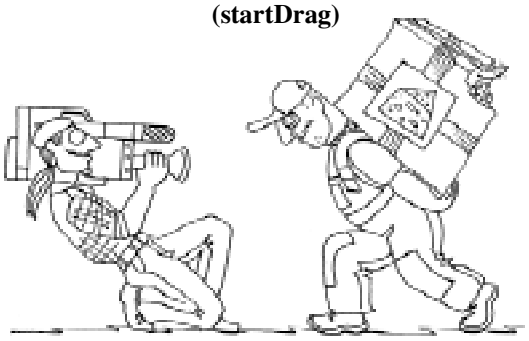
Штенников Дмитрий Геннадьевич

MACROMEDIA FLASH. ОСНОВЫ ПРОГРАММИРОВАНИЯ. ПРОСТЕЙШИЕ СКРИПТЫ. УРОК 4

Мы продолжаем знакомство с созданием интерактивных анимаций в среде Macromedia Flash MX 2004. Вы познакомитесь с созданием пользовательских курсоров мыши и с методами, позволяющими перетаскивать кнопки и Movie Clip (MC) по сцене, а также ознакомитесь со специальными суффиксами, позволяющими легче получать доступ к свойствам и методам.

ПЕРЕТАСКИВАНИЕ ОБЪЕКТОВ (startDrag)



Для перетаскивания объектов можно воспользоваться стандартными методами **startDrag** и **stopDrag**. Выполним несколько примеров и разберемся в требуемой последовательности действий.

Пример 1

Изначально методы **startDrag** и **stopDrag**, предназначенные для запуска и остановки перетаскивания, были созданы для работы с MC, но вся беда заключалась в том, что наиболее удобные события **press** – нажатие и **release** – отпускание были прописаны для кнопки, поэтому приходилось поступать способом, описанным ниже. Создайте символ. Преобразуйте его в кнопку (выделите графический примитив, нажмите F8 и выберите вариант **Button** (кнопка)) (рисунок 1).

Еще раз выделите созданную только что кнопку и преобразуйте ее в MC (рисунок 2).

Войдите в режим редактирования MC, щелкнув по нему дважды левой клавишей мыши (ЛКМ), затем щелкните ЛКМ по кнопке, раскройте панель **Actions** и напишите в ней:

```
on(press){
     this.startDrag();
}
on(release){
     this.stopDrag();
}
```

При этом обратите внимание, что в данном примере было решено использовать указатель на текущий объект **this**. В принципе, если вам не хочется использовать адресацию, то можно обойтись и без нее:

```
on(press){
     startDrag ("");
}
on(release){
     stopDrag ();
}
```

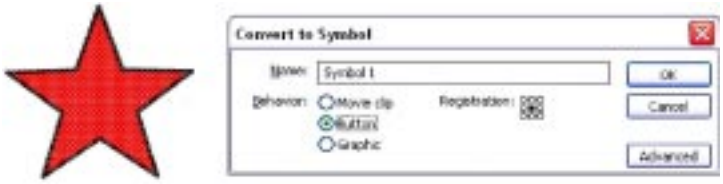


Рисунок 1.

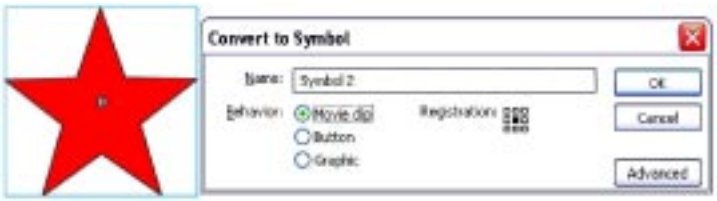


Рисунок 2.

Итак, при помощи нажатой ЛКМ нарисованный МС возможно передвигать по экрану. Разница в двух примененных выше скриптах заключается в том, что первый скрипт использует указатель на текущий объект **this**, а второй скрипт использует уже устаревшую, но все же по-прежнему работающую конструкцию, оставшуюся со времени, когда язык использовал процедурное программирование.

Пример 2

Попробуйте выполнить операции в иной последовательности, а именно, вначале создать МС и только затем кнопку, после чего в качестве команды для кнопки написать один из представленных выше скриптов. Попробуйте найти разницу в действиях этих скриптов.

Подсказка: в одном случае перетаскивается непосредственно МС, в другом сцена. Это хорошо заметно, если создать сразу несколько объектов на сцене.

