ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЛЕСОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационных технологий и моделирования

О.А. Карасева

МИРОВЫЕ ИНФОРМАЦИОННЫЕ РЕСУРСЫ

Часть 1. Основы создания сценариев. VBScript

Методические указания
по выполнению лабораторно-практического цикла
для студентов очной, очно-заочной и заочной форм обучения
специальности 080801 «Прикладная информатика
в экономике».
Направление 080800 «Прикладная информатика»

Екатеринбург
2009

Печатаются по рекомендации методической комиссии ФЭУ.
Протокол № 8 от 15 апреля 2009 г.

Рецензент − доцент кафедры ИТМ Монтиле А.И.

Редактор Н.А. Майер
Оператор Г.И. Романова

Подписано в печать 20.05.09		Внеплановая
Плоская печать	Формат 60×84 1/16	Тираж 100 экз.
Заказ №	Печ. л. 3,25	Цена 11 руб. 00 коп.

Редакционно-издательский отдел УГЛТУ
Отдел оперативной полиграфии УГЛТУ

ВВЕДЕНИЕ

Данные методические указания предназначены для получения представления о программировании на Visual Basic Script, подмножестве языка Visual Basic.

VBScript позволяет решать задачи, связанные с Internet, а именно создавать сценарии (или скрипты) управления объектами (кнопками, списками, ниспадающими меню и т.д.) на Web-страничках. Для понимания этого материала нужно иметь представление о структуре HTML-документа, а также основах программирования на Visual Basic. Описанные сценарии могут быть использованы в браузере Microsoft Internet Explorer (IE). Другие браузеры попросту не понимают этого языка. В настоящее время существуют всего два языка создания сценариев по управлению объектами - Microsoft VBScript и Sun JavaScript. Оба поддерживаются IE. Браузер же компании Netscape воспринимает только JavaScript.

Объектная модель Internet Explorer

Основным достоинством VBScript является возможность управления браузером. Объектная модель Internet Explorer определяет набор объектов, их свойства, методы и события, которыми можно оперировать при помощи VBScript. Объектная модель представляет собой структуру, включающую одиннадцать объектов, назначение которых приведено в табл.1.

Таблица 1

Объектная модель Internet Explorer

Название объекта	Назначение объекта
Window	Представляет собой окно браузера
Frame	Представляет собой так называемый фрейм. Данных объектов может быть несколько в окне браузера. Каждый из фреймов отображает одну из Web-страниц. Таким образом, фрейм можно рассматривать как окно в окне

Окончание таблицы 1

Название объекта	Назначение объекта
Location	Позволяет получить информацию о текущем значении URL страницы
Navigator	Предоставляет сведения об используемом браузере
History	Служит для доступа к хронологическому списку документов
Script	Данный объект содержит сценарии, определенные тегом <SCRIPT>
Document	Представляет собой документ, который содержится в текущем окне
Link	Включает в себя массив гиперссылок документа
Anchor	Содержит массив якорей документа
Form	Представляет собой массив форм документа
Element	Представляет собой массив встроенных элементов управления HTML, заключенных в форме или документе

Все перечисленные объекты являются дочерними от объекта Window.

Основы создания сценариев

Если для задания программного сценария используется VBScript, следует указать строку вида:

<SCRIPT LANGUAGE="VBScript"> …… </ SCRIPT>

Размещение сценария в HTML-документе имеет несколько альтернативных решений. Наиболее часто его программный код располагают между тегами <HEAD> и </HEAD>. При этом сценарий выглядит как в листинге 1, приведенном ниже. Также код сценария может быть размещен в пределах теговой пары <BODY>, </BODY>.

Существует несколько способов вызова сценария. Наиболее используемый и характерный для Visual Basic основывается на объявлении процедуры, имя которой состоит из имени элемента управления и названия события, обрабатываемого процедурой, разделенных символом подчеркивания. Использование этого способа вызова сценария продемонстрировано в листинге 1. Название *button*- кнопки - *TestB* составляет первую часть имени процедуры *TestB_OnClick*, а вторая часть имени состоит из названия

события – *OnCIick*, генерирующегося при нажатии на эту кнопку. Описанная таким образом процедура будет выполняться всякий раз при нажатии пользователем кнопки.

Далее вам предлагается, воспользовавшись блокнотом, набрать коды простых сценариев для знакомства с объектами языка VBScript.

