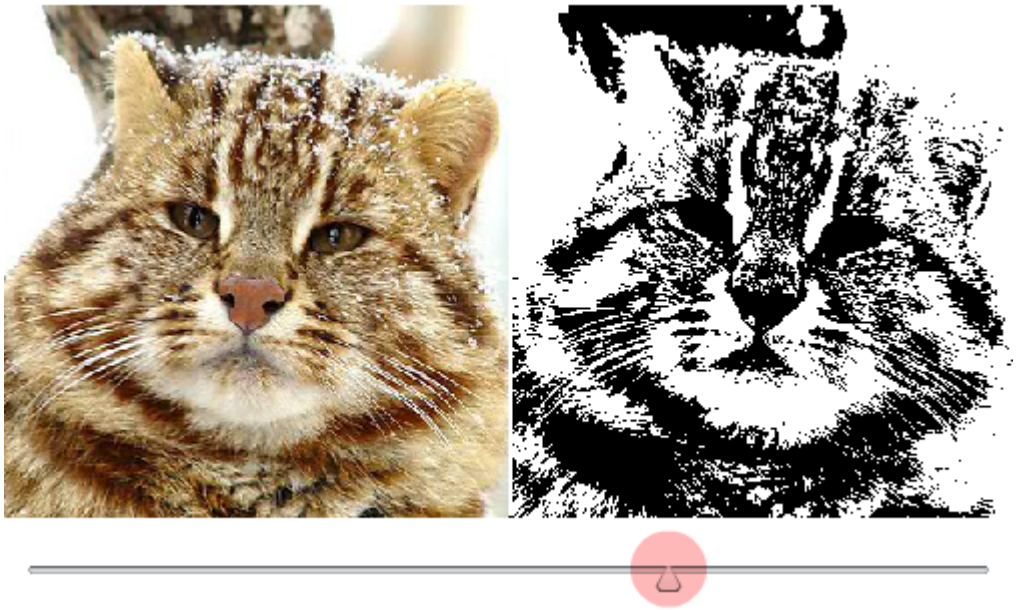
1. понимание растровых и растровых данных

что порог

> Эта настройка принимает все пиксели изображения и ... подталкивает их к чистому или чистому черному

что мы должны делать

> вот **демон-версия Live** этого примера с некоторыми дополнительными изменениями, такими как использование пользовательского интерфейса для изменения порогового уровня во время выполнения.



порог в действии скрипт 3 из as3 официальной документации

Проверяет значения пикселей в изображении с заданным порогом и устанавливает пиксели, которые передают тест новым значениям цвета. Используя метод threshold (), вы можете выделить и заменить диапазоны цветов в изображении и выполнить другие логические операции с пикселями изображения.

Логика тестирования порогового метода () заключается в следующем:

1. Если ((pixelValue & mask) операция (порог и маска)), то установите цвет пикселя;
2. В противном случае, если copySource == true, установите для пикселя соответствующее значение пикселя из sourceBitmap.

Я просто прокомментировал следующий код с точно указанными именами в качестве цитируемого описания.

```
import flash.display.BitmapData;
import flash.display.Bitmap;
import flash.geom.Rectangle;
import flash.geom.Point;

var bmd:BitmapData = new wildcat(); // instantied a bitmapdata from library a wildcat
var bmp:Bitmap = new Bitmap(bmd); // our display object to previewing bitmapdata on stage
addChild(bmp);
monochrome(bmd); // invoking threshold function

/**
    @param bmd, input bitmapData that should be monochromed
*/
function monochrome(bmd:BitmapData):void {
    var bmd_copy:BitmapData = bmd.clone(); // holding a pure copy of bitmapdata for
comparation steps
    // this is our "threshold" in description above, source pixels will be compared with this
```

```
value
    var level:uint = 0xFFAAAAAA; // #AARRGGBB. in this case i used RGB(170,170,170) with an
alpha of 1. its not median but standard
    // A rectangle that defines the area of the source image to use as input.
    var rect:Rectangle = new Rectangle(0,0,bmd.width,bmd.height);
    // The point within the destination image (the current BitmapData instance) that
corresponds to the upper-left corner of the source rectangle.
    var dest:Point = new Point();
    // thresholding will be done in two section
    // the last argument is "mask", which exists in both sides of comparation
    // first, modifying pixels which passed comparation and setting them all with "color"
white (0xFFFFFFFF)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, ">", level, 0xFFFFFFFF, 0xFFFFFFFF);
    // then, remaining pixels and make them all with "color" black (0xFF000000)
    bmd.bitmapData.threshold(bmd_copy, rect, dest, "<=", level, 0xFF000000, 0xFFFFFFFF);
    // Note: as we have no alpha channel in our default BitmapData (pixelValue), we left it to
its full value, a white mask (0xffffffff)
}
```

Прочитайте Манипуляция и фильтрация растровых изображений онлайн: https://riptutorial.com/ru/actionscript-3/topic/8055/манипуляция-и-фильтрация-растровых-изображений

глава 8: Объектно-ориентированное программирование

Examples

«Перегруженный» конструктор через статический метод

Перегрузка конструктора недоступна в As3.

Чтобы обеспечить другой способ получения экземпляра класса, может быть предоставлен `public static` метод для использования в качестве альтернативного «конструктора».

Примером этого является `flash.geom.Point` , представляющий 2D-объект. Координаты для определения точки могут быть

- **декартовой** в регулярном конструкторе

  ```
  public function Point(x:Number = 0, y:Number = 0)
  ```

  пример использования:

  ```
  var point:Point = new Point(2, -.5);
  ```

- **полярный** по статическому методу

  ```
  public static function polar(len:Number, angle:Number):Point
  ```

  пример использования:

  ```
  var point:Point = Point.polar(12, .7 * Math.PI);
  ```

  Потому что это не фактический конструктор, нет `new` ключевого слова.

set & get функции

Чтобы обеспечить инкапсуляцию, переменные-члены класса должны быть `private` и доступны только для `public` помощью общедоступных методов `get` / `set` access. Это обычная практика префикса частных полей с _

```
public class Person
{
    private var _name:String = "";

    public function get name():String{
```

```
        return _name;
        //or return some other value depending on the inner logic of the class
    }


    public function set name(value:String):void{
        //here you may check if the new value is valid
        //or maybe dispatch some update events or whatever else
        _name = value;
    }
```

Иногда вам даже не нужно создавать `private` поле для пары `get` / `set` .
Например, в элементе управления, таком как пользовательская радиогруппа, вам нужно знать, какой переключатель выбран, но вне класса вам нужен только способ `get` / `set` только выбранное значение:

```
public function get selectedValue():String {
    //just the data from the element
    return _selected ? _selected.data : null;
}
public function set selectedValue(value:String):void {
    //find the element with that data
    for (var i:int = 0; i < _elems.length; i++) {
        if (_elems[i].data == value) {
            _selected = _elems[i];//set it
            processRadio();//redraw
            return;
        }
    }
}
```

пакеты

Пакеты - это пучки классов. Каждый класс должен быть объявлен в пакете с использованием оператора `package` . За оператором `package` следует имя вашего пакета или за ним ничего не следует добавлять в пакет верхнего уровня. Подпакеты создаются с использованием точечной (`.`) Делимитации. За операцией пакета следует блок, который будет содержать *определение одного* `class` . Примеры:

```
package {
    // The top level package.
}

package world {
    // A package named world.
}

package world.monsters {
    // A package named monsters within a package named world.
}
```

Пакеты должны соотноситься с файловой структурой классов относительно исходного корня. Предполагая, что у вас есть корневая папка источника с именем `src` ,

вышеуказанное может быть правильно представлено в файловой системе следующим образом:

```
src
    TopLevelClass.as

    world
        ClassInWorldPackage.as
        AnotherClassInWorldPackage.as

        monsters
            Zombie.as
```

Метод переопределения

Когда вы `extend` класс, вы можете `override` методы, которые определяет унаследованный класс, используя ключевое слово `override` :

```
public class Example {
    public function test():void {
        trace('It works!');
    }
}

public class AnotherExample extends Example {
    public override function test():void {
        trace('It still works!');
    }
}
```

Пример:

```
var example:Example = new Example();
var another:AnotherExample = new AnotherExample();

example.test(); // Output: It works!
another.test(); // Output: It still works!
```

Вы можете использовать ключевое слово `super` для ссылки на исходный метод из унаследованного класса. Например, мы могли бы изменить тело `AnotherExample.test()` на:

```
public override function test():void {
    super.test();
    trace('Extra content.');
}
```

В результате чего:

```
another.test(); // Output: It works!
                //         Extra content.
```

Переопределение конструкторов классов немного отличается. `override` опущен, и доступ к

унаследованному конструктору выполняется просто с помощью функции `super()` :

```
public class AnotherClass extends Example {
    public function AnotherClass() {
        super(); // Call the constructor in the inherited class.
    }
}
```

Вы также можете переопределить методы `get` и `set` .

геттер и сеттер

Getters и seters - это методы, которые ведут себя как свойства. это означает, что они имеют функциональную структуру, но при их использовании они используются так же, как и свойства:

Структура геттерных функций :

они должны иметь `get` ключевое слово после `function` ключевого слова и перед именем функции, без аргументов, типа возврата указан и должен возвращать значение:

```
public function get myValue():Type{
    //anything here
    return _desiredValue;
}
```

Синтаксис :

для получения значения из геттера, синтаксис такой же, как получение значения из свойства (no parens `()`).

```
trace(myValue);
```

Структура функций сеттера :

они должны `set` ключевое слово после ключевого слова `function` и перед именем функции, с одним аргументом и возвратом значения.

```
public function set myValue(value:Type):void{
    //anything here
    _desiredProperty=value;
}
```

Синтаксис :

для установки значения сеттера, синтаксис будет таким же, как установка значения для свойства (с использованием знака равенства `=` затем значение).

```
myValue=desiredValue;
```

установка геттера и сеттера для одного значения :

> Примечание: если вы создаете только getter или только setter с именем, это свойство будет доступно только для чтения или только для установки.

чтобы сделать свойство как читаемым, так и настраиваемым, должно создать геттер и сеттер с:

1. то же имя.
2. тот же тип (тип возвращаемого значения для геттера и тип входного значения (аргумент) для сеттера,

> Примечание: геттеры и сеттеры не должны иметь имя, аналогичное другим свойствам или методам.

Использование геттеров и сеттеров:

Использование геттеров и сеттеров, а не обычных свойств имеет много преимуществ:

1. **создание свойств только для чтения или только для установки:**
например, количество детей в экранном объекте. он не может быть установлен.

2. **Доступ к частной собственности:**
пример:

```
private var _private:Type=new Type();
//note that function name "private" is not same as variable name "_private"
public function get private():Type{
    return _private;
}
```

3. **когда требуется некоторое изменение после установки значения:**
в этом примере изменение этого свойства должно быть уведомлено:

```
public static function set val:(input:Type):void{
    _desiredProperty=input;
    notifyValueChanged();
}
```

и многие другие виды использования

Прочитайте Объектно-ориентированное программирование онлайн:
https://riptutorial.com/ru/actionscript-3/topic/2042/объектно-ориентированное-программирование

глава 9: Оптимизация производительности

Examples

Векторная графика

Векторные графические изображения представлены множеством данных, которые должны вычисляться ЦП (векторные точки, дуги, цвета и т. Д.). Все, кроме простых фигур с минимальными точками и прямыми линиями, будет потреблять огромное количество ресурсов ЦП.

Существует флаг «Кэш как битмап», который может быть включен. Этот флаг хранит результат рисования векторного объекта DisplayObject для более быстрого перерисовки. Ловушка этого заключается в том, что если есть какие-либо преобразования, применяемые к объекту, все это нужно перерисовать и повторно кэшировать. Это может быть медленнее, чем вообще не включать его, если применяются кадровые преобразования (вращение, масштабирование и т. Д.).

