ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 62 за 2012 г.

Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н.

Реализация программных средств, обеспечивающих управление доставкой видео и аудио данных на базе HTML5 и Flash

Рекомендуемая форма библиографической ссылки: Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н. Реализация программных средств, обеспечивающих управление доставкой видео и аудио данных на базе HTML5 и Flash // Препринты ИПМ им. М.В.Келдыша. 2012. № 62. 15 с. URL: http://library.keldysh.ru/preprint.asp?id=2012-62

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Ю.А.Бухштаб, А.А.Воробьев, Н.Н.Евтеева

Реализация программных средств, обеспечивающих управление доставкой видео и аудио данных на базе HTML5 и Flash

Москва — 2012

Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н.

Реализация программных средств, обеспечивающих управление доставкой видео и аудио данных на базе HTML5 и Flash

В работе рассматривается реализация программных средств, обеспечивающих полнофункциональное виртуальное редактирование мультимедийной потоковой информацией, транслируемой с территориально разнесенных серверов и представленной в различных форматах. Эти программные средства способны функционировать в средах всех основных браузеров, автоматически определяя формат очередного поступающего потока и поддерживая воспроизведение результирующей видео и аудио последовательности без задержек между фрагментами.

Ключевые слова: мультимедиа, потоковое видео, HTML5, Flash, SMIL

Bukhshtab Yury Alexandrovich, Vorobiov Andrey Arturovich, Evteeva Natalia Nikolaevna

Development of software tools providing the control of video and audio data delivery on the base of HTML5 and Flash

This paper describes the implementation of software tools that provide a complete virtual editing multimedia streaming broadcast information from geographically dispersed servers and presented in various formats. This software is able to operate in environments of all major browsers and automatically identify the format of the next incoming flow and supporting the playback of the resulting video and audio sequences without delay between the fragments.

Key words: multimedia, streaming video, HTML5, Flash, SMIL

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 11-07-00258-а.

В статье описываются результаты продолжения работ по созданию принципов построения и реализации программных средств управления потоковыми видео и аудио данными, размещенными в различных доменах телекоммуникационной сети, поддерживающих возможность полнофункционального виртуального редактирования этих данных. В ходе этих работ была поставлена задача разработать универсальный программный модуль, обеспечивающий создание единого виртуального видео и аудио контента из потоков, не только транслируемых с территориально разнесенных серверов, но и представленных в различных форматах [1]. При этом такой модуль должен быть способен функционировать в средах разных браузеров, автоматически определяя формат очередного поступающего потока и поддерживая воспроизведение видео и/или аудио последовательности как одного медиа объекта, без задержек между составляющими его фрагментами.

Используемые методы управления потоковыми видео и аудио данными

Разрабатываемые программные средства позволяют создать медиа контент, воспроизводимый специальным программным модулем (в дальнейшем будем называть его плеером), и представляющий из себя результат виртуального склеивания, удаления, выделения и переупорядочивания видео и аудио файлов и их фрагментов. Исходные видео и аудио файлы могут быть размещены на различных серверах, иметь различный формат (например, flv, mp4, mp3, webm, ogg), но воспроизводиться плеером непрерывно, как единое целое.

Фактически, результатом работы является некоторое описание в виде структур перекрестных ссылок, определяющих связи и временные соотношения между исходными и виртуальными потоками и их фрагментами и определяющих последовательности воспроизводимых фрагментов, с указанием их URL, тайм-кодов начала и конца и, возможно, формата. Это описание хранится на сервере или генерируется динамически серверной программой в виде управляющего файла и предоставляется плееру для воспроизведения.

Для задания подобных последовательностей обычно используется язык SMIL (Synchronized Multimedia Integration Language) [2]. SMIL - это язык основанный на XML и в нем заложено больше возможностей, чем только описание воспроизводимой последовательности потоковых фрагментов, но некоторое его подмножество используются именно для этих целей, например, в плеерах RealPlayer и QuickTime. Однако не все потоковые технологии поддерживают язык SMIL или имеют аналогичные возможности синхронизации данных. Поскольку SMIL не обеспечивает поддержку синхронизации потоковых данных в современных средах HTML5 [3] и Flash [4], возникла необходимость разработать специальную структуру описания последовательности воспроизводимых фрагментов потоковых данных,

подобную той, которая используется в языке SMIL, и обеспечить интерпретацию такой структуры в этих средах.