Пример 3

Ситуация изменилась с момента, когда появилась возможность написания скриптов на МС. В таком случае возможно нарисовать графический примитив и преобразовать его в МС (рисунок 3), после чего раскрыть панель **Actions**:

```
onClipEvent(mouseDown){
     this.startDrag();
}
onClipEvent(mouseUp){
     this.stopDrag();
}
```

Если вы собираетесь перетаскивать один объект по сцене, все нормально, но если ставится задача перетаскивать более чем один символ по сцене, то возникнут проблемы.

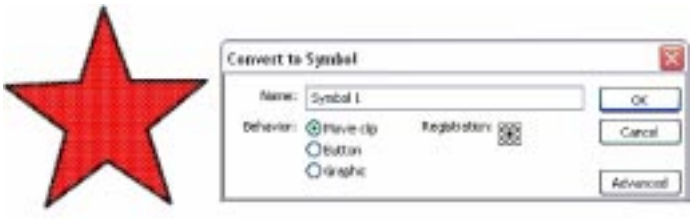


Рисунок 3.

Пример 4

В последней версии Flash не только МС, но и кнопки получили имя (**Instance Name**), поэтому стало возможным применить методы **startDrag** и **stopDrag** непосредственно к кнопкам, следовательно, скрипт:

```
on(press){
     this.startDrag();
}
on(release){
     this.stopDrag();
}
```

можно применить непосредственно к кнопкам, но, увы, как и в примере 3, перетаскивание по сцене более чем одного символа осложнено.

СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО КУРСОРА

Для создания собственного курсора воспользуемся изученным выше методом **startDrag()**. Для того чтобы создать свой

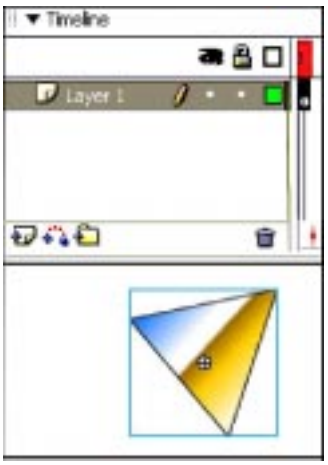


Рисунок 4.

собственный курсор, необходимо выполнить следующую последовательность операций:

1) Создайте свой собственный МС и разместите его на сцене. МС может быть анимированным или нет, в зависимости от поставленных вами задач (рисунок 4).

2) Перейдите в режим написания скрипта на МС (для этого щелкните ЛКМ по МС и раскройте панель Actions) и в этом режиме запишите следующий скрипт:

```
onClipEvent(load){
startDrag(this, true);
}
```

В данном случае мы воспользовались модифицированным методом **startDrag**, в котором появился ранее не используемый параметр **true**, означающий, что теперь **Registration Point** МС (то есть та точка, относительно которой МС будет перемещаться) будет совпадать с центром курсора.

3) Запустите тестовый режим и посмотрите, что у вас получилось. Как вы видите, сразу после загрузки ролика за курсором мыши передвигается нарисованный вами МС. Но при этом центр созданного таким образом курсора позиционируется как раз под центром, поэтому необходимо выполнить следующую операцию:

4) Войдите в режим редактирования МС, щелкнув по МС дважды ЛКМ (рисунок 5).

5) Выделите содержимое вашего МС (в данном случае это простой графический примитив) и при помощи ЛКМ передвиньте его таким образом, чтобы крестик, показывающий **Registration Point** МС, совпадал с указателем курсора (рисунок 6).

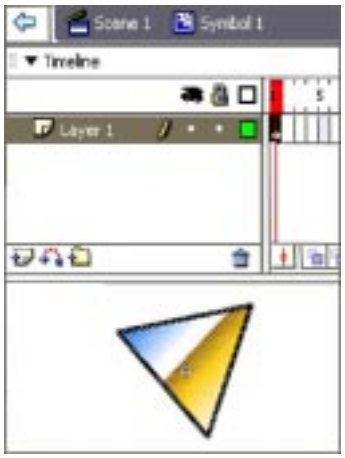


Рисунок 5.

Рисунок 6.

6) Запустите клип. Теперь действительно указатель курсора совпадает с **Registration Point**.

Есть и второй способ. Если МС, созданный для курсора, направлен под углами 0°, 45°, 90°, 135°, 180° и т. д. к горизонтальной линии, то возможно сделать следующее.