Как правило, рендеринг графики с использованием растровых изображений намного более эффективен, чем использование векторной графики. Библиотеки, такие как flixel, используют это для рендеринга спрайтов на «холсте» без снижения частоты кадров.

Текст

Текст рендеринга потребляет много CPU. Шрифты отображаются так же, как векторная графика, и содержат множество векторных точек для каждого символа. Изменение кадрового покрова *приведет к* ухудшению производительности. Флаг «Cache as bitmap» чрезвычайно полезен, если используется правильно, что означает, что вы должны избегать:

- Часто изменяйте текст.
- Преобразование текстового поля (поворот, масштабирование).

Простые методы, такие как перенос текстовых обновлений в оператор `if` будут иметь большое значение:

```
if (currentScore !== oldScore) {
    field.text = currentScore;
}
```

Текст можно визуализировать с помощью антиалиасированного рендерера, встроенного в Flash, или с помощью «шрифтов устройства». Использование «шрифтов устройства» делает визуализацию текста намного быстрее, хотя при этом текст кажется зазубренным (aliased). Кроме того, шрифты устройства требуют, чтобы шрифт был предварительно

установлен вашим конечным пользователем, или текст может «исчезнуть» на ПК пользователя, хотя он выглядит отлично на вашем компьютере.

```
field.embedFonts = false; // uses "device fonts"
```

Вектор и для каждого массива vs и для

Использование `Vector.<T>` Типа и `for each` цикла является более производительным , чем обычный массив и `for` цикла:

Хорошо:

```
var list:Vector.<Sprite> = new <Sprite>[];

for each(var sprite:Sprite in list) {
    sprite.x += 1;
}
```

Плохой:

```
var list:Array = [];

for (var i:int = 0; i < list.length; i++) {
    var sprite:Sprite = list[i];

    sprite.x += 1;
}
```

Быстрое удаление элементов массива

Если вам не требуется, чтобы массив находился в каком-либо определенном порядке, небольшой трюк с `pop()` предоставит вам огромные выигрыши в производительности по сравнению с `splice()` .

Когда вы `splice()` массив, индекс последующих элементов в этом массиве должен быть уменьшен на 1. Этот процесс может потреблять большой кусок времени, если массив большой, а объект, который вы удаляете, ближе к началу этого массива ,

Если вы не заботитесь о порядке элементов в массиве, вместо этого вы можете заменить элемент, который хотите удалить, с элементом, который вы `pop()` из конца массива. Таким образом, индексы всех остальных элементов массива остаются теми же, и процесс не ухудшается по мере увеличения длины вашего массива.

Пример:

```
function slowRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);
```

```
    if (index >= 0) list.splice(index, 1);
}

function fastRemove(list:Array, item:*):void {
    var index:int = list.indexOf(item);

    if (index >= 0) {
        if (index === list.length - 1) list.pop();

        else {
            // Replace item to delete with last item.
            list[index] = list.pop();
        }
    }
}
```

Векторы вместо массивов

Flash Player 10 представил тип Vector. <*>, Который был быстрее, чем массив. Однако это не совсем так. Только следующие типы Vector быстрее, чем аналоги Array, благодаря тому, как они реализованы в Flash Player.

- `Vector.<int>` - вектор 32-битных целых чисел
- `Vector.<uint>` - вектор 32-разрядных целых без знака
- `Vector.<Double>` - вектор 64-битных поплавков

Во всех других случаях использование массива будет более эффективным, чем использование векторов, для всех операций (создание, манипуляция и т. Д.). Однако, если вы хотите «сильно напечатать» свой код, вы можете использовать Vectors, несмотря на замедление. FlashDevelop имеет синтаксис, который позволяет выпадающим `/*ObjectType*/Array` завершения кода работать даже для массивов, используя `/*ObjectType*/Array` .

```
var wheels:Vector.<Wheel> // strongly typed, but slow

var wheels:/*Wheel*/Array // weakly typed, but faster
```

Повторное использование и объединение графики

Создание и настройка объектов `Sprite` и `TextField` во время выполнения может быть дорогостоящим, если вы создаете сотни тысяч из них на одном кадре. Поэтому общий трюк - это объединение этих объектов для последующего повторного использования. Помните, что мы просто не пытаемся оптимизировать время создания (`new Sprite()`), а также конфигурацию (настройку свойств по умолчанию).

Допустим, мы строили компонент списка, используя сотни объектов TextField. Когда вам нужно создать новый объект, проверьте, можно ли повторно использовать существующий объект.

```
var pool:Array = [];

if (pool.length > 0){

    // reuse an existing TextField
    var label = pool.pop();

}else{
    // create a new TextField
    label = new TextField();

    // initialize your TextField over here
    label.setDefaultTextFormat(...);
    label.multiline = false;
    label.selectable = false;
}

// add the TextField into the holder so it appears on-screen
// you will need to layout it and set its "text" and other stuff seperately
holder.addChild(label);
```

Позже, когда вы уничтожаете свой компонент (или удаляете его с экрана), не забудьте добавить неиспользуемые метки обратно в пул.

```
foreach (var label in allLabels){
    label.parent.removeChild(label); // remove from parent Sprite
    pool.push(label); // add to pool
}
```

В большинстве случаев лучше всего создать пул за использование вместо глобального пула. Недостатки создания глобального пула - вам нужно повторно инициализировать объект каждый раз, чтобы извлечь его из пула, чтобы отменить настройки, выполняемые другими функциями. Это одинаково дорого и в значительной степени отрицает повышение эффективности использования пула в первую очередь.

Прочитайте Оптимизация производительности онлайн: https://riptutorial.com/ru/actionscript-3/topic/2215/оптимизация-производительности

глава 10: Основы разработки игр

Вступление

[! [введите описание изображения] [1]] [1] **основы** разработки игры. ----------
Обратите внимание : этот набор руководств / статей содержит много концепций, которые могут быть предоставлены в виде разделенных тем. мы должны освежить их в уме и немного узнать о реализации наиболее важных частей видеоигры с помощью actionscript-3.
[1]: https://i.stack.imgur.com/CUIsz.png

Examples

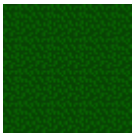
изометрический персонаж анимация + движение

① понятия, используемые в этом примере:

Учебный класс	использование
`URLRequest` + `Loader` + `Event`	Загрузка карты атласа (спрайт) с внешнего пути.
`BitmapData` + `Sprite` + `beginBitmapFill` + `Matrix` + `stageWidth` & `stageHeight`	рисование загруженных ресурсов в bitmapdata, используя черепичные растровые изображения, рисование с преобразованием.
`MovieClip` + `scrollRect` + `Bitmap` + `Rectangle`	создание и перемещение символа movieclip используя битмап в качестве временной шкалы.
`KeyboardEvent`	обнаружение пользовательских входов
`Event.EXIT_FRAME`	реализация функции Loop

---

② Ресурсы: **(нет разрешения на использование этих ресурсов в коммерческих целях)**



③ Код и комментарии:

> **Примечание:** FPS 15 Используется для этого учебника, его рекомендуется, но если вам нужно больше, вы должны сами изменить часть кода.

сначала мы должны скачать наши ресурсы из внешних URL-адресов.

```
const src_grass_tile_url:String = "https://i.stack.imgur.com/sjJFS.png";
const src_character_atlas_url:String = "https://i.stack.imgur.com/B7ztZ.png";

var loader:Loader = new Loader();
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setGround);
loader.load(new URLRequest(src_grass_tile_url));
```

---

setGround будет скопирован, как только `src_grass_tile_url` будет загружен и готов к использованию. в сопровождении реализации setGround, чтобы получить ресурс и нарисовать его в качестве фона игры

```
function setGround(e:Event):void {
    /* drawing ground */
    /* loader is a displayObject, so we can simply draw it into the bitmap data*/
    /* create an instance of Bitmapdata with same width and height as our window*/
    /* (also set transparent to false because grass image, does not contains any transparent
pixel) */
    var grass_bmd:BitmapData = new BitmapData(loader.width, loader.height, false, 0x0);
    /* time to draw */
    grass_bmd.draw(loader); // drawing loader into the bitmapData
    /* now we have to draw a tiled version of grass_bmd inside a displayObject Sprite to
displaying
       BitmapData on stage */
    var grass_sprite:Sprite = new Sprite();
    // for drawing a bitmap inside sprite, we must use <beginBitmapFill> with graphic property
of the sprite
    // then draw a full size rectangle with that Fill-Data
    // there is a repeat mode argument with true default value so we dont set it true again.
    // use a matrix for scalling grass Image during draw to be more cute!
    var mx:Matrix = new Matrix();
    mx.scale(2, 2);
    grass_sprite.graphics.beginBitmapFill(grass_bmd, mx);
    grass_sprite.graphics.drawRect(0, 0, stage.stageWidth, stage.stageHeight);
    // now add sprite to displayobjectcontainer to be displayed
    stage.addChild(grass_sprite);

    // well done, ground is ready, now we must initialize our character
    // first, load its data, i just re-use my loader for loading new image, but with another
complete handler (setCharacter)
    // so remove existing handler, then add new one
    loader.contentLoaderInfo.removeEventListener(Event.COMPLETE, setGround);
    loader.contentLoaderInfo.addEventListener(Event.COMPLETE, setCharacter);
    loader.load(new URLRequest(src_character_atlas_url));
}
```

код хорошо прокомментирован, после того, как мы сделали с землей, его время для реализации персонажа. character также содержит ресурс, который должен загружаться

одинаково. поэтому в конце `setGround` мы направляемся к `setCharacter` который является еще одним полным обратным вызовом.

```
function setCharacter(e:Event):void {
    // let assuming that what is really character!
    // a set of images inside a single image!
    // that images are frames of our character (also provides movement for different
directions)
    // first load this
    var character_bmd:BitmapData = new BitmapData(loader.width, loader.height, true, 0x0); //
note character is transparent
    character_bmd.draw(loader);
    // take a look at sprite sheet, how many frames you see?
    // 42 frames, so we can get width of a single frame
    const frame_width:uint = character_bmd.width / 42; // 41 pixels
    // as i show you above, to displaying a BitmapData, we have to draw it using a
DisplayObject,
    // another way is creating a Bitmap and setting its bitmapdata
    var character_bmp:Bitmap = new Bitmap(character_bmd);
    // but its not enough yet, a movieClip is necessary to cliping and animating this bitmap
(as a child of itself)
    var character_mc:MovieClip = new MovieClip();
    character_mc.addChild(character_bmp);
    character_bmp.name = "sprite_sheet"; // setting a name to character_bmp, for future
accessing
    character_mc.scrollRect = new Rectangle(0, 0, frame_width, character_bmd.height); //
cliping movieclip, to dusplaying only one frame
    character_mc.name = "character"; // setting a name to character_mc, for future accessing
    stage.addChild(character_mc); // adding it to stage.
    // well done, we have a character, but its static yet! 2 steps remaining. 1 controlling 2
animating
    // at first setting a control handler for moving character in 8 directions.
    stage.addEventListener(KeyboardEvent.KEY_DOWN, keyDown);
    stage.addEventListener(KeyboardEvent.KEY_UP, keyUp);
}
```

теперь персонаж готов к управлению. он хорошо отображен и готов к управлению. поэтому при подключении клавиатуры и прослушивании клавиш со стрелками в следующем коде:

```
// we storing key stats inside <keys> Object
var keys:Object = {u:false, d:false, l:false, r:false};
function keyDown(e:KeyboardEvent):void {
    switch (e.keyCode) {
        case 38: //up
            keys.u = true;
            break;
        case 40: //down
            keys.d = true;
            break;
        case 37: //left
            keys.l = true;
            break;
        case 39: //right
            keys.r = true;
            break;
    }
}
function keyUp(e:KeyboardEvent):void {
```

```
    switch (e.keyCode) {
        case 38: //up
            keys.u = false;
            break;
        case 40: //down
            keys.d = false;
            break;
        case 37: //left
            keys.l = false;
            break;
        case 39: //right
            keys.r = false;
            break;
    }
}
```

`keys:Object` хранит логическую переменную 4 для каждой клавиши со стрелкой, перемещение процесса должно выполняться внутри функции обновления (Loop) игры, поэтому мы должны передавать ей статистику клавиатуры. позволяет реализовать функцию **Loop** .

```
// initialize game Loop function for updating game objects
addEventListener(Event.EXIT_FRAME, loop);

// speed of character movement
const speed:Number = 5;
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;
}
```

скорость - вспомогательная константа, определяет скорость характера. выше код представляет собой простое движение в 8 направлениях с этим низким приоритетом: `Up > Down Left > Right` . поэтому, если стрелки вверх и вниз будут нажаты в одно и то же время, символ перемещается только *вверх* (не замерзает).