Описания на языке XML могут использоваться и в WEB приложениях. XML файлы могут быть считаны и разобраны программами на JavaScript, поэтому можно разработать формат задания последовательности фрагментов для воспроизведения плеером в виде описания на XML. В этом формате необходимо предусмотреть средства задания фрагментов, с указанием URL медиа-файла, тайм-кодов, возможность наложения аудио фрагмента на видео фрагмент и, возможно, некоторые дополнительные указания плееру.

Примерное описание воспроизводимой последовательности может выглядеть так:

```
<SMIL>
<LAYOUT  width="480" height="360" />
<SEQ>

<VIDEO src="http://serv1.com/content/video1.mp4" timebeg="5:30" timeend="8:46" />

<PAR>
<VIDEO src="http://serv2.com/lecture/proff.webm" timebeg="33.6" timeend="22:15" />
<AUDIO src="http://ourserv.ru/lecture/trasl-proff.ogg" timebeg="15.2"/>
</PAR>

<VIDEO src="http://serv1.com/content/video1.mp4" timebeg="8:46.1" timeend="10:21.5"/>

</SEQ>
</SMIL>
```

Здесь показана последовательность из трех видеофрагментов. Фрагменты задаются тегом <VIDEO> с указанием их URL (параметр src) и тайм-кодов начала и конца (параметры timebeg и timeend). Кроме того, на второй видеофрагмент наложен звук из другого источника, заданный тегом <AUDIO>. Пара <VIDEO> и <AUDIO> вложена в тег <PAR>, указывающий на их параллельное воспроизведение и синхронизацию. Дополнительно тегом <LAYOUT > задан рекомендуемый размер экрана видеоплеера.

Такое описание может быть статически размещено на WEB странице или динамически считано с сервера и передано плееру. Для динамического считывания XML файлов в JavaScript имеются специальные средства. Для этого используется встроенный объект XMLHttpRequest, с помощью которого

осуществляется считывание XML документа. На самом деле можно считывать произвольные текстовые файлы, но при считывании XML документов можно использовать ряд дополнительных возможностей. Самая простая процедура считывания выглядит так:

```
//Создается объект XMLHttpRequest
var xReq = new XMLHttpRequest();
//Указывается метод, URL считываемого файла, третий параметр false
//указывает на синхронное считывание
xReq.open('GET',  "http://ourserv.ru/lecture/lect1.xml", false);
//Посылаем запрос серверу
xReq.sent();
```

После этого в свойствах объекта имеем:

xReq.responseText – считанный документ в виде строки;

xReq.responseXML – ссылка на древовидную структуру, в которой содержится разобранный XML документ.

Таким способом можно считать небольшой, статически размещенный на сервере документ. Но у этого объекта имеются и дополнительные возможности, такие как асинхронное считывание (программа не ждет, пока документ считается полностью), задание дополнительных поисковых параметров для сервера, если документ генерируется серверной программой.

Объект XMLHttpRequest имеется в браузерах FireFox, Opera, Chrome. В IE необходимо использовать специальный ActiveX объект, но его возможности и результат такие же.

Однако использование этих средств имеет существенный недостаток: нет возможности осуществлять междоменное считывание документов. То есть, если WEB страница, которая задает считывание документа, расположена, например, по адресу http://mydomen.ru/page.html, то она может считывать XML документы только с этого же домена или его поддоменов (например, можно считывать документы с адресами http://mydomen.ru/list1.xml или http://subd.mydomen.ru/list2.xml), но нельзя считать документы с какого-то другого сервера (например, с сервера по адресу http://youdomen.ru/list.xml).

Это обстоятельство накладывает существенное ограничение. Например, мы создаем описание некоторого медиа-контента и хотим разрешить кому-то воспроизводить его на своей WEB странице. Мы можем дать разрешение использовать наш плеер, так как он является программой на JavaScript, и любая страница может обращаться к этой программе без ограничений. Но если мы разместим SMIL файл с описанием транслируемого контента на нашем сервере, то никакая страница, загружаемая с другого сервера, не сможет считать этот файл. В случае статического SMIL файла мы можем передать его для размещения на другом сервере, но в случае, если этот документ генерируется

динамически программой на нашем сервере, то такая возможность полностью исключена.