При создании МС выберем **Registration Point** в одной из 8 позиций (см., например, рисунок 7).

Теперь сделаем курсор невидимым, для этого допишем скрипт на МС.

```
onClipEvent(load){
mouse.hide();
startDrag(this, true);
}
```

Все. Курсор мыши пропадает, а вместо него по экрану движется тот курсор, который вы сами нарисовали.

К аналогичному результату приводит следующий скрипт:

```
onClipEvent(load){
mouse.hide();
startDrag("", true);
}
```

Проверьте его работоспособность на своем курсоре.

Пример 5

Аналогичного действия можно добиться при обращении к курсору мыши не через МС, а через линейку времени (**timeline**). Для этого необходимо создать МС на сцене, в панели **Properties** задать ему имя, например **"a"**. После чего щелкнуть

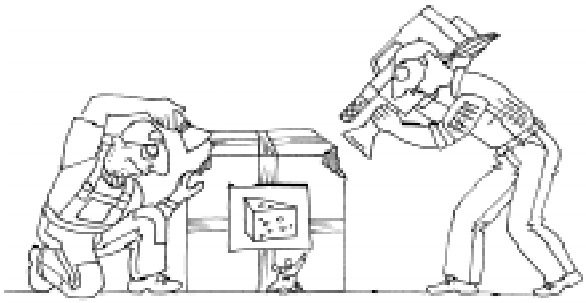
Рисунок 7.

ЛКМ по первому кадру, раскрыть панель **Actions** и написать скрипт:

```
startDrag("a",true);
mouse.hide();
```

Курсор мыши готов.

ПОЛУЧЕНИЕ КООРДИНАТ МЫШИ



Вы можете использовать координаты мыши (**_xmouse и _ymouse**) в своей работе с МС. В каждом МС содержится своя собственная информация.

Для наблюдения свойств **_xmouse** и **_ymouse** в основной линейке кадров и в линейке кадров МС необходимо обрабатывать координату мыши и помещать ее в переменную.

Следующее выражение может быть помещено в каждую из линеек кадров, для того чтобы передавать координаты мыши в основную сцену, она же **_root**, она же **_level0**:

```
x_pos = _level0._xmouse;
```

Для того чтобы определить позицию мыши внутри МС, необходимо задействовать его имя, которое задается в поле **Instance Name** панели **Instance**.

Нарисуйте на сцене прямоугольник, выделите его, нажмите клавишу F8 и назовите ваш прямоугольник Movie Clip-ом, после чего задайте ему **Instance name**.

Создайте два динамических текстовых поля на сцене, в которые вы будете выводить глобальные координаты мыши, и два поля внутри МС.

Для того чтобы получить глобальные координаты мыши, в скрипте на ключевом кадре можно записать:

```
xpos = _root._xmouse;
ypos = _root._ymouse;
```

Как видно из написанного скрипта, мы используем две переменные, в которые передаются значения текущей позиции курсора.

Для определения позиции курсора внутри МС можно поместить аналогичную запись или на самом МС записать:

```
onClipEvent(enterFrame){
     xmousePosition = _xmouse;
     ymousePosition = _ymouse;
}
```

Для обработки координат мыши можно использовать и следующее творчество:

```
onClipEvent(mouseMove){
     x_pos = _root._xmouse;
     y_pos = _root._ymouse;
}
```

СОЗДАНИЕ ФОРМЫ

Чтобы создать текстовое поле, возьмем инструмент **Text Tool** и, нажав ЛКМ в том месте, где надо поставить это поле, вытянем его до нужных размеров. Рядом поставим название этого поля, например, пусть это будет **Name:**

Далее в окне **Text Options** выставляем параметр текста **Input Text** и задаем имя переменной (**Variable**), равной **Name**. Также поставим галочку на **Border/bg**.

Повторите эти шаги для текстовых полей **Homepage**, **e-mail** и **Comments**. Для поля **Comments** установите параметр **Multiline** и **Word Wrap**. **Multiline** позволит вводить в несколько строк, а **Word Wrap** будет гарантировать, что весь текст будет замечен.

Создайте две кнопки **Send** и **Clear**.