отлично сработано!!! осталось только один шаг, анимация, самая важная часть этого учебника

что такое анимация? набор ключевых кадров, который содержит по меньшей мере один кадр
позволяет создавать наши ключевые фреймы Объект, который содержит имя ключевых кадров
а также некоторые данные о начале и конце кадра этого ключевого кадра
Обратите внимание , что в изометрических играх каждый ключевой кадр содержит 8 направлений (может быть уменьшен до 5 с использованием переворота)

```
var keyframs:Object = {
```

```
    idle: {up:[0,0], up_right:[1,1], right:[2,2], down_right:[3,3], down:[4,4]}, // [2,2]
means start frame is 2 and end frame is 2
    run: {up:[5,10], up_right:[11,16], right:[17,22], down_right:[23,28], down:[29,34]}
};
```

мы должны игнорировать оставшиеся кадры, этот пример обеспечивает только простоя и анимацию запуска
например, начальный кадр бездействующей анимации с направлением справа: < keyframs.idle.right [0]>
теперь позволяет реализовать функцию Animator

```
var current_frame:uint;
function animate(keyframe:Array):void {
    // how it works
    // just called with a keyframe with direction (each frame),
    // if keyframe is what is already playing, its just moved to next frame and got updated
(or begning frame for loop)
    // other wise, just moved to begining frame of new keyframe
    if (current_frame >= keyframe[0] && current_frame <= keyframe[1]) { // check if in bound
        current_frame++;
        if (current_frame > keyframe[1]) // play back if reached
            current_frame = keyframe[0];
    } else {
        current_frame = keyframe[0]; // start new keyframe from begining
    }
    // moving Bitmap inside character MovieClip
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    var sprite_sheet:Bitmap = character.getChildByName("sprite_sheet") as Bitmap;
    sprite_sheet.x = -1 * current_frame * character.width;
}
```

прочитайте комментарии вышеприведенной функции, однако основным заданием этой функции является перемещение *sprite_sheet Растровое* `Bitmap` внутри *символа* `MovieClip` .

мы знаем, что каждое обновление должно выполняться внутри функции Loop, поэтому мы будем вызывать эту функцию из Loop со связанными ключевыми кадрами. это обновленная функция Loop:

```
// speed of character movement
const speed:Number = 8;
var last_keyStat:Object = {u:false, d:false, l:false, r:false}; // used to getting a backup of
previous keyboard stat for detecting correct idle direction
// this function will be called on each frame, with same rate as your project fps
function loop(e:Event):void {
    if (keys.u) stage.getChildByName("character").y -= speed;
    else if (keys.d) stage.getChildByName("character").y += speed;
    if (keys.l) stage.getChildByName("character").x -= speed;
    else if (keys.r) stage.getChildByName("character").x += speed;

    // animation detection
    if (keys.u && keys.l) { animate(keyframs.run.up_right); flip(true); }
    else if (keys.u && keys.r) { animate(keyframs.run.up_right); flip(false); }
    else if (keys.d && keys.l) { animate(keyframs.run.down_right); flip(true); }
    else if (keys.d && keys.r) { animate(keyframs.run.down_right); flip(false); }
    else if (keys.u) { animate(keyframs.run.up); flip(false); }
```

```
    else if (keys.d) { animate(keyframs.run.down); flip(false); }
    else if (keys.l) { animate(keyframs.run.right); flip(true); }
    else if (keys.r) { animate(keyframs.run.right); flip(false); }
    else {
        // if character dont move, so play idle animation
        // what is the best practice to detecting idle direction?
        // my suggestion is to sotring previous keyboard stats to determining which idle
direction is correct
        // do any better thing if possible (absolutely is possible)
        // i just simply copy it from above, and replaced (keys) with (last_keyStat) and (run)
with (idle)
        if (last_keyStat.u && last_keyStat.l) { animate(keyframs.idle.up_right); flip(true); }
        else if (last_keyStat.u && last_keyStat.r) { animate(keyframs.idle.up_right);
flip(false); }
        else if (last_keyStat.d && last_keyStat.l) { animate(keyframs.idle.down_right);
flip(true); }
        else if (last_keyStat.d && last_keyStat.r) { animate(keyframs.idle.down_right);
flip(false); }
        else if (last_keyStat.u) { animate(keyframs.idle.up); flip(false); }
        else if (last_keyStat.d) { animate(keyframs.idle.down); flip(false); }
        else if (last_keyStat.l) { animate(keyframs.idle.right); flip(true); }
        else if (last_keyStat.r) { animate(keyframs.idle.right); flip(false); }
    }
    // update last_keyStat backup
    last_keyStat.u = keys.u;
    last_keyStat.d = keys.d;
    last_keyStat.l = keys.l;
    last_keyStat.r = keys.r;
}
```

читать комментарии, мы просто просто обнаруживаем истинный ключевой кадр с помощью клавиатуры. то также сделать то же самое для обнаружения простоя анимации. для незанятых анимаций у нас нет ключевого ввода для использования, чтобы определить, какой символ направления включен, поэтому вспомогательная переменная simle может быть удобна для хранения предыдущего состояния клавиатуры (last_keyStat).

также есть новая функция `flip` которая является другой вспомогательной функцией, используемой для моделирования отсутствующих анимаций (left + up_left + down_left), и эта функция выполняет некоторые исправления, которые комментируются ниже:

```
// usage of flip function is because of Movieclip registration point, its a fix
// as the registration point of MovieClip is not placed in center, when flipping animation
(for non existing directions inside spritesheet)
// character location changes with an unwanted value equal its width, so we have to prevent
this and push it back or forward during flip
function flip(left:Boolean):void {
    var character:MovieClip = stage.getChildByName("character") as MovieClip;
    if (left) {
        if (character.scaleX != -1) {
            character.scaleX = -1;
            character.x += character.width; // comment this line to see what happen without
this fix
        }
    } else {
        if (character.scaleX != 1) {
            character.scaleX = 1;
```

```
            character.x -= character.width; // comment this line to see what happen without
this fix
        }
    }
}
```

наша работа заканчивается здесь. особая благодарность редактору, которая делает этот учебник более непримиримым. Также здесь представлена демо-версия этого учебника плюс внешняя ссылка полного кода.

④ Внешние ссылки:

- **полный код**
- **Демо-версия**

Прочитайте Основы разработки игр онлайн: https://riptutorial.com/ru/actionscript-3/topic/8237/основы-разработки-игр

глава 11: Отзывчивый дизайн приложения

Examples

Основное адаптивное приложение

```
package
{
    import flash.display.Sprite;
    import flash.display.StageAlign;
    import flash.display.StageScaleMode;
    import flash.events.Event;

    public class Main extends Sprite
    {
        //Document Class Main Constructor
        public function Main()
        {
            //Sometimes stage isn't available yet, so if not, wait for it before proceeding
            if (!stage) {
                addEventListener(Event.ADDED_TO_STAGE, stageReady);
            }else {
                stageReady();
            }
        }

        protected function stageReady(e:Event = null):void {
            //align the stage to the top left corner of the window/container
            stage.align = StageAlign.TOP_LEFT;
            //don't scale the content when the window resizes
            stage.scaleMode = StageScaleMode.NO_SCALE;

            //listen for when the window is resized
            stage.addEventListener(Event.RESIZE, stageResized);
        }

        protected function stageResized(e:Event):void {
            //use stage.stageWdith & stage.stageHeight to repostion and resize items
        }
    }
}
```

Выполняя длительные процессы и не получая безответственного приложения

Бывают ситуации, когда вам нужно вычислить что-то действительно большое в вашем приложении Flash, не прерывая работу пользователя. Для этого вам нужно разработать свой длительный процесс как многоэтапный процесс с сохраненным состоянием между итерациями. Например, вам необходимо выполнить фоновое обновление множества внутренних объектов, но если вы хотите обновить их все сразу простым `for each (var o in objects) { o.update(); }` , Flash ненадолго (или не так кратко) перестает отвечать на

запросы пользователя. Таким образом, вам нужно выполнить одно или несколько обновлений для каждого кадра.

```
private var processing:Boolean;      // are we in the middle of processing
private var lastIndex:int;           // where did we finish last time
var objects:Vector.<UpdatingObject>; // the total list of objects to update
function startProcess():Boolean {
    if (processing) return false; // already processing - please wait
    startProcessing=true;
    lastIndex=0;
    if (!hasEventListener(Event.ENTER_FRAME,iterate))
        addEventListener(Event.ENTER_FRAME,iterate); // enable iterating via listener
}
private function iterate(e:Event):void {
    if (!processing) return; // not processing - skip listener
    objects[lastIndex].update(); // perform a quantum of the big process
    lastIndex++; // advance in the big process
    if (lastIndex==objects.length) {
        processing=false; // finished, clear flag
    }
}
```

Передовая обработка может включать использование `getTimer()` для проверки прошедшего времени и предоставления другой итерации, если время не закончилось, разделение `update()` на несколько функций, если обновление слишком велико, отображение прогресса в другом месте, настройка процесса итерации для адаптации к изменениям список объектов для обработки и многое другое.

Прочитайте Отзывчивый дизайн приложения онлайн: https://riptutorial.com/ru/actionscript-3/topic/1615/отзывчивый-дизайн-приложения

глава 12: Отправка и получение данных с серверов

Examples

Выполнение запроса от Flash

Классы `URLRequest` и `URLLoader` работают вместе, чтобы запросы от Flash к внешним ресурсам. `URLRequest` определяет информацию о запросе, например, тело запроса и тип метода запроса, и `URLLoader` ссылается на это для выполнения фактического запроса и предоставляет средство для уведомления, когда ответ получен от ресурса.

Пример:

```
var request:URLRequest = new URLRequest('http://stackoverflow.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

function responseReceived(event:Event):void {
    trace(event.target.data); // or loader.data if you have reference to it in
                              // this scope.
}
```

Добавление переменных в ваш запрос

Класс `URLVariables` позволяет вам определять данные для отправки вместе с `URLRequest` .

Пример:

```
var variables:URLVariables = new URLVariables();

variables.prop = "hello";
variables.anotherProp = 10;

var request:URLRequest = new URLRequest('http://someservice.com');
request.data = variables;
```

Вы можете отправить запрос через `URLLoader` или открыть URL-адрес запроса с переменными, указанными в querystring, используя `navigateToURL` .

Изменение метода HTTP (GET, POST, PUT и т. Д.)

Класс `URLRequestMethod` содержит константы для различных типов запросов, которые вы можете сделать. Эти константы должны быть выделены для свойства `method` `URLRequest` :

```
var request:URLRequest = new URLRequest('http://someservice.com');
request.method = URLRequestMethod.POST;
```

Обратите внимание, что только `GET` и `POST` доступны вне среды выполнения AIR.

Мои данные ответа всегда равны нулю, что означает «асинхронный»?

Когда Flash запрашивает данные из внешнего источника, эта операция является *асинхронной* . Самое основное объяснение того, что это означает, заключается в том, что данные загружаются «в фоновом режиме» и запускают обработчик событий, который вы назначаете `Event.COMPLETE` когда он получен. Это может произойти в любой момент вашего приложения.