Существуют некоторые обходные пути решения этой проблемы. Например, использование прокси-серверов. Но этот метод требует дополнительных настроек как сервера, на котором размещен SMIL документ, так и сервера, на котором размещена WEB страница. Это не всегда возможно.

Другой возможностью является использование фреймов. При этом на WEB странице размещается не сам плеер, а фрейм (задаваемый элементом <iframe>). В этот фрейм с нашего сервера считывается документ, который содержит плеер. При таком использование плеер может считать SMIL файл с нашего сервера. Но у WEB страницы возникают ограничения на взаимодействия с плеером. Например, затруднена возможность изменения размеров экрана, у разработчика WEB страницы нет возможности изменить дизайн элементов управления плеером (в частности, поменять внешний вид кнопок).

Решением этой проблемы может быть использование языка JavaScript для описания воспроизводимой последовательности фрагментов. Код на JavaScript может быть считан с любого сервера (нет ограничений на считывание между разными доменами).

Обычно программы на JavaScript задаются на WEB страницах статически, с помощью элементов вида:

<script src="http://serv.com/prog.js" />

Однако, все современные браузеры позволяют динамическое создание, считывание и исполнение программ на JavaScript. Для этого используются методы DHTML.

var os=createElement("script"); – создание элемента scrip

os.src="http://serv.com/prog.js"; – задание адреса, с которого должен быть считан код

document.getElementByTag("head").appendChild(os) – подключение элемента к странице. В этот момент код программы на JavaScript считывается и исполняется.

Возможно, недостатком такого метода является то, что так можно считать только программу на языке JavaScript, которая немедленно исполняется. Считается, что такое использование может представлять опасность, если программа на сервере будет подменена неким злоумышленником, и приведет к считыванию вредоносного кода, вирусов или другого, нежелательного содержимого. Впрочем, представляется, что такая подмена не более вероятна чем подмены статически заданных JavaScript файлов или html страниц.)

Вообще говоря, описание последовательностей фрагментов на операторных языках, к которым относится JavaScript, может быть не очень удобным. Но в JavaScript есть возможность задания структурированных данных с использованием достаточно простой и наглядной нотации JSON (JavaScript Object Notation).

Фактически, описание последовательности фрагментов можно представить в виде некоторой древовидной структуры вложенных объектов и массивов. Описание элемента VIDEO на XML это описание структуры, имеющей поля src, timebeg, timeend. Тегом <PAR> задается структура, состоящая из подструктур VIDEO и AUDIO. А тег <SEQ> описывает массив структур, содержащих описания фрагментов.

В JSON также возможно описывать структуру массивов. При этом описание массива имеет вид:

[элемент0, элемент1, элемент2…]

Где элемент i – элемент массива, который может быть элементарным типом (число, строка), структурой или другим массивом.

Описание структуры имеет вид:

{поле1:значение, поле2: значение….},

где поле i – имя поля структуры, а значение – значение поля, которое может быть элементарного типа, другой структурой или массивом.

В принципе, структуры правильнее было бы называть объектами JavaScript. От структур в таких языках как C или Java они отличаются тем, что не имеют заранее предопределенного списка полей, тип полей так же не фиксируется, и в любой момент к объекту можно добавить новые поля.

Приведенный выше пример описания на XML в нотации JSON будет выглядеть так:

```
Var pls= {
layout: {width:480, height:360},
seq:[
  {video:{src:"http://serv1.com/content/video1.mp4",timebeg:"5:30",
timeend:"8:46"}},
  {
    video:{src:"http://serv2.com/lecture/proff.webm",
timebeg:"33.6",timeend:"22:15"},
    audio:{src:"http://ourserv.ru/lecture/trasl-proff.ogg", timebeg:"15.2"}
  },
  {video:{src:"  http://serv1.com/content/video1.mp4",  timebeg:"8:46.1",
timeend:"10:21.5"}}
]
```

```
};
```