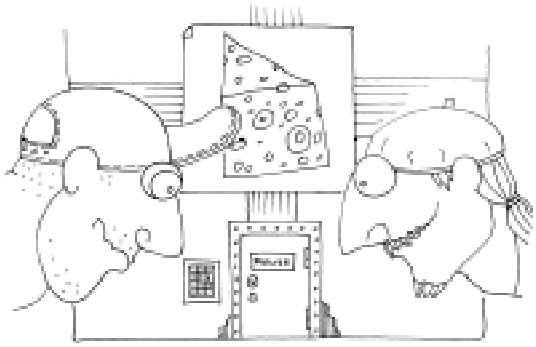
К кнопке **Clear** добавим:

```
on(release){
name = ""
homepage = ""
email = ""
comments = ""
}
```

К кнопке **Send** добавим:

```
on (release){
Load Variables ("http://www.server.com/
cgi-bin/myform.cgi", 1, vars=POST)
}
```

СОЗДАНИЕ ПАРОЛЯ (password)



Создайте новый МС и установите в нем 2 слоя. Первый **frame** слоя 1 имеет:

1. **Input text** с именем переменной **login**;
2. Кнопку **Send**;
3. **Actions** для первого кадра :

```
password = login;
```

Во втором слое фрейма 2 установите следующее:

GotoAndPlay (1);

Десятому фрейму этого же слоя дайте название **"error"**.

Первый слой на 10 фрейме имеет кнопку **Retry**.

Теперь добавим **actions** для кнопок.

Кнопка **Send**:

```
on (release) {
if (password eq "server") {
getURL ("http://www.server.ru ",
                        "_blank");
} else if (password eq "action") {
getURL ("http://www.server.ru ",
                        "_blank");
} else if (password eq "goga") {
getURL ("http://www.server.ru",
                        "_blank");
} else {
gotoAndStop ("error");
}
}
```

Кнопка **Retry**:

```
on (release, dragOver) {
gotoAndPlay (1);
}
```

Как это работает?

Когда мы вводим пароль, переменной **login** присваивается значение: **login=***«то, что вы ввели в поле для пароля»*. Далее, при нажатии на кнопку **Send**, проверяется, является ли это слово паролем. Сначала оно проверяется с **"server"**. Если это тот пароль, то считывается следующая строка, к которой идет переход по заданной ссылке. Если нет, – проверяется с другим паролем, потом со следующим, пока не наступит **"else"** – что является переходом на фрейм **"error"**.

СУФФИКСЫ ДЛЯ АКТИВАЦИИ ФУНКЦИИ ЗАВЕРШЕНИЯ КОДА

Специальные суффиксы существуют для облегчения ввода свойств и методов, характерных для объектов. Дело в том, что тип переменных во Flash явным образом может не задаваться, и одна и та же переменная может в разные моменты времени существовать и как целочисленная, и как вещественная, и как строковая, поэтому, в отличие от многих пакетов для программирования, содержит лишь скрытые подсказки. Для их активации необходимо использовать специальные суффиксы, приведенные в таблице 1.

Вы увидите эту функцию в действии, если наберете в панели Actions один из примеров и символ точки. Например, введите **myString_str**. – появится выпадающий список свойств и методов для объекта **String**. Не забывайте о точке (рисунок 8).

Таким образом, в дальнейшем, вы сможете легче писать программы на языке ActionScript.

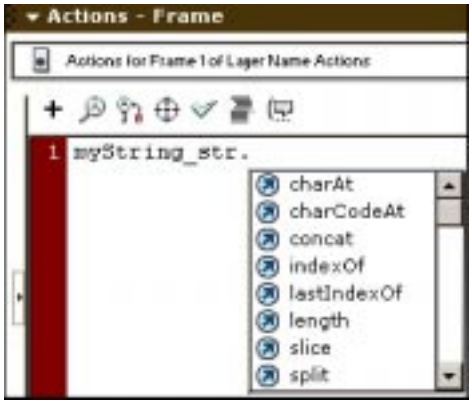
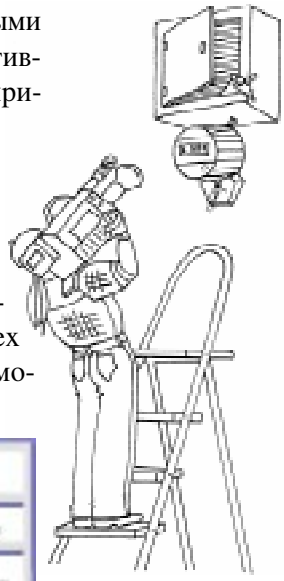


Рисунок 8.

Таблица 1.