Ваши данные **НЕ** будут доступны сразу после вызова `load()` на вашем `URLLoader` . Вы **должны** подключить прослушиватель событий для `Event.COMPLETE` и взаимодействовать с ответом там.

```
var request:URLRequest = new URLRequest('http://someservice.com');
var loader:URLLoader = new URLLoader();

loader.addEventListener(Event.COMPLETE, responseReceived);
loader.load(request);

trace(loader.data); // Will be null.

function responseReceived(event:Event):void {
    trace(loader.data); // Will be populated with the server response.
}

trace(loader.data); // Will still be null.
```

Вы не можете обойти это с помощью любых небольших трюков, например, с помощью `setTimeout` или аналогичного:

```
setTimeout(function() {
    trace(loader.data); // Will be null if the data hasn't finished loading
                        // after 1000ms (which you can't guarantee).
}, 1000);
```

Междоменные запросы

Flash не будет загружать данные из домена, отличного от того, на котором работает ваше приложение, если только этот домен не имеет политики crossdomain XML в корне домена (например, `http://somedomain.com/crossdomain.xml`) или где-то, может быть нацелена на `Security.loadPolicyFile()` . Файл crossdomain.xml - это то, где вы можете указать домены, которые могут запрашивать ваш сервер для данных из приложения Flash.

Пример *наиболее разрешающего* crossdomain.xml:

```
<?xml version="1.0" ?>
<cross-domain-policy>
  <allow-access-from domain="*"/>
  <allow-http-request-headers-from domain="*" headers="*"/>
</cross-domain-policy>
```

> Обратите внимание, что этот пример **не должен использоваться в производственных средах** , используйте более ограничительный экземпляр.

Более ограничительный конкретный crossdomain.xml будет выглядеть так:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-
policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="master-only" />

    <allow-access-from domain="*.domain.com" to-ports="80,843,8011" />
    <allow-access-from domain="123.123.123.123" to-ports="80,843,8011" />
</cross-domain-policy>
```

Ресурсы:

- Спецификация файла политики crossdomain .

Прочитайте Отправка и получение данных с серверов онлайн:
https://riptutorial.com/ru/actionscript-3/topic/1893/отправка-и-получение-данных-с-серверов

глава 13: Понимание «Ошибка 1009: невозможно получить доступ к свойству или методу ссылки на нулевой объект»

Вступление

Ошибка 1009 - это общая ошибка, возникающая при попытке получить значение из переменной или свойства с `null` значением. Приведенные примеры вызывают различные случаи возникновения этой ошибки вместе с некоторыми рекомендациями по устранению ошибки.

замечания

Страшный и часто задаваемый вопрос «Ошибка 1009: невозможно получить доступ к свойству или методу ссылки на нулевой объект» - это сигнал о том, что некоторые данные отображаются нулевыми, но их пытаются использовать в качестве заселенного объекта. Есть довольно много типов проблем, которые могут вызвать это поведение, и каждый из них должен быть протестирован против кода, где возникла ошибка.

Examples

Этап недоступен

Иногда разработчики пишут код, который хочет получить доступ к `stage` , или к `stage` Flash, чтобы добавить слушателей. Он может работать в первый раз, а затем внезапно не работает и создает ошибку 1009. Этот код может быть даже на временной шкале, так как это первая инициатива по добавлению кода там, и многие учебники, которые все еще существуют, используют слот кода временной шкалы для размещения кода.

```
public class Main extends MovieClip {
    public function Main() {
        stage.addEventListener(Event.ENTER_FRAME,update); // here
```

Причина, по которой этот код не работает, прост: сначала создается экземпляр экранного объекта, затем добавляется в список отображения, а пока он не отображается в списке отображения, `stage` имеет значение null.

Хуже, если код выглядит следующим образом:

```
stage.addEventListener(Event.ENTER_FRAME,update); // here
```

размещается на временной шкале. Он может даже работать некоторое время, в то время как `Main` объект удаляется на сцену через графический интерфейс. Затем их SWF загружается из другого SWF, и внезапно код прерывается. Это происходит потому, что кадры `Main` создаются по-другому, когда SWF загружается непосредственно игроком и когда загрузка обрабатывается асинхронно. Решение состоит в том, чтобы использовать прослушиватель `Event.ADDED_TO_STAGE` и поместить весь код, который обращается к нему, и поместить сам прослушиватель в AS-файл вместо временной шкалы.

Недопустимый тип

```
function listener(e:Event):void {
    var m:MovieClip=e.target as MovieClip;
    m.x++;
}
```

Если такой прослушиватель прикреплен к объекту, который не является потомком `MovieClip` (например, `Sprite`), то при неудачном провале будет проигнорировано, и любые последующие операции с его результатом будут вызывать ошибку 1009.

Неинтересный объект

```
var a:Object;
trace(a); // null
trace(a.b); // Error 1009
```

Здесь ссылка на объект объявляется, но никогда не назначается значение, будь то с `new` или присваиванием ненулевого значения. Запрос его свойств или метода приводит к ошибке 1009.

Многоуровневое выражение

```
x=anObject.aProperty.anotherProperty.getSomething().data;
```

Здесь любой объект до точки может оказаться нулевым, а использование методов, возвращающих сложные объекты, только увеличивает сложность для отладки нулевой ошибки. Худший случай, если метод подвержен посторонним отказам, скажем, получение данных по сети.

Результат необработанной функции

```
s=this.getChildByName("garbage");
if (s.parent==this) {...}
```

`getChildByName()` является одной из многих функций, которые могут возвращать значение null, если возникла ошибка при обработке его ввода. Поэтому, если вы получаете объект из

любой функции, которая может возвращать значение null, сначала проверьте значение null. Здесь свойство немедленно запрашивается без предварительной проверки, если `s` равно null, это приведет к ошибке 1009.

Забытый слушатель событий

```
addEventListener(Event.ENTER_FRAME,moveChild);
function moveChild(e:Event):void {
    childMC.x++;
    if (childMC.x>1000) {
        gotoAndStop(2);
    }
}
```

В этом примере будет перемещен `childMC` (добавлен в `Main` во время разработки), но сразу же будет `gotoAndStop()` 1009, как только будет `gotoAndStop()` , если этот `childMC` не существует во фрейме 2. Основная причина этого заключается в том, что всякий раз, ключевой кадр (кадр, который не наследует набор объектов предыдущего кадра), либо с использованием `gotoAndStop()` , `gotoAndPlay()` с кадром назначения, отделенным от текущего кадра ключевым фреймом, либо обычным воспроизведением, если SWF является анимация, содержимое текущего кадра **уничтожается,** а новое содержимое создается с использованием данных, хранящихся в графическом интерфейсе. Таким образом, если новый кадр не имеет дочернего `childMC` именем `childMC` , запрос свойства возвращает null, а 1009 будет `childMC` .

Тот же принцип применяется, если вы добавляете двух слушателей событий, но удаляете только один или добавляете слушателя к одному объекту, но пытаетесь удалить из другого. `removeEventListener` не будет предупреждать вас, если объект не имеет соответствующего `removeEventListener` событий, поэтому прочитайте код, который добавляет и удаляет прослушиватели событий.

Также обратите внимание: использование объектов `Timer` вызове `setInterval()` И `setTimeout()` также создает прослушиватели событий, и они также должны быть очищены должным образом.

Недействительная ссылка на объект на основе кадра

Иногда `gotoAndStop()` вызывается в середине кода, который ссылается на некоторые свойства на основе кадра. Но **сразу же после изменения рамки** все ссылки на свойства, существующие в текущем фрейме, недействительны, поэтому любая обработка, которая их включает, должна быть немедленно прекращена.

Существует два общих сценария такой обработки: во-первых, цикл не заканчивается после `gotoAndStop()` , как здесь:

```
for each (bullet in bullets) {
    if (player.hitTestObject(bullet)) gotoAndStop("gameOver");
```

```
}
```

Здесь ошибка 1009 означает, что `player` MC был уничтожен во время обработки `gotoAndStop()` , но цикл продолжается и ссылается на ссылку now-null, чтобы получить `hitTestObject()` . Если условие скажет `if (bullet.hitTestObject(player))` , ошибка будет # 2007 «Параметр hitTestObject не должен быть нулевым». Решение состоит в том, чтобы поставить оператор `return` сразу после вызова `gotoAndStop()` .

Второй случай - это несколько прослушивателей событий в одном и том же событии. Как это:

```
stage.addEventListener(Event.ENTER_FRAME,func1);
stage.addEventListener(Event.ENTER_FRAME,func2);
function func1(e:Event):void {
    if (condition()) {
        gotoAndStop(2);
    }
}
```

Здесь, если `condition()` истинно, первый прослушиватель выполнил бы `gotoAndStop()` , но второй слушатель все равно будет выполнен, и если он ссылается на объекты в кадре, будет вызвана ошибка 1009. Решение состоит в том, чтобы избежать нескольких прослушивателей в одном событии, в одном объекте, лучше иметь один прослушиватель, который обрабатывает все ситуации на этом событии и может нормально завершаться, если требуется изменение фрейма.

Прочитайте Понимание «Ошибка 1009: невозможно получить доступ к свойству или методу ссылки на нулевой объект» онлайн: https://riptutorial.com/ru/actionscript-3/topic/2098/понимание--ошибка-1009--невозможно-получить-доступ-к-свойству-или-методу-ссылки-на-нулевой-объект-

глава 14: Работа с видео

Examples

Загрузка и воспроизведение внешнего видеофайла

ссылка : `NetConnection` , `NetStream` , `Video`

связанные темы : Работа со звуком

Основной пример воспроизведения внешнего видеофайла (FLV, MP4, F4V). Код также будет воспроизводить аудиофайлы M4A.

```
var nc:NetConnection = new NetConnection();
nc.connect(null);

var ns:NetStream = new NetStream(nc);

var myVideo:Video = new Video();
addChild(myVideo);

myVideo.attachNetStream(ns);

ns.play("http://www.yourwebsite.com/somefile.mp4");
```

Обратите внимание, что в коде используется `nc.connect.null` ? Это связано с тем, что в этом случае нет необходимости создавать двустороннее одноранговое соединение (например, как ожидается в приложении видеочата), так как мы играем сохраненный файл.

Установив `nc.connect.null` необходимо предоставить ссылку на файл, который находится либо на веб-сервере, либо на локальном (то же местоположение / папку), что и на SWF.

- Для использования **веб-** файла: `ns.play("http://www.yourwebsite.com/somefile.mp4");`
- Для **локального** файла используйте: `ns.play("somefile.mp4");`

С NetStatusEvent

```
package {
    import flash.events.NetStatusEvent;
    import flash.net.NetStream;
    import flash.net.NetConnection;
    import flash.events.Event;
    import flash.media.Video;
    import flash.display.Sprite;
    public class VideoWithNetStatus extends Sprite {
```

```
        private var video:Video = new Video();
        private var nc:NetConnection;
        private var ns:NetStream;

        public function VideoWithNetStatus() {
            nc = new NetConnection();
            nc.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
            nc.connect(null);//or media server url
        }

         private function onStatus(e:NetStatusEvent):void{
             switch(e.info.code){
                 case 'NetConnection.Connect.Success':
                     connectStream();
                 break;
                 default:
                     trace(e.info.code);//to see any unhadled events
             }
         }
         private function connectStream():void{
             ns = new NetStream(nc);
             ns.addEventListener(NetStatusEvent.NET_STATUS, onStatus);
             addChild(video);
             video.attachNetStream(ns);
             ns.play('url/to/video.flv');
         }
    }
}
```

Прочитайте Работа с видео онлайн: https://riptutorial.com/ru/actionscript-3/topic/2406/работа-с-видео

глава 15: Работа с временной шкалой

Examples

Ссылка на основную шкалу времени или класс документа из других MovieClips

На временной шкале любого объекта `DisplayObject` который присоединен как потомок дерева отображения, вы можете использовать свойство `root` . Это свойство указывает на основную временную шкалу в случае нестандартного класса документа или класса документа, если вы его определяете.

Поскольку `root` задан `DisplayObject` , компилятор не позволит вам получить доступ к пользовательским методам или свойствам, определенным на основной временной шкале или в вашем классе документа, как:

```
root.myCustomProperty = 10;
root.myCustomMethod();
```

Чтобы обойти это, вы можете ввести `root` в свой класс документов в том случае, если у вас есть класс документа:

```
(root as MyDocumentClass).myCustomMethod();
```

Или `MovieClip` в случае отсутствия класса документа:

```
(root as MovieClip).myCustomMethod();
```

Причина, по которой в `MovieClip` работает, заключается в том, что `MovieClip` `dynamic` . Это означает, что компилятор позволяет объявлять на нем свойства и метод выполнения, предотвращая ошибки во время компиляции при попытке получить доступ к свойствам или методам, которые явно не определены в `MovieClip` . Недостатком этого является то, что вы теряете всю безопасность типа компиляции. Вам намного лучше объявить класс документа и придать этому.

Прочитайте Работа с временной шкалой онлайн: https://riptutorial.com/ru/actionscript-3/topic/2459/работа-с-временной-шкалой

глава 16: Работа с геометрией

Examples

Получение угла между двумя точками

Использование математики ванили:

```
var from:Point = new Point(100, 50);
var to:Point = new Point(80, 95);

var angle:Number = Math.atan2(to.y - from.y, to.x - from.x);
```

Используя новый вектор, представляющий разницу между двумя первыми:

```
var difference:Point = to.subtract(from);

var angle:Number = Math.atan2(difference.y, difference.x);
```

> Примечание: `atan2()` возвращает радианы, а не градусы.

Получение расстояния между двумя точками

Использование математики ванили:

```
var from:Point = new Point(300, 10);
var to:Point = new Point(75, 40);

var a:Number = to.x - from.x;
var b:Number = to.y - from.y;

var distance:Number = Math.sqrt(a * a + b * b);
```

Использование встроенных функций `Point` :

```
var distance:Number = to.subtract(from).length; // or
var distance:Number = Point.distance(to, from);
```

Преобразование радианов в градусы

```
var degrees:Number = radians * 180 / Math.PI;
```

Преобразование градусов в радианы

```
var radians:Number = degrees / 180 * Math.PI;
```

Значение круга в градусах и радианах

- Целый круг - `360` градусов или `Math.PI * 2` радианов.
- Половина этих значений должна составлять `180` градусов или `Math.PI`
- Четверть составляет `90` градусов или `Math.PI / 2` радианов.

Чтобы получить сегмент в процентах от всего круга в радианах:

```
function getSegment(percent:Number):Number {
    return Math.PI * 2 * percent;
}

var tenth:Number = getSegment(0.1); // One tenth of a circle in radians.
```

Перемещение точки вдоль угла

Предполагая, что у вас есть угол, который вы хотите переместить, и объект с значениями `x` и `y` вы хотите переместить:

```
var position:Point = new Point(10, 10);
var angle:Number = 1.25;
```

Вы можете перемещаться вдоль оси `x` с помощью `Math.cos` :

```
position.x += Math.cos(angle);
```

И ось `y` с `Math.sin` :

```
position.y += Math.sin(angle);
```

Вы можете, конечно, умножить результат `Math.cos` и `Math.sin` на расстояние, которое вы хотите путешествовать:

```
var distance:int = 20;

position.x += Math.cos(angle) * distance;
position.y += Math.sin(angle) * distance;
```

> Примечание. Угол входа должен быть в радианах.

Определите, находится ли точка внутри области прямоугольника

Вы можете проверить, находится ли точка внутри прямоугольника с помощью `Rectangle.containsPoint()` :

```
var point:Point = new Point(5, 5);
var rectangle:Rectangle = new Rectangle(0, 0, 10, 10);
```

```
var contains:Boolean = rectangle.containsPoint(point); // true
```

Прочитайте Работа с геометрией онлайн: https://riptutorial.com/ru/actionscript-3/topic/3201/работа-с-геометрией

глава 17: Работа с датой и временем

Examples

Ante meridiem (AM) или Post meridiem (PM) для 12-часовых часов

```
function meridiem(d:Date):String {
    return (d.hours > 11) ? "pm" : "am";
}
```

Количество дней в указанном месяце

```
/**
 * @param year   Full year as int (ex: 2000).
 * @param month  Month as int, zero-based (ex: 0=January, 11=December).
 */
function daysInMonth(year:int, month:int):int {
    return (new Date(year, ++month, 0)).date;
}
```

Является ли указанный год високосным годом

```
function isLeapYear(year:int):Boolean {
    return daysInMonth(year, 1) == 29;
}
```

Независимо от того, соблюдается ли сейчас летнее время

```
function isDaylightSavings(d:Date):Boolean {
    var months:uint = 12;
    var offset:uint = d.timezoneOffset;
    var offsetCheck:Number;

    while (months--) {
        offsetCheck = (new Date(d.getFullYear(), months, 1)).timezoneOffset;

        if (offsetCheck != offset)
            return (offsetCheck > offset);
    }

    return false;
}
```

Сегодняшняя дата начала, в полночь

```
function today(date:Date = null):Date {
    if (date == null)
        date = new Date();
```

```
    return new Date(date.fullYear, date.month, date.date, 0, 0, 0, 0);
}
```

Прочитайте Работа с датой и временем онлайн: https://riptutorial.com/ru/actionscript-3/topic/1926/работа-с-датой-и-временем

глава 18: Работа с объектами отображения

Синтаксис

1. `addChild(child)` - добавляет новый элемент в дочернее дерево этого объекта в качестве верхнего элемента.
2. `addChildAt(child, index)` - добавляет новый элемент в дочернее дерево этого объекта в указанной позиции. Самый нижний элемент имеет индекс 0.
3. `getChildAt(index)` - возвращает дочерний элемент с заданным индексом.
4. `getChildIndex(child)` возвращает индекс *прямого* дочернего элемента этого объекта. В противном случае создается исключение.
5. `removeChild(child)` - удаляет указанный прямой дочерний элемент из дочернего дерева этого объекта. Выдает исключение, если родительский родитель этого ребенка не равен `this` .
6. `removeChildAt(index)` - удаляет дочерний элемент, выбранный по индексу вместо ссылки. Выдает исключение, если дочернее дерево не является таким широким.
7. `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` - добавлен в Flash Player 11, удаляет подмножество детей по диапазону индексов или всех дочерних, если `removeChildren(beginIndex:int = 0, endIndex:int = 0x7fffffff))` без параметров.
8. `setChildIndex(child,index)` - изменяет индекс ребенка на новое значение, перемещая всех детей между ними, чтобы занять освобожденное место.
9. `swapChildren(child1,child2)` - свопит позиции двух детей в список отображения, не влияя на позиции других детей.
10. `swapChildrenAt(index1,index2)` - `swapChildrenAt(index1,index2)` детей, расположенных по их индексам.

замечания

Список отображения фактически является деревом и визуализируется с использованием первого алгоритма глубины. Любой объект, указанный ранее, будет отображаться ранее и может быть скрыт объектами, перечисленными позже. Все методы, которые можно использовать против дерева, можно применять для работы со списком отображения.

Examples

Введение в список отображения

В AS3 отображаемые активы не отображаются до тех пор, пока они не будут добавлены в список отображения.

Время выполнения AIR / Flash имеет иерархическую структуру отображения (отношения

родительского ребенка, в которой дети могут иметь своих собственных детей), причем эта `stage` является родителем верхнего уровня.

Чтобы добавить что-то в список отображения, вы используете `addChild` или `addChildAt` . Вот основной пример рисования круга и его добавления в список отображения:

```
var myCircle:Shape = new Shape();

myCircle.graphics.beginFill(0xFF0000); //red
myCircle.graphics.drawCircle(25, 25, 50);
myCircle.graphics.endFill();

this.addChild(myCircle); //add the circle as a child of `this`
```

Чтобы увидеть объект в приведенном выше примере, `this` (контекст кода) также должно быть в списке отображения, а также любых родителях, которых оно может иметь. В AS3 эта `stage` является самой главной родительской.

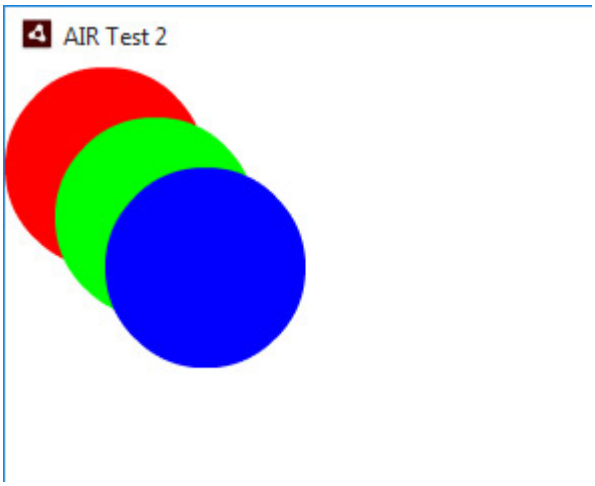
> Объекты отображения могут иметь только один родитель. Поэтому, если у ребенка уже есть родительский элемент, и вы добавляете его к другому объекту, он будет удален из предыдущего родителя.

Z-Order / Layering

Допустим, вы скопировали код из предыдущего примера, так что у вас было 3 круга:

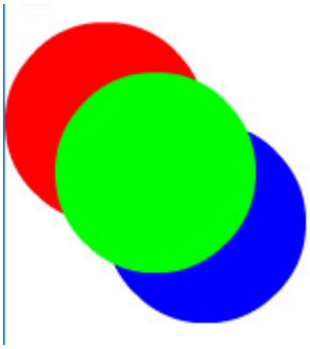
```
var redCircle:Shape = new Shape();
redCircle.graphics.beginFill(0xFF0000); //red
redCircle.graphics.drawCircle(50, 50, 50); //graphics.endFill is not required

var greenCircle:Shape = new Shape();
greenCircle.graphics.beginFill(0x00FF00); //green
greenCircle.graphics.drawCircle(75, 75, 50);

var blueCircle:Shape = new Shape();
blueCircle.graphics.beginFill(0x0000FF); //blue
blueCircle.graphics.drawCircle(100, 100, 50);

this.addChild(redCircle);
this.addChild(greenCircle);
this.addChild(blueCircle);
```

Поскольку метод `addChild` добавляет ребенка поверх всего остального в том же родителе, вы получите этот результат с элементами, расположенными в том же порядке, в котором вы используете addChild:



Если вы хотите, чтобы ребенок был многослойным относительно своих братьев и сестер, вы можете использовать `addChildAt` . С помощью `addChildAt` вы передаете другой параметр, указывающий индекс (z-порядок), на котором должен находиться ребенок. `0` является самым нижним положением / слоем.

```
this.addChild(redCircle);
this.addChild(greenCircle);
this.addChildAt(blueCircle,0); //This will add the blue circle at the bottom
```



Теперь синий круг находится под его братьями и сестрами. Если позже вы хотите изменить индекс дочернего элемента, вы можете использовать метод `setChildIndex` (для родителя ребенка).

```
this.setChildIndex(redCircle, this.numChildren - 1); //since z-index is 0 based, the top most
position is amount of children less 1.
```

Это изменит красный круг, чтобы он был выше всего остального. Приведенный выше код дает тот же результат, что и `this.addChild(redCircle)` .

Удаление объектов отображения

Чтобы удалить объекты, у вас есть `removeChildAt` методы `removeChild` и `removeChildAt` а также метод `removeChildren` .