Это исполняемый код программы на JavaScript, который присваивает переменной pls структуру, состоящую из полей layout и seq. Поле layout – структура из полей width и height. А поле seq – массив из трех элементов. В свою очередь каждый элемент массива является структурой, содержащей поле video и поле audio. Эти поля сами являются структурами, содержащими поля src, timebeg и timeend. После считывания и исполнения этого кода, на WEB странице появится JavaScript переменная pls, которая может быть использована плеером для получения информации о воспроизводимых фрагментах. По своему содержанию эта информация полностью аналогична той, которая содержится в свойстве responseXML объекта XMLHttpRequest.

Использование нотации JSON может быть непривычным и не очень удобным для задания и корректировки информации вручную, но если информация о воспроизводимых фрагментах генерируется динамически серверной программой, то такая проблема не возникает.

Реализация воспроизведения потоковой информации в среде HTML5

В ходе проводимых работ была реализована пилотная версия плеера, позволяющего воспроизводить последовательность видео и аудио потоков или их фрагментов, транслируемых с распределенных серверов, как единый видеофайл. Кроме этого плеер способен воспроизводить дополнительный аудиофайл параллельно с видеофайлом, подменяя звук из этого видеофайла. При разработке плеера предполагалось, что он должен воспроизводить файлы различных форматов, наиболее часто используемых в сети Интернет, передаваемых потоком по протоколу HTTP.

Этот плеер может быть размещен на WEB странице и способен работать в браузерах, поддерживающих стандарт HTML5. Текущая реализация разработана для браузеров с версиями не ниже FireFox 4, Opera 10.6, Chrome 6, Internet Explorer 9.

Разработанный плеер способен воспроизводить видеофайлы форматов WebM, Mpeg4 (H.264), OGG и FLV и аудиофайлы форматов OGG (Vorbis) и MP3. Видеоклипы и фрагменты в последовательности могут передаваться из различных источников (с разных серверов).

В процессе своей работы плеер воспроизводит информацию средствами, имеющимися в браузере, а если их недостаточно, то пытается использовать дополнительные программы, имеющиеся в программной среде. Так форматы WebM и OGG будут воспроизводиться с помощью браузеров FireFox, Opera и Chrome, средствами, имеющимися в HTML5, а форматы Mpeg4 или MP3 будут воспроизводиться только в Chrome, а в FireFox и Opera плеер будет пытаться использовать дополнительные возможности, имеющиеся в системе, такие как плагин Flash. При использовании Flash плеер создаст соответствующий элемент

в том же месте экрана, где расположено его основное окно, и воспроизведение будет идти через этот, дополнительный элемент. Такое переключение между элементами при переходе с фрагмента на фрагмент происходит практически незаметно для пользователя.

При наложении дополнительного звукового файла плеер создает невидимый элемент (элемент AUDIO в HTML5, плагин Flash) через который передается аудиопоток. Звук из основного видеопотока при этом заглушается.

Встраивание плеера на WEB страницу производится с помощью HTML и Javascript. Сам плеер написан на Javascript, поэтому прежде всего необходимо подключить файлы с кодом плеера.

```
<script src="vplayer.js" type="text/javascript" ></script>
```

На самой странице должен быть создан какой-либо элемент, в котором будет размещаться экран плеера:

```
<style type="text/css" media="screen">
#video {position: relative; width: 480px; height: 360px; background-color: #000;}
</style>
<div id="video"></div>
```

Здесь создан пустой элемент DIV. У него обязательно должен быть задан идентификатор, в данном случае "video". Кроме того, средствами CSS задан начальный размер элемента (ширина 480, высота 360) и черный цвет. Задание начальных размеров элемента не является обязательным, так как плеер сам установит необходимый размер, но желательно, чтобы браузер сразу произвел разметку страницы на этапе загрузки.