Приступая к работе, помните, что принцип компетентности по Питеру: «Чтобы избегать ошибок, надо набираться опыта; чтобы набираться опыта, надо делать ошибки».

Листинг 1. Расположение кода сценария в теговой nape <HEAD>, </HEAD>

```
<HTML>
<HEAD> <TITLE>Пример 1</TITLE>
<SCRIPT LANGUAGE="VBScript">
 Sub TestB_OnClick
       Alert   "ДА!"
End  Sub
</SCRIPT> </HEAD>
<BODY BGCOLOR="White"> <P ALIGN="Center" ><INPUT TYPE="button"
NAME="TestB"   VALUE = "Test"></P> </BODY>
 </HTML>
```

Существует еще один сходный вариант вызова. При помощи атрибута FOR тега <SCRIPT> можно задать элемент управления, обработчик события которого располагается непосредственно под тегом <SCRIPT>, а само событие объявляется атрибутом EVENT, как показано в листинге 2.

Листинг 2. Использование атрибутов FOR и EVENT тега <SCRIPT>

```
<HTML>
<HEAD>
<TITLE>Пример 2</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<P ALIGN="center"> <INPUT  TYPE="button" NAME="TestB"  VALUE="Test" >
</P>
<SCRIPT  FOR ="TestB"   EVENT="OnClick"
LANGUAGE="VBScript">
Alert "ДА!!"
</SCRIPT>
</BODY>
</HTML>
```

РАБОТА С БРАУЗЕРОМ

Объект Window

Объект *Window* содержит несколько методов, свойств и событий. Некоторые из них перечислены в табл. 2-4. Объект *Window* является исходным объектом, так что вам не требуется добавлять его имя к свойствам или дочерним объектам. Например, вместо *Window*.name вы могли бы использовать только *name*.

Таблица 2

Свойства объекта *Window*

Свойство	Назначение
Frames	Используется, когда в текущем документе существует набор кадров (фреймов). Это свойство является массивом фреймов на странице
Location	Определяет местоположение текущего окна
Name	Соответствует имени того окна, чьи свойства определяются
Parent	Указывает родительский фрейм или окно, чьи свойства определяются на данный момент

Таблица 3

События загрузки объекта *Window*

Событие	Назначение
OnLoad	Вызывается, когда загружается страница, содержащая данное событие. Используется в теге <BODY> как атрибут при вызове процедуры
OnUnLoad	Вызывается, когда выгружается страница, содержащая данное событие. Используется в теге <BODY> как атрибут при вызове процедуры

Таблица 4

Несколько основных методов объекта *Window*

Метод	Назначение
Open	Открывает или закрывает документ внутри текущего окна или другого определенного окна. У него есть два аргумента: имя файла, который вы хотите открыть, и имя окна, на котором вы хотите его разместить. Например, Window.Open(filename,windowname), где filename - имя файла, а windowname - имя окна. Для этого метода требуются оба параметра
Prompt	Определяет всплывающую подсказку. Имеет два параметра: текст подсказки и любой заданный по умолчанию текст, который входит в подсказку. Например: Window.Prompt(ExpString, DftString), где ExpString - текст подсказки, а DftString - исходный текст в подсказке. Оба параметра необязательны. Если этот метод связан с переменной (как x=prompt("stuff","more stuff")), то эта переменная получит любые данные, которые были введены в подсказку, если не была нажата "отмена"

Окончание таблицы 4

Close	Используется для закрытия окна
Navigate	Переключает окно на другой адрес URL. Например: Window.Navigate(URL), где URL - имя того URL, на который нужно перейти

Рассмотрим некоторые свойства объекта *Window*.

DefaultStatus. Предоставляет возможность задать содержимое строки состояния браузера по умолчанию, то есть в случае, когда строка состояния не отображает специальное предопределенное сообщение. Код примера, отображенного в листинге 3, помещает в строку состояния текст "Это пример!" (рис.1).