```
removeChild(redCircle); //this will take redCircle off the display list
```

```
removeChildAt(0); //this will take the bottom most object off the display list

removeChildren(); //this will clear all children from the display list

removeChildren(1); //this would remove all children except the bottom most

removeChildren(1,3); //this would remove the children at indexes 1, 2 & 3
```

События

Когда дочерний элемент добавляется в список отображения, некоторые события запускаются с этим дочерним элементом.

- `Event.ADDED`
- `Event.ADDED_TO_STAGE`

И наоборот, есть также события удаления:

- `Event.REMOVED`
- `Event.REMOVED_FROM_STAGE`

Adobe Animate / Flash Professional

При работе с временными рамками FlashProfessional / Adobe Animate добавление чего-то к временной шкале автоматически обрабатывает нюансы отображения. Они автоматически добавляются и удаляются из списка отображения по временной шкале.

Однако хорошо иметь в виду, что:

> Если вы управляете кодом с помощью родительского элемента экранного объекта, созданного временной шкалой (с помощью addChild / setChildIndex), этот ребенок больше не будет удален автоматически по временной шкале и должен быть удален с помощью кода.

Расслоение

Могут возникнуть ситуации, когда вы решаете, что один набор экранных объектов всегда должен быть выше другого набора объектов, например, стрелки над головами, взрывы над чем-то, что только что взорвалось, и т. Д. Чтобы выполнить это как можно проще, вам нужно назначить и создайте набор `Sprite` s, расположите их по порядку снизу вверх, затем просто добавьте все объекты «выше», установленные на слой выше того, который используется для объектов «ниже».

```
var monsters:Vector.<Monster>;
var bullets:Vector.<Bullet>; // desired: bullets strictly above monsters
var monsterLayer:Sprite=new Sprite();
var bulletLayer:Sprite=new Sprite();
```

```
addChild(monsterLayer);
addChild(bulletLayer);
```

Затем, когда вы добавляете `Monster` в список отображения, добавьте его в `monsterLayer` , и всякий раз, когда вы добавляете `Bullet` , добавьте `bulletLayer` для достижения желаемого эффекта.

Удалить все объекты из списка отображения

Если вы используете Flash Player 11+, встроенный метод removeChildren - лучший способ удалить всех детей:

```
removeChildren(); //a start and end index can be passed
```

Для устаревших приложений то же самое можно сделать с помощью цикла:

```
while (numChildren > 0) {
    removeChildAt(0);
}
```

Переход от кадров к переключению содержимого вручную

Раньше разработчик Flash использовал фреймы, поскольку они изначально доступны в Flash-проигрывателе, для размещения различных экранов их приложения (чаще всего это игра). В конце концов они могут наткнуться на проблему, что что-то пошло не так, потому что они использовали фреймы и не обратили внимания на трудности, которые возникают из-за этого, и ищут способы как сохранить структуру кадров, так и устранить препятствие использования фреймов с его осложнениями. Решение состоит в том, чтобы использовать классы потомков `Sprite` или экспортировать фреймы в качестве `MovieClip` с одним фреймом (для тех, которые разрабатываются в Adobe Flash CS), и вручную переключать содержимое с помощью `addChild()` и `removeChild()` .

Класс менеджера должен иметь все классы дочерних фреймов, и всякий раз, когда вызывается переход, можно использовать функцию, подобную этой:

```
var frames:Vector.<DisplayObject>; // this holds instances to ALL children
var currentFrame_alt:int; // current frame. Can't use the property
function changeFrame(frame:int):void {
    removeChild(frames[currentFrame_alt]);
    addChild(frames[frame]);
    currentFrame_alt=frame;
}
```

Все дети могут одновременно отправку и слушать события с `Event.ADDED_TO_STAGE` , используемых в качестве точки входа для все , что происходит после того, как `gotoAndStop()` ориентируется этот кадром, и любые исходящие переходы могут быть закодированы как события , основанных на строках, которые прослушивают в `Main` классе, который затем

выполняет переход.

```
frames[0].addEventListener("startGame",startGame); // assuming frame 0 is a "Play" button
function startGame(e:Event):void {
    changeFrame(1); // switch to frame 1 - will display frames[1]
}
```

Конечно, набор строк должен быть предопределен, например, на экране ввода может быть две кнопки для запуска игры: «Начать игру» и «Начать приглушение», например, и кнопки должны отправлять разные события, которые затем будут обрабатываться по-разному в классе менеджера.

Этот шаблон может быть настолько глубоким, насколько вам нужно. Если какой-либо кадр проекта содержит MovieClip с несколькими кадрами, он также может быть разделен на спрайты с помощью этого метода.

Прочитайте Работа с объектами отображения онлайн: https://riptutorial.com/ru/actionscript-3/topic/1628/работа-с-объектами-отображения

глава 19: Работа с событиями

замечания

События представляют собой фрагменты данных, которые программа может создавать, обменивать и реагировать. Асинхронный поток событий отправляется по списку отображения флеш-движком в качестве реакции на внешние события, такие как движения мыши или другой отображаемый кадр. Каждый другой поток событий и вся обработка событий являются синхронными, поэтому, если часть кода генерирует событие, все реакции на него обрабатываются до выполнения следующей строки кода, также если есть несколько слушателей события, все из них будет работать до того, как будет обработано следующее событие.

Существует несколько основных событий, связанных с программированием Flash. `Event.ENTER_FRAME` генерируется до того, как Flash рисует другой фрейм, он сигнализирует, что весь список отображения `Event.ENTER_FRAME` к рисованию и может использоваться как синхронный таймер. `MouseEvent.CLICK` и его братья и сестры могут использоваться для получения ввода от пользователя, а `TouchEvent.TOUCH_TAP` является аналоговым для сенсорных экранов. `KeyboardEvent.KEY_DOWN` и `KEY_UP` предоставляют средства для приема пользовательского ввода с клавиатуры, однако их использование в мобильном отделе практически невозможно из-за устройств, не имеющих физической клавиатуры. Наконец, `Event.ADDED_TO_STAGE` отправляется после того, как экранный объект получает доступ к этапу и входит в глобальный список отображения, который получает все события, которые могут пузыриться вверх и вниз по списку отображения.

Большинство событий во Flash являются специфичными для компонента. Если вы разрабатываете свой собственный компонент, который будет использовать события Flash, используйте класс `flash.events.Event` потомка и его статические свойства `String` для создания набора событий вашего компонента.

Examples

Пользовательские события с данными о событиях

```
package
{
    import flash.events.Event;

    public class CustomEvent extends Event
    {
        public static const START:String = "START";
        public static const STOP:String  = "STOP";

        public var data:*;
```

```
        public function CustomEvent(type:String, data:*,
                                    bubbles:Boolean=false, cancelable:Boolean=false)
        {
            super(type, bubbles, cancelable);

            if (data)
                this.data = data;
        }
    }
}
```

Чтобы отправить настраиваемое событие:

```
var dataObject:Object = {name: "Example Data"};

dispatchEvent(new CustomEvent(CustomEvent.START, dataObject))
```

Для прослушивания пользовательских событий:

```
addEventListener(CustomEvent.STOP, stopHandler);

function stopHandler(event:CustomEvent):void
{
    var dataObject:* = event.data;
}
```

Обработка событий из списка отображения

```
package {
import flash.events.EventDispatcher;

public class AbstractDispatcher extends EventDispatcher {

    public function AbstractDispatcher(target:IEventDispatcher = null) {
        super(target);
    }

}
}
```

Чтобы отправить событие на экземпляр:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.dispatchEvent(new Event(Event.CHANGE));
```

Для прослушивания событий в экземпляре:

```
var dispatcher:AbstractDispatcher = new AbstractDispatcher();
dispatcher.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(event:Event):void
{
```

```
}
```

Обработка основных событий

Вспышка отправляет **`Events`** для большинства своих объектов. Одним из самых основных событий является **`ENTER_FRAME`** , который отправляется (при частоте кадров SWF) для каждого объекта списка отображения.

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

var s:Sprite = new Sprite();
s.addEventListener(Event.ENTER_FRAME, onEnterFrame);

function onEnterFrame(e:Event)
{
    trace("I am called on every frame !");
}
```

Эта функция будет вызываться асинхронно на каждом кадре. Это означает, что функция, которую вы назначаете в качестве `onEnterFrame` события `onEnterFrame` , обрабатывается перед любым другим кодом ActionScript, который привязан к затронутым фреймам.

Добавьте свои собственные события

Вы можете создавать свои собственные события и отправлять их, расширяя класс `Event` .

```
import flash.events.Event;

class MyEvent extends Event
{
    var data: String;

    static public var MY_EVENT_TYPE = "my_event_my_event_code";

    public function MyEvent(type: String, data: String)
    {
        this.data = data;
    }

    override public function clone():Event
    {
        return new MyEvent(type, data);
    }
}
```

Затем вы можете отправить его и прослушать, используя `EventDispatcher` . Обратите внимание, что большинство flash-объектов являются диспетчерами событий.

```
import flash.events.EventDispatcher;

var d = new EventDispatcher();
```

```
d.addEventListener(MyEvent.MY_EVENT_TYPE, onType);

function onType(e: MyEvent)
{
    trace("I have a string: "+e.data);
}

d.dispatchEvent(new MyEvent(MyEvent.MY_EVENT_TYPE, "Hello events!"));
```

Обратите внимание, что метод `clone` требуется, если вы хотите переадресовать свое событие.

Простая структура событий мыши

Благодаря использованию `event types` вы можете легко уменьшить раздувание кода, которое часто возникает при определении событий для многих объектов на сцене, путем фильтрации событий в 1 функции, а не определения многих функций обработки событий.

Представьте, что у нас есть 10 объектов на сцене с именем `object1` , `object2` ... `object10`

Вы можете сделать следующее:

```
var i: int = 1;
while(getChildByName("object"+i) != null){
    var obj = getChildByName("object"+i)
    obj.addEventListener(MouseEvent.CLICK, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OVER, ObjectMouseEventHandler);
    obj.addEventListener(MouseEvent.MOUSE_OUT, ObjectMouseEventHandler);
    obj.alpha = 0.75;
    i++;
}

function ObjectMouseEventHandler(evt:Event)
{
    if(evt.type == "click")
    {
        trace(evt.currentTarget + " has been clicked");
    }
    else
    {
        evt.currentTarget.alpha = evt.type == "mouseOver" ? 1 : 0.75;
    }
}
```

Преимущества этого метода включают:

1. Не нужно указывать количество объектов для применения событий.
2. Не нужно было точно знать, с каким объектом было взаимодействовать, но все же применяют функциональность.
3. Легкое применение событий навалом.

Прочитайте Работа с событиями онлайн: https://riptutorial.com/ru/actionscript-3/topic/1925/работа-с-событиями

глава 20: Работа с таймерами

Examples

Пример таймера обратного отсчета

```
package {
import flash.events.TimerEvent;
import flash.utils.Timer;

public class CountdownTimer extends Timer {

    public var time:Number = 0;

    public function CountdownTimer(time:Number = Number.NEGATIVE_INFINITY, delay:Number =
1000) {
        super(delay, repeatCount);

        if (!isNaN(time))
            this.time = time;

        repeatCount = Math.ceil(time / delay);

        addEventListener(TimerEvent.TIMER, timerHandler);
        addEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

    override public function start():void {
        super.start();
    }

    protected function timerHandler(event:TimerEvent):void {
        time -= delay;
    }

    protected function timerCompleteHandler(event:TimerEvent):void {
    }

    override public function stop():void {
        super.stop();
    }

    public function dispose():void {
        removeEventListener(TimerEvent.TIMER, timerHandler);
        removeEventListener(TimerEvent.TIMER_COMPLETE, timerCompleteHandler);
    }

}
}
```

Этот `CountdownTimer` расширяет `Timer` и используется точно так же, за исключением того, что время отсчитывается.

Пример использования:

```
var timer:CountdownTimer = new CountdownTimer(5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.addEventListener(TimerEvent.TIMER_COMPLETE, completeHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time remaining: " + event.target.time);
}

function completeHandler(event:TimerEvent):void {
    trace("Timer complete");
}
```

Вышеприведенный пример выводит:

```
[trace] Time remaining: 4000
[trace] Time remaining: 3000
[trace] Time remaining: 2000
[trace] Time remaining: 1000
[trace] Time remaining: 0
[trace] Timer complete
```

Интервалы и тайм-ауты