Необходимо создать объект плеера. Удобнее всего это сделать после загрузки страницы в функции, которая вызывается в качестве обработчика загрузки. При создании объекта плееру передается идентификатор элемента, в котором будет размещен экран, а также ширина и высота экрана. Эта информация передается в виде структуры (объекта JavaScript). Код создания объекта плеера может быть такой:

```
<script>
var vpl;
function loadpg() {

  vpl=new VPlayer({vpid: "video", width: 480, height: 360, funState: vpstate});

}
</script>
```

<body onLoad="loadpg()">

Здесь переменная vpl будет содержать ссылку на созданный объект плеера, функция loadpg вызывается в качестве обработчика завершения загрузки страницы (указано в элементе BODY). Объект плеера создается конструктором VPlayer, которому передается структура со следующими полями:

vpid – идентификатор элемента, в котором будет размещен экран,

width – ширина экрана,

height – высота экрана,

funState – ссылка на функцию обратного вызова. Эту функцию будет вызывать плеер, чтобы информировать внешнее окружение о своем состоянии.

Для начала работы плеера ему необходимо задать источник получения видео и аудио информации. Плееру передается последовательность фрагментов воспроизводимых файлов. Эта последовательность может быть задана различными способами: непосредственно на странице в виде структуры JavaScript или в виде считываемых с сервера файлов. Файлы задаются в виде URL адреса и могут быть двух видов: в виде описаний структуры на JavaScript в нотации JSON или XML файлы с описанием последовательности на SMIL-подобном языке.

Для передачи последовательности фрагментов используются следующие функции:

vpl.OpenS (pllist) – задание последовательности в виде структуры,

pllist – ссылка на структуру,

vpl.OpenJson (url) – задание в виде файла на JavaScript. url – адрес файла,

vpl.OpenXML (url)) – задание в виде файла на XML. url – адрес файла.

Задание последовательности фрагментов на JavaScript в виде структуры имеет следующий вид:

```
var pllist= {
seq:
[
{
  video:{src:"http://serv1.ru/clip1.webm", timeBeg:10, timeEnd:30}
},
{

  video:{src:"http://serv2.ru/clip2.webm", timeBeg:200, timeEnd:300},
  audio:{src:"http://serv1.ru /voice1.ogg", timeBeg:20}
},
{
  video:{src:"http://serv1.ru/clip1.webm", timeBeg:30, timeEnd:180},
```

```
  audio:{src:"http://serv3.ru/voice2.webm", timeBeg:60}
}
]
};
```

Здесь задается объект pllist. Поле seq этого объекта задает последовательность воспроизводимых фрагментов в виде массива. Каждый элемент массива представляет из себя структуру с полями video и audio. Поле video задает видеофрагмент, а поле audio, если оно задано, аудиофайл накладываемый на видео. Видео и аудио файлы также задаются в виде структур с полями src – URL файла, timeBeg – тайм-код начала фрагмента, timeEnd – тайм-код конца фрагмента. Если не задано timeBeg, то считается, что фрагмент начинается с начала файла. Если не задано timeEnd, то для видео файла считается, что фрагмент продолжается до конца файла, а для аудиофайлов до конца видеофрагмента.

Последовательность фрагментов во внешнем файле JavaScript задается точно так же, но в качестве имени структуры нужно использовать предопределенный идентификатор vp_playlist:

```
vp_playlist= {
…..
}
```

Задание последовательности фрагментов в файле XML имеет следующий вид:

```
<xml version="1.0" encoding="UTF-8"?>
<SMIL>
<SEQ>
<VIDEO src=" http://serv1.ru/clip1.webm " timeBeg="10" timeEnd="30" />
<PAR>
<VIDEO src=" http://serv2.ru/clip2.webm "  timeBeg="200", timeEnd="300"/>
<AUDIO src=" http://serv1.ru /voice1.ogg " timebeg="20"/>
</PAR>
<PAR>
<VIDEO src=" http://serv1.ru/clip1.webm"  timeBeg="30" timeEnd="180"/>
<AUDIO src=" http://serv3.ru/voice2.webm " timebeg="60"/>
</PAR>
</SEQ>
</SMIL>
```

Последовательность фрагментов задается элементом с тегом <SEQ></SEQ>. Элементами последовательности могут быть или простые

видеофрагменты, заданные элементами VIDEO, или видеофрагменты с наложенным звуком, задаваемые элементами VIDEO и AUDIO, заключенные в элемент <PAR></PAR>. Элементы VIDEO и AUDIO имеют атрибуты src, timeBeg и timeEnd, смысл которых такой же, как описано выше.