Тип объекта	Суффикс	Пример
String	_str	myString_str
Array	_array	myArray_array
MovieClip	_mc	myMovieClip_mc
TextField	_txt	myTextField_txt
Date	_date	myDate_date
Sound	_sound	mySound_sound
XML	_xml	myXML_xml
Color	_color	myColor_color
Button	_btn	myButton_btn
TextFormat	_fmt	myTextFormat_fmt
XMLSocket	_xmlsocket	myXmlSocket_xmlsocket
FListBox	_lb	myFListBox_lb
FScrollBar	_sb	myFScrollBar_sb
FComboBox	_cb	myFComboBox_cb
FScrollPane	_sp	myFScrollPane_sp
FMessageBox	_mb	myFMessageBox_mb
FDraggablePane	_dp	myFDraggablePane_dp
FTicker	tick_ (префикс)	tick_myFTickerMain
FTree	_tree	myFTree_tree
FTreeNode	_tn	myFTreeNode_tn
FIconButton	_ib	myFIconButton_ib
FProgressBar	_pr	myFProgressBar_pr

Теперь Вы ознакомились с самыми общими принципами создания интерактивных проектов на Flash, настало время приступить к разбору прикладных задач.



СОЗДАНИЕ СЧЕТА ОПЛАТЫ ЭЛЕКТРОЭНЕРГИИ

Интерфейс довольно прост (рисунок 9), он состоит из одной кнопки и трех текстовых полей, два текстовых поля могут иметь свойство **dynamic text** – это поля для вывода суммы, которую необходимо внести за оплату электроэнергии, и для вывода сообщения о том, хватило ли вносимой суммы для оплаты, а вот поле, в которое будет вводиться сумма оплаты, должно иметь свойство **input text**.

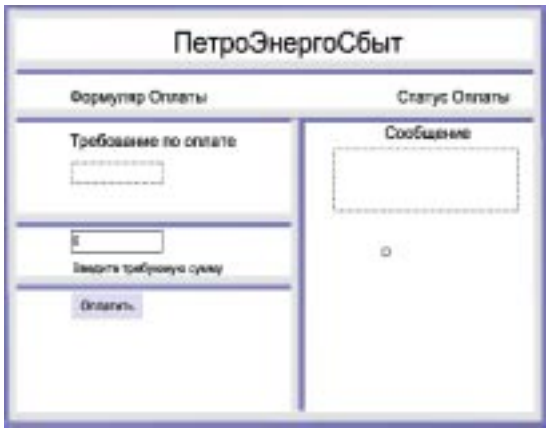


Рисунок 9.

Теперь откройте программу Блокнот и наберите в ней: &electricBill=60. Сохраните файл под названием Electric_Bill.txt в той же папке, куда в дальнейшем запишете файл из Flash. Закройте Блокнот.

Займемся снова файлом во Flash. Текстовому полю, в котором будет показываться требуемая сумма об оплате, подгруженная из текстового файла, необходимо задать имя, для этого необходимо щелкнуть по нему ЛКМ и в панели

Properties задать ему **Instance name:** **owed_txt** (рисунок 10).

Для поля, в которое будет выводиться сообщение об удачной или неудачной оплате, необходимо задать имя **message_txt** (рисунок 11).

Текстовое поле, в которое необходимо вводить сумму для оплаты, назовем: **paid_txt** (рисунок 12).

Теперь создадим МС, в который будем выводить дополнительное сообщение о достаточности оплаты по счетам. Как это делается, вы помните из предыдущих частей. Важно отметить, что в МС потребуется создать минимум 4 ключевых кадра. Содержимое первого ключевого кадра будет пустым, во втором ключевом кадре будет содержаться сообщение о том, что оплата недостаточна, в третьем ключевом кадре – о том, что оплата соответствует запросу, а в четвертом – сообщение о том, что оплата превысила требования. Для того чтобы без осложнений переходить в требуемый ключевой кадр, каждый из них должен иметь метку, задаваемую в панели **Properties** (рисунок 13).

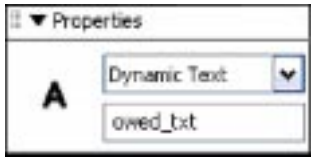


Рисунок 10.

Рисунок 11.

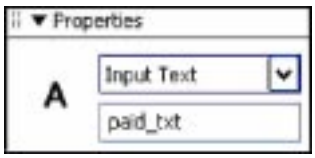


Рисунок 12.