```
import flash.utils.*;
var intervalId:uint=setInterval(schroedingerCat,1000);
// execute a function once per second and gather interval ID
trace("Cat's been closed in the box.");
function schroedingerCat():void {
    if (Math.random()<0.04) {
        clearInterval(intervalId); // stop repeating by ID
        trace("Cat's dead.");
        return;
    }
    trace("Cat's still alive...");
}

var bombId:uint;
function plantBomb(seconds:Number):uint {
    trace("The bomb has been planted, and will blow in "+seconds.toFixed(3)+" seconds!");
    var id:uint=setTimeout(boom,seconds*1000); // parameter is in milliseconds
    return id;
}
function defuseBomb(id:uint):void {
    clearTimeout(id);
    trace("Bomb with id",id,"defused!");
}
function boom():void {
    trace("BOOM!");
}
```

`setInterval()` используется для выполнения повторных задач асинхронно с заданными интервалами. Внутренний объект `Timer` используется, возвращаемое значение типа `uint` - это его внутренний идентификатор, с помощью которого вы можете получить доступ и остановить повторение, вызвав `clearInterval()` . `setTimeout()` и `clearTimeout()` работают

аналогично, но вызов предоставленной функции выполняется только один раз. Вы можете предоставить дополнительные аргументы для обеих функций набора, они будут переданы функции по порядку. Количество аргументов и их тип не проверяются во время компиляции, поэтому вы должны указать странную комбинацию аргументов или функцию, которая их требует, и не получает ее, при этом возникает ошибка «Ошибка № 1063: несоответствие счетчика аргументов».

Вы можете выполнять обе операции `setInterval` и `setTimeout` с регулярными объектами `Timer` , используя либо 0 или 1 для свойства `repeatCount` , либо 0 для неопределенных повторов, 1 для одного.

Пример случайного таймера

```
package {
    import flash.events.TimerEvent;
    import flash.events.TimerEvent;
    import flash.utils.Timer;

    public class RandomTimer extends Timer {

        public var minimumDelay:Number;
        public var maximumDelay:Number;
        private var _count:uint = 0;
        private var _repeatCount:int = 0;

        public function RandomTimer(min:Number, max:Number, repeatCount:int = 0) {
            super(delay, repeatCount);

            minimumDelay = min;
            maximumDelay = max;
            _repeatCount = repeatCount;
        }

        override public function start():void {
            delay = nextDelay();
            addEventListener(TimerEvent.TIMER, timerHandler);
            super.start();
        }

        private function nextDelay():Number {
            return (minimumDelay + (Math.random() * (maximumDelay - minimumDelay)));
        }

        override public function stop():void {
            removeEventListener(TimerEvent.TIMER, timerHandler);
            super.stop();
        }

        protected function timerHandler(event:TimerEvent):void {
            _count++;
            if ((_repeatCount > 0) && (_count >= _repeatCount)) {
                stop();
                dispatchEvent(new TimerEvent(TimerEvent.TIMER_COMPLETE));
            }
            delay = nextDelay();
        }
```

```
        override public function reset():void {
            _count = 0;
            super.reset();
        }
    }
}
```

Этот `RandomTimer` расширяет `Timer` и используется точно так же, за исключением того, что он отправляет случайные интервалы.

Пример использования, диспетчирование в случайном порядке от 1 до 5 секунд:

```
var t:int = getTimer();

var timer:RandomTimer = new RandomTimer(1000, 5000);
timer.addEventListener(TimerEvent.TIMER, timerHandler);
timer.start();

function timerHandler(event:TimerEvent):void {
    trace("Time since last dispatch: " + (getTimer() - t));
    t = getTimer();
}
```

Вышеприведенный пример выводит:

```
[trace] Time since last dispatch: 1374
[trace] Time since last dispatch: 2459
[trace] Time since last dispatch: 3582
[trace] Time since last dispatch: 1335
[trace] Time since last dispatch: 4249
```

Прочитайте Работа с таймерами онлайн: https://riptutorial.com/ru/actionscript-3/topic/2798/работа-с-таймерами

глава 21: Работа с числовыми значениями

Examples

Является ли число четным значением

```
function isEven(n:Number):Boolean {
    return ((n & 1) == 0);
}
```

Примеры:

```
isEven(1); // false
isEven(2); // true

isEven(1.1); // false
isEven(1.2); // false
isEven(2.1); // true
isEven(2.2); // true
```

Является ли число нечетным значением

```
function isOdd(n:Number):Boolean {
    return ((n & 1) == 1);
}
```

Примеры:

```
isOdd(1); // true
isOdd(2); // false

isOdd(1.1); // true
isOdd(1.2); // true
isOdd(2.1); // false
isOdd(2.2); // false
```

Округление до ближайшего X

Чтобы округлить значение до ближайшего кратного x:

```
function roundTo(value:Number, to:Number):Number {
    return Math.round(value / to) * to;
}
```

Пример:

```
roundTo(8, 5); // 10
```

```
roundTo(17, 3); // 18
```

Ошибки округления чисел с плавающей запятой

```
/**
 * @param n Number to be rounded.
 * @param precision Decimal places.
 * @return Rounded Number
 */
function roundDecimal(n:Number, precision:Number):Number {
    var factor:int = Math.pow(10, precision);
    return (Math.round(n * factor) / factor);
}
```

Примеры:

```
trace(0.9 - 1); // -0.09999999999999998

trace(roundDecimal(0.9 - 1, 1));     // -0.1
trace(roundDecimal(0.9 - 1, 2));     // -0.1

trace(roundDecimal(0.9 - 1.123, 1)); // -0.2
trace(roundDecimal(0.9 - 1.123, 2)); // -0.22
trace(roundDecimal(0.9 - 1.123, 3)); // -0.223
```

Отображение номеров с требуемой точностью

```
var a:Number=0.123456789;
trace(a);                    // 0.123456789
trace(a.toPrecision(4));     // 0.1235
trace(a.toFixed(4));         // 0.1235
trace(a.toExponential(4));   // 1.2345e-1
trace(a.toString(16));       // 0 - works for integer part only
var b:Number=12345678.9876543; // a bigger number to display rounding
trace(b);                    // 12345678.9876543
trace(b.toPrecision(4));     // 1.235e+7
trace(b.toFixed(4));         // 12345678.9877
trace(b.toExponential(4));   // 1.2345e+7
trace(b.toString(16));       // bc614e
b=1.0e+16;
trace(b.toString(36));       // 2qgpckvng1s
```

Прочитайте Работа с числовыми значениями онлайн: https://riptutorial.com/ru/actionscript-3/topic/1899/работа-с-числовыми-значениями

глава 22: Работа со звуком

Синтаксис

- Sound.play (startTime: Number = 0, loop: int = 0, sndTransform: flash.media: SoundTransform = null): SoundChannel // Воспроизводит загруженный звук, возвращает SoundChannel

Examples

Остановить воспроизведение звука

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.events.Event;

var snd:Sound; = new Sound();
var sndChannel:SoundChannel
var sndTimer:Timer;

snd.addEventListener(Event.COMPLETE, soundLoaded);
snd.load(new URLRequest("soundFile.mp3")); //load after adding the complete event

function soundLoaded(e:Event):void
{
    sndChannel = snd.play();

    //Create a timer to wait 1 second
    sndTimer = new Timer(1000, 1);
    sndTimer.addEventListener(TimerEvent.TIMER, stopSound, false, 0, true);
    sndTimer.start();
}

function stopSound(e:Event = null):void {
    sndChannel.stop(); //Stop the sound
}
```

Бесконечная петля звука

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("filename.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play(0, int.MAX_VALUE); // There is no way to put "infinite"
}
```

Вам также не нужно ждать загрузки звука перед вызовом функции `play()` . Таким образом, это будет выполнять ту же работу:

```
snd = new Sound(new URLRequest("filename.mp3"));
snd.play(0, int.MAX_VALUE);
```

И если вы действительно хотите по какой-то причине `int.MAX_VALUE` звук inifinite time (`int.MAX_VALUE` будет зацикливаться на 1-м звуке около 68 лет, не считая паузы, которую вызывает mp3), вы можете написать примерно так:

```
var st:SoundChannel = snd.play();
st.addEventListener(Event.SOUND_COMPLETE, repeat);
function repeat(e:Event) {
    st.removeEventListener(Event.SOUND_COMPLETE, repeat);
    (st = snd.play()).addEventListener(Event.SOUND_COMPLETE, repeat);
}
```

Функция `play()` возвращает новый экземпляр объекта `SoundChannel` каждый раз, когда он `SoundChannel` . Мы передаем его переменной и прослушиваем его событие SOUND_COMPLETE. В случае обратного вызова слушатель удаляется из текущего объекта `SoundChannel` а новый создается для нового объекта `SoundChannel` .

Загружать и воспроизводить внешний звук

```
import flash.net.URLRequest;
import flash.media.Sound;
import flash.events.Event;

var req:URLRequest = new URLRequest("click.mp3");
var snd:Sound = new Sound(req);

snd.addEventListener(Event.COMPLETE, function(e: Event)
{
    snd.play();
}
```

Прочитайте Работа со звуком онлайн: https://riptutorial.com/ru/actionscript-3/topic/2043/работа-со-звуком

глава 23: Рисование битовых карт

Examples

Нарисуйте экранный объект в растровые данные

Вспомогательная функция для создания растровой копии объекта. Это можно использовать для преобразования векторных объектов, текста или сложного вложенного Sprite в плоское растровое изображение.

```
function makeBitmapCopy(displayObj:IBitmapDrawable, transparent:Boolean = false, bgColor:uint
= 0x00000000, smooth:Boolean = true):Bitmap {

    //create an empty bitmap data that matches the width and height of the object you wish to
draw
    var bmd:BitmapData = new BitmapData(displayObj.width, displayObj.height, transparent,
bgColor);

    //draw the object to the bitmap data
    bmd.draw(displayObj, null, null, null, null, smooth);

    //assign that bitmap data to a bitmap object
    var bmp:Bitmap = new Bitmap(bmd, "auto", smooth);

    return bmp;
}
```

Использование:

```
var txt:TextField = new TextField();
txt.text = "Hello There";

var bitmap:Bitmap = makeBitmapCopy(txt, true); //second param true to keep transparency
addChild(bitmap);
```

Нарисуйте экранный объект с любыми координатами точки регистрации

```
    public function drawDisplayObjectUsingBounds(source:DisplayObject):BitmapData {
        var bitmapData:BitmapData;//declare a BitmapData
        var bounds:Rectangle = source.getBounds(source);//get the source object actual size
        //round bounds to integer pixel values (to aviod 1px stripes left off)
        bounds = new Rectangle(Math.floor(bounds.x), Math.floor(bounds.y),
Math.ceil(bounds.width), Math.ceil(bounds.height));

        //to avoid Invalid BitmapData error which occures if width or height is 0
        //(ArgumentError: Error #2015)
        if((bounds.width>0) && (bounds.height>0)){
            //create a BitmapData
            bitmapData = new BitmapData(bounds.width, bounds.height, true, 0x00000000);

            var matrix:Matrix = new Matrix();//create a transform matrix
```

```
            //translate if to fit the upper-left corner of the source
            matrix.translate(-bounds.x, -bounds.y);
            bitmapData.draw(source, matrix);//draw the source
            return bitmapData;//return the result (exit point)
        }
        //if no result is created - return an empty BitmapData
        return new BitmapData(1, 1, true, 0x00000000);
    }
```

Замечание: для `getBounds()` для возврата допустимых значений объект должен иметь доступ к сцене хотя бы один раз, в противном случае значения являются фиктивными. Код может быть добавлен, чтобы гарантировать, что переданный `source` имеет этап, а если нет, его можно добавить на сцену, а затем удалить снова.

Анимация листа спрайтов

Лист спрайта по определению представляет собой растровое изображение, содержащее определенную анимацию. В старых играх используется лист спрайтов с типом сетки, т. Е. Каждый кадр занимает равную область, а кадры выравниваются по краям для формирования прямоугольника, возможно, с некоторыми незанятыми пространствами. Позже, чтобы минимизировать размер растрового изображения, листы спрайтов начинают «упаковываться», удаляя лишние пробелы вокруг прямоугольника, содержащего каждый кадр, но все же каждый кадр является прямоугольником, упрощающим операции копирования.

Чтобы оживить лист спрайта, можно использовать два метода. Во-первых, вы можете использовать `BitmapData.copyPixels()` чтобы скопировать определенный регион вашего листа спрайтов в отображаемый `Bitmap` , создавая анимированный символ. Этот подход лучше, если вы используете один отображаемый `Bitmap` рисунок, на котором размещается весь рисунок.