В начале работы, когда плеер получает последовательность фрагментов, он пытается установить соединение со всеми заданными файлами. Если какой-либо фрагмент оказывается недоступным, то дальнейшая работа невозможна и выдается сообщение об ошибке. Такое предварительное соединение необходимо, чтобы плеер мог определить продолжительность каждого фрагмента и общую продолжительность клипа. Кроме того, осуществляется предварительная буферизация фрагментов.

Плеер не содержит никаких интерфейсных средств управления, таких как кнопки, полосы быстрого поиска, информационные панели. Эти средства могут иметь различный внешний вид, расположение, что во многом зависит от общего дизайна WEB страницы. Предполагается, что разработчик WEB страницы должен самостоятельно разработать и создать необходимые элементы. Но плеер содержит ряд функций-методов, которые можно использовать для управления. В качестве реакции на интерфейсные кнопки управления используются следующие функции:

vpl.butPlay() – начать проигрывание,

vpl.butPause () – пауза,

vpl.butPlayPause () – комбинированная функция переключения между состоянием проигрывания и паузы,

vpl.butStop () – остановка, после этого при следующем вызове butPlay проигрывание начнется с начала файла,

vpl.butStep (d) – эта функция в состоянии паузы перемещает "головку" плеера на время d вперед если d >0 и назад, если d<0. d задается в миллисекундах,

vpl.setVolume (v) – установка уровня громкости, v – число от 0 до 100. 0 – нет звука, 100 – максимальная громкость.

vpl.setMute ([m]) – убирает или устанавливает звук. Если m <> 0, устанавливается режим без звука, если m =0, устанавливается режим со звуком, если m не задано, то переключает между режимами.

Для быстрого поиска по тайм-коду используются следующие функции:

vpl.seekBegin ()
vpl.seekSet (p)
vpl.seekEnd ()

Обычно поиск по тайм-коду осуществляется с помощью линейки с ползунком и мыши. Когда пользователь нажимает кнопку мыши над ползунком или линейкой, вызывается функция seekBegin. При перемещении мыши вызывается функция seekSet, которая задает позицию в видеопоследовательности в процентах от ее общей продолжительности. При отпускании кнопки мыши вызывается функция seekEnd, завершающая процедуру поиска, и воспроизведение продолжается. Между вызовами seekBegin и seekEnd может быть много вызовов функций seekSet, при этом если плеер успевает осуществить поиск, то он показывает изображение с найденным тайм-кодом, но воспроизведение продолжится только после вызова seekEnd.

Выше указывалось, что при создании объекта плеера ему передается ссылка на функцию обратного вызова. С помощью вызова этой функции плеер информирует окружение о своем состоянии. Плеер вызывает функцию с интервалом 50 мсек и передает ей информацию в виде структуры со следующими полями:

currentTime – текущий тайм-код,
duration – общая продолжительность клипа,
muted – 0, если режим со звуком, <>0, если без звука,
volume – текущий уровень громкости,
state – текущее состояние: 0 – предварительное соединение еще не установлено, 1- готовность, 2- воспроизведение, 3 – пауза, 4 – состояние поиска по таймкоду, 5 – достигнут конец клипа, 15 – ошибка.

Необходимо отметить, что разработанный плеер написан на языке JavaScript без использования популярных библиотек, таких как dojo, jQuery или Prototype. Такой подход выбран специально, так как эти библиотеки не всегда совместимы между собой. А при использовании JavaScript "в чистом виде" плеер может быть использован на страницах разрабатываемых с использованием любой из этих библиотек.

Реализация элемента для воспроизведения файлов в формате FLV

Используемый в плеере элемент для воспроизведения файлов в формате FLV создан в системе Flash и написан на языке ActiveScript. Элемент представляет собой экран, воспроизводимый с помощью встроенного объекта Video. Кроме этого используется встроенный невидимый объект Sound, который нужен для воспроизведения и управления звуком. Элемент не имеет никаких отображаемых средств управления. Управление Flash элементом осуществляется из плеера. При программировании в системе Flash имеется

возможность объявить, что некоторые функции, написанные на ActiveScript, могут быть вызваны из внешнего окружения — программы на JavaScript в Web странице. Был разработан набор интерфейсных функций так, что их набор по возможностям был похож на интерфейс видео элемента HTML5.