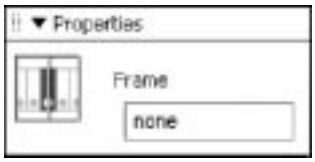


Рисунок 13.

Ключевые кадры назовем: **none**, **underpaid**, **paid_in_full**, **overpaid**. В результате чего **Timeline** приобретает вид, приведенный на рисунке 14.

Для того чтобы проигрывание данного клипа не шло автоматически, необходимо в первом ключевом кадре задать функцию остановки проигрывания:

```
stop();
```

Вернитесь на сцену. В ключевом кадре введите скрипт, при помощи которого будут загружены данные из текстового файла, который был ранее создан.

```
var externalData:LoadVars =
                    new LoadVars();
externalData.onLoad = function(){
owed_txt.text =
        externalData.electricBill;
}
externalData.load("Electric_Bill.txt");
```

При помощи данного скрипта был описан объект **externalDataLoadVars**, принадлежащий классу **LoadVars** и для созданного объекта на событие загрузки была описана функция по присвоению тексту в поле **owed_txt** значения этого объекта. Последняя сточка указывает, из какого файла будет браться текст. Это достаточно сложная конструкция, но, с точки зрения программирования, наиболее правильная. В принципе можно использовать:

```
loadVariablesNum("Electric_Bill.txt", 0);
owed_txt.text = electricBill;
stop();
```

Но в таком случае на сцене было бы необходимо использовать два ключевых кадра, а не один, как в данном примере.

Теперь займемся кнопкой, а в ней необходим следующий скрипт:

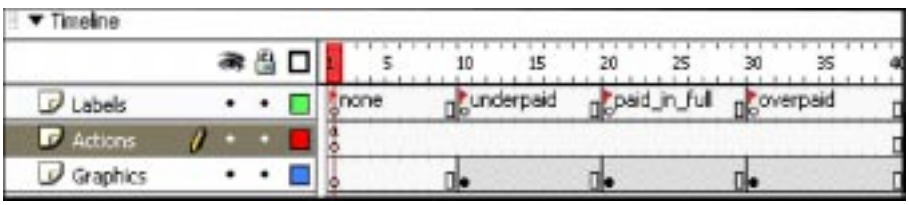
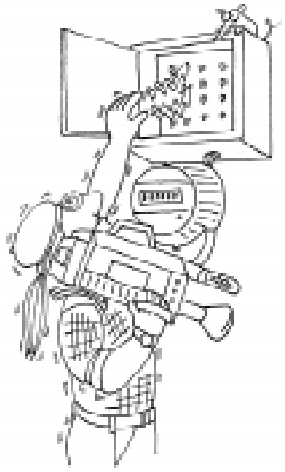


Рисунок 14.

```
on (press) {
  var amountPaid:Number = Number (paid_txt.text);
  var amountOwed:Number = Number (owed_txt.text);
  if (isNaN (amountPaid)) {
    message_txt.text = "What you entered is not a proper dollar amount.
                                                    Please try again.";
  } else if (amountPaid < amountOwed) {
    var difference:Number = amountOwed - amountPaid;
    stamp_mc.gotoAndStop ("underpaid");
   message_txt.text = "You underpaid your bill by " + difference + " dollars.";
  } else if (amountPaid > amountOwed) {
    var difference:Number = amountOwed - amountPaid;
    stamp_mc.gotoAndStop ("overpaid");
   message_txt.text = "You overpaid your bill by " + difference + " dollars.";
  } else {
    stamp_mc.gotoAndStop ("paid_in_full");
    message_txt.text = "You have paid your bill in full.";
  }
}
on (release) {
  stamp_mc.gotoAndStop ("none");
  message_txt.text = "";
}
```



Как видим, данный скрипт производит сравнение значений, введенных в текстовое поле для платежа, со значениями из текстового файла, после чего переходит в тот или иной ключевой кадр МС и выдает соответствующее сообщение, прошла ли оплата. Стоит отметить, что в строчках

```
var amountPaid:Number = Number (paid_txt.text);
var amountOwed:Number = Number (owed_txt.text);
```

var и **:Number** приведены для большей корректности записи, можно обойтись и без особых излишеств.

Материалы, представленные в статье, используются автором в учебном процессе Санкт-Петербургского регионального центра Федерации Интернет Образования.

Штенников Дмитрий Геннадьевич,
доцент кафедры физики
СПбГУ ИТМО.