```
var spriteSheet:BitmapData;
var frames:Vector.<Rectangle>; // regions of spriteSheet that represent frames
function displayFrameAt(frame:int,buffer:BitmapData,position:Point):void {
    buffer.copyPixels(spriteSheet,frames[frame],position,null,null,true);
}
```

Второй метод можно использовать , если у вас есть много `Sprite` s или `Bitmap` s в списке отображения, и они имеют тот же спрайт лист. Здесь целевой буфер больше не является единственным объектом, но каждый объект имеет свой собственный буфер, поэтому правильная стратегия заключается в том, чтобы манипулировать свойством `bitmapData` растровых изображений. Однако перед тем, как это сделать, лист спрайта следует разделить на отдельные рамы.

```
public class Stuff extends Bitmap {
    static var spriteSheet:Vector.<BitmapData>;
    function displayFrame(frame:int) {
        this.bitmapData=spriteSheet[frame];
```

```
    }
    // ...
}
```

Прочитайте Рисование битовых карт онлайн: https://riptutorial.com/ru/actionscript-3/topic/2814/рисование-битовых-карт

глава 24: Типы

Examples

Литье под давлением

Типовое литье выполняется либо оператором `as` :

```
var chair:Chair = furniture as Chair;
```

Или, обернув значение в `Type()` :

```
var chair:Chair = Chair(furniture);
```

Если бросок терпит неудачу с , `as` результат этого броска `null` . Если сбой завершается путем переноса в `Type()` , `TypeError` .

Тип функции

Функции имеют тип `Function` :

```
function example():void { }
trace(example is Function); // true
```

На них могут ссылаться другие переменные с типом `Function` :

```
var ref:Function = example;
ref(); // ref.call(), ref.apply(), etc.
```

И они могут быть переданы как аргументы для параметров, тип которых - `Function` :

```
function test(callback:Function):void {
    callback();
}

test(function() {
    trace('It works!');
}); // Output: It works!
```

Тип класса

Написаны ссылки на объявления `Class` :

```
var spriteClass:Class = Sprite;
```

Вы можете использовать переменные типизированного `Class` для создания экземпляров этого класса:

```
var sprite:Sprite = new spriteClass();
```

Это может быть полезно для передачи аргумента типа `Class` функции, которая может создать и экземпляр предоставленного класса:

```
function create(type:Class, x:int, y:int):* {
    var thing:* = new type();

    thing.x = x;
    thing.y = y;

    return thing;
}

var sprite:Sprite = create(Sprite, 100, 100);
```

Аннотирующие типы

Вы можете указать компилятору тип значения, аннотируя его с помощью `:Type` :

```
var value:int = 10; // A property "value" of type "int".
```

Функциональные параметры и типы возврата также могут быть аннотированы:

```
// This function accepts two ints and returns an int.
function sum(a:int, b:int):int {
    return a + b;
}
```

Попытка присвоить значение с несоответствующим типом приведет к типу `TypeError` :

```
var sprite:Sprite = 10; // 10 is not a Sprite.
```

Проверка типов

Вы можете использовать оператор `is` для проверки того, имеет ли значение определенный тип:

```
var sprite:Sprite = new Sprite();

trace(sprite is Sprite); // true
trace(sprite is DisplayObject); // true, Sprite inherits DisplayObject
trace(sprite is IBitmapDrawable); // true, DisplayObject implements IBitmapDrawable
trace(sprite is Number); // false
trace(sprite is Bitmap); // false, Bitmap inherits DisplayObject
                         // but is not inherited by Sprite.
```

Существует также `instanceof` оператор (устаревшее) , который работает почти идентично `is` исключением того, что она *возвращает `false` при проверке реализованных интерфейсов* и типов INT / UINT.

, `as` оператор может также использоваться так же , как `is` оператор. Это особенно полезно, если вы используете некоторую интеллектуальную среду IDE, такую как FlashDevelop, которая предоставит вам список всех возможных свойств явного типа объекта. Пример:

```
for (var i:int = 0; i < a.length; i++){
    var d:DisplayObject = a[i] as DisplayObject;
    if (!d) continue;
    d.//get hints here
    stage.addChild(d);
}
```

Для того, чтобы получить тот же эффект с `is` можно было бы написать (sligthly менее удобно):

```
for (var i:int = 0; i < a.length; i++){
    if (a[i] is DisplayObject != true) continue;
    var d:DisplayObject = a[i] as DisplayObject;
    stage.addChild(d);
}
```

Просто имейте в виду , что при проверке coditions с `as` оператора, данное значение будет преобразовано в кулак указанного типа , а затем в результате этой операции будет проверяться , если не ложь, так что будьте осторожны при использовании его с возможными значениями ложных / NaN:

```
if(false as Boolean) trace("This will not be executed");
if(false as Boolean != null) trace("But this will be");
```

Ниже таблицы показаны некоторые базовые значения и типы с результатом операторов типа. Зеленые ячейки будут оценивать значение true, красный - false, a greay вызовет ошибки компиляции / времени выполнения.

		Sprite	IBitmapDrawable	Object	Class	uint	int
new Sprite()	as	[object Sprite]	[object Sprite]	[object Sprite]	null	null	null
	is	true	true	true	false	false	false
	instanceof	true	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
true	as	null	null	true	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
false	as	null	null	false	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
0.3	as	null	null	0.3	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
1	as	null	null	1	null	1	1
	is	false	false	true	false	true	true
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
-1	as	null	null	-1	null	null	-1
	is	false	false	true	false	false	true
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
NaN	as	null	null	NaN	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
"string"	as	null	null	string	null	null	null
	is	false	false	true	false	false	false
	instanceof	false	false	true	false	false	false
	(<columnName>)	error	error	error	error	error	error
Boolean	as	null	null	[class Boolean]	[class Boolean]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	true	true	true	true	true	true
String	as	null	null	[class String]	[class String]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
Number	as	null	null	[class Number]	[class Number]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	NaN	NaN	NaN	NaN	NaN	NaN
int	as	null	null	[class int]	[class int]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	0	0	0	0	0	0
uint	as	null	null	[class uint]	[class uint]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	0	0	0	0	0	0
Class	as	null	null	[class Class]	[class Class]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
Object	as	null	null	[class Object]	[class Object]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	[class Sprite]	[class IBitmapDrawable]	[class Object]	[class Class]	[class uint]	[class i
IBitmapDrawable	as	null	null	[class IBitmapDrawable]	[class IBitmapDrawable]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	error	error	error	error	error	error
Sprite	as	null	null	[class Sprite]	[class Sprite]	null	null
	is	false	false	true	true	false	false
	instanceof	false	false	true	true	false	false
	(<columnName>)	error	error	error	error	error	error

поэтому следует, что нет способа определить типизированный массив как `Array<T>` . Однако существует специальный класс `Vector.<T>` который работает аналогичным образом, за исключением того, что вы *должны указывать конкретный класс* при создании экземпляра вектора. Это означает, что нет возможности создавать абстракции поверх типа `Vector.<T>` (например, расширять его и добавлять новые функции), что является огромным недостатком.

Более простой способ взглянуть на это состоит в том, что каждый класс, который вы определяете автоматически, имел сопутствующий класс с именем `Vector.<NameOfYourClass>` .

С учетом сказанного, все еще существуют огромные преимущества для типа `Vector.<T>` для обычного массива:

- Увеличивается производительность при работе с `Vector.<T>` vs массивы [1] .
- Вы получаете тип `TypeError` типа компиляции, если вы пытаетесь вставить значения non-T в коллекцию.
- IDE предоставляют полезную информацию о подсказке типа для объектов внутри экземпляра `Vector.<T>` .

Примеры создания `Vector.<T>` :

```
var strings:Vector.<String> = new Vector.<String>(); // or
var numbers:Vector.<Number> = new <Number>[];
```

---

[1] Векторы фактически обеспечивают заметные улучшения производительности по сравнению с массивами при работе с примитивными типами (`String` , `int` , `uint` , `Number` и т. Д.).

Прочитайте Типы онлайн: https://riptutorial.com/ru/actionscript-3/topic/2803/типы

глава 25: Шаблон Singleton

замечания

Шаблон singleton имеет целью разрешить существование только одного экземпляра класса в любой момент времени.

Предотвращение прямого создания экземпляра через конструктор обычно предотвращается, делая его закрытым. Однако это невозможно в As3, и поэтому необходимо использовать другие способы управления количеством экземпляров.

Examples

Синглтон-исполнитель через частную инстанцию

В этом подходе доступ к одному осуществляется через статический метод:

```
Singleton.getInstance();
```

Чтобы принудительно применять только один экземпляр синглтона, частная статическая переменная сохраняет экземпляр, а любые дополнительные попытки создать экземпляр экземпляра выполняются внутри конструктора.

```
package {

public class Singleton {

    /** Singleton instance */
    private static var _instance: Singleton = new Singleton();

    /** Return singleton instance. */
    public static function getInstance():Singleton {
        return _instance;
    }

    /** Constructor as singleton enforcer. */
    public function Singleton() {
        if (_instance)
            throw new Error("Singleton is a singleton and can only be accessed through
Singleton.getInstance()");
    }

}
}
```

Прочитайте Шаблон Singleton онлайн: https://riptutorial.com/ru/actionscript-3/topic/1437/шаблон-singleton

кредиты

S. No	Главы	Contributors
1	Начало работы с ActionScript 3	BadFeelingAboutThis, Community, Jason Sturges, joshtynjala, Kit Grose, null, Programmer Dancuk, Vesper
2	Генерация случайных величин	alebianco, BadFeelingAboutThis, HITMAN, Jason Sturges, Marty, mnoronha, null, Vesper, xims
3	Двоичные данные	Paweł Audionysos
4	Жизненный цикл списка отображения	Jason Sturges, mnoronha
5	Загрузка внешних файлов	BadFeelingAboutThis, Marty
6	Использование прокси-класса	Organis
7	Манипуляция и фильтрация растровых изображений	payam_sbr, VC.One
8	Объектно-ориентированное программирование	HITMAN, Marty, null, www0z0k
9	Оптимизация производительности	Community, Marty
10	Основы разработки игр	payam_sbr
11	Отзывчивый дизайн приложения	BadFeelingAboutThis, null, Vesper
12	Отправка и получение данных с серверов	Marty, null, xims
13	Понимание «Ошибка	Vesper, www0z0k

	1009: невозможно получить доступ к свойству или методу ссылки на нулевой объект»	
14	Работа с видео	VC.One, www0z0k
15	Работа с временной шкалой	Marty
16	Работа с геометрией	Marty, mnoronha, www0z0k
17	Работа с датой и временем	Jason Sturges, mnoronha
18	Работа с объектами отображения	BadFeelingAboutThis, Jason Sturges, Vesper
19	Работа с событиями	blue112, Jason Sturges, mnoronha, null, VC.One, Vesper, Zze
20	Работа с таймерами	Jason Sturges, mnoronha, Vesper, www0z0k
21	Работа с числовыми значениями	Jason Sturges, Jonny Henly, Marty, Vesper
22	Работа со звуком	BadFeelingAboutThis, blue112, Paweł Audionysos, VC.One
23	Рисование битовых карт	BadFeelingAboutThis, Marty, Vesper, www0z0k
24	Типы	BadFeelingAboutThis, joshtynjala, Marty, Paweł Audionysos
25	Шаблон Singleton	commovere, Jason Sturges, mnoronha, null