Используются следующие функции:

FlsetSrc (url) – устанавливает соединение с указанным url,

Flplay() – начинает воспроизведение,

Flpause() – пауза,

FlsetCurrentTime(t) – устанавливает текущий тайм-код,

FlcurrentTime() – возвращает текущий тайм-код,

FlsetMuted(m) – устанавливает режим со звуком или без звука,

Flmuted() – возвращает текущий режим со звуком или без звука,

FlsetVolume(v) – устанавливает громкость,

Flvolume() – возвращает текущую громкость,

Flduration() – возвращает продолжительность файла,

Flerror() – возвращает код ошибки или 0, если не было ошибки

Flpaused() - возвращает не 0, если режим паузы

Flseeking() – возвращает не 0, если поиск по тайм-коду еще не завершен.

Flended() – возвращает не 0, если достигнут конец файла.

Программы, реализующие Flash элемент, содержатся в соответствующем swf файле. Вызов и загрузка этого файла осуществляется по ссылке на WEB странице. Система безопасности браузера требует, что бы этот swf файл был размещен на том же сервере, с которого была вызвана WEB страница. Иначе вызов функций ActiveScript из страницы будет невозможен.

Flash элемент может воспроизводить видео и аудио файлы форматов FLV, MP4 (H.264) и MP3. При разработке Flash элемента особенное внимание уделялось возможности быстрого перехода на требуемый тайм-код. При установке соединения с медиа потоком прежде всего считываются метаданные, и проверяется наличие таблицы ключевых кадров в них. Если таблица ключевых кадров имеется, то Flash элемент предполагает, что файл передается с сервера с использованием какого-либо специального серверного модуля, имеющего возможность передавать нужную часть файла по специальному запросу. Эта таблица считывается и сохраняется во внутренних массивах программы.

Программа на ActiveScript имеет возможность проверять какая часть

видеофайла уже загружена в буфер. Если при переходе по тайм-коду переход осуществляется на часть, которая уже загружена в буфер, то производится вызов внутренней функции seek. Эта же функция используется, если в файле нет таблицы ключевых кадров. Однако в этом случае, если требуемая часть не загружена в буфер, то потребуется время на ожидание загрузки.

Если у файла есть таблица ключевых кадров, и требуемая часть файла еще не находится в буфере, то программа должна отправить серверу новый запрос с указанием какая часть файла требуется. Передача и обработка текущего запроса при этом прекращается. По таблице ключевых кадров программа определяет ключевой кадр ближайший к требуемому тайм-коду. Таблицы различаются для различных типов файлов. Для файлов FLV в таблицах содержатся тайм-коды ключевых кадров и смещение от начала файла, с которого размещены пакеты этого кадра. Для файлов MPEG4 в таблице находятся только тайм-коды ключевых кадров. В связи с этим запрос нужной части файла осуществляется по-разному. При запросе FLV файлов в запросе нужно указать номер байта, с которого требуется передавать информацию, а при запросе MPEG4 файлов указывается требуемый тайм-код.

Литература

1. Бухштаб Ю.А., Воробьев А.А., Евтеева Н.Н. Методы виртуального редактирования в распределенной среде видео и аудио потоков, представленных в различных форматах // Препринты ИПМ им. М.В.Келдыша. – 2011. – № 67. – 12 с. – URL: http://library.keldysh.ru/preprint.asp?id=2011-67

2. Synchronized Multimedia Integration Language. - W3C Recommendation 01 December 2008. – URL: http://www.w3.org/TR/2008/REC-SMIL3-20081201/

3. HTML5 – A vocabulary and assotiated APIs for HTML and XHTML. – W3C Working Draft 25 October 2012. – URL: http://www.w3.org/TR/html5/

4. Controlling external video playback with ActionScript. - Flash Professional, Help and tutorials.
– URL: http://help.adobe.com/en_US/flash/cs/using/WS0ED77F00-2006-49aa-8399-92B6D5D8CE00.html