Листинг 3. Работа с методом *DefaultStatus*

```
<HTML>
 <HEAD>
<SCRIPT LANGUAGE="VBScript">
DefaultStatus = "Это пример!"
</SCRIPT>
<TITLE>DefaultStatus </TITLE>
</HEAD>
<BODY  BGCOLOR="#FFFFFF">
<P ALIGN="center" > <FONT  COLOR="#FF0000"
SIZE="4">Свойство объекта Window</FONT>
<FONT  color="#FF0000"   SIZE="5">
<EM><STRONG>DefauitStatus</STROMG</EM> </FONT>
</P>
</FORM> </BODY> </HTML>
```

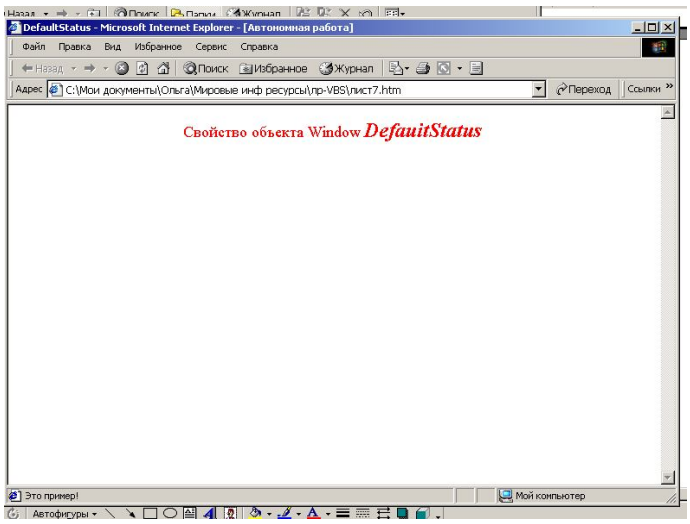

Рис.1 Работа с методом *DefaultStatus*

Объект *Window* поддерживает ряд методов, приведенных ниже.

Alert. Служит для выдачи сообщений в специальном окне. Данный метод включает в себя один параметр, представляющий собой строку вывода, и имеет следующий вид:

```
Window.Alert строка_сообщения
```

Листинг 4 демонстрирует простейшее использование метода *Alert* (рис.2).

Листинг 4. Работа с методом *Alert*

```
<HTML>
 <HEAD>
<SCRIPT   LANGUAGE="VBScript">
 Sub AlertB_OnClick
     Alert   "Вы нажали кнопку!"
End Sub
 </SCRIPT>
<TITLE>Метод Alert</TITLE> </HEAD>
<BODY  BGCOLOR="#FFFFFF">
<P ALIGN="center" ><FONT  COLOR="#FF0000"
SIZE="4">Метод </FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
<EM><STRONG >Alert </STRONG  ></EM ></FONT>
</P>
<FORM METHOD="POST">
<P ALIGN="center" ><INPUT
TYPE="button" NAME="AlertB" VALUE="Тест на метод Alert"> </P>
</FORM> </BODY>
</HTML>
```

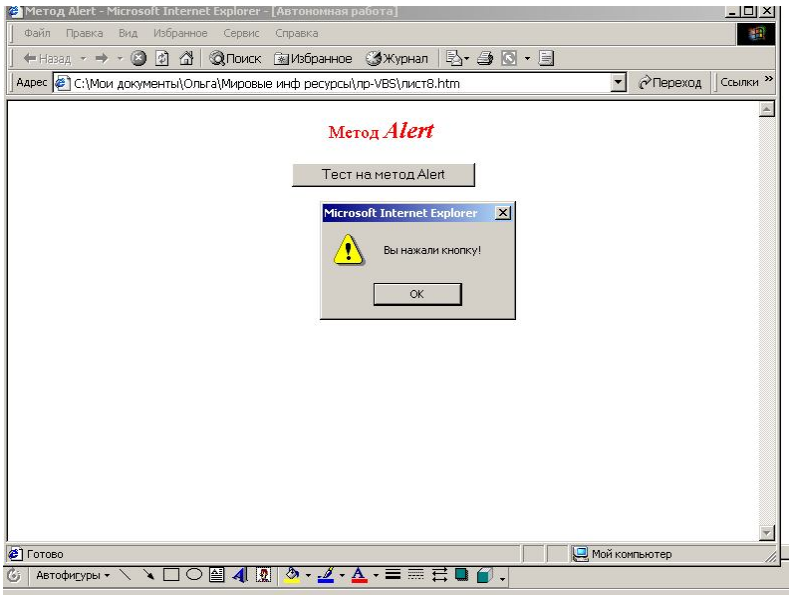

Рис.2. Работа с методом *Alert*

Confirm. В отличие от окна сообщения метода *Alert*, окно метода *Confirm* (рис. 3) включает в себя не одну кнопку *OK, а* две − *OK* и *Cancel.* При выборе кнопки *OK* метод возвращает логическое значение True, а при выборе кнопки *Cancel* - логическое значение *False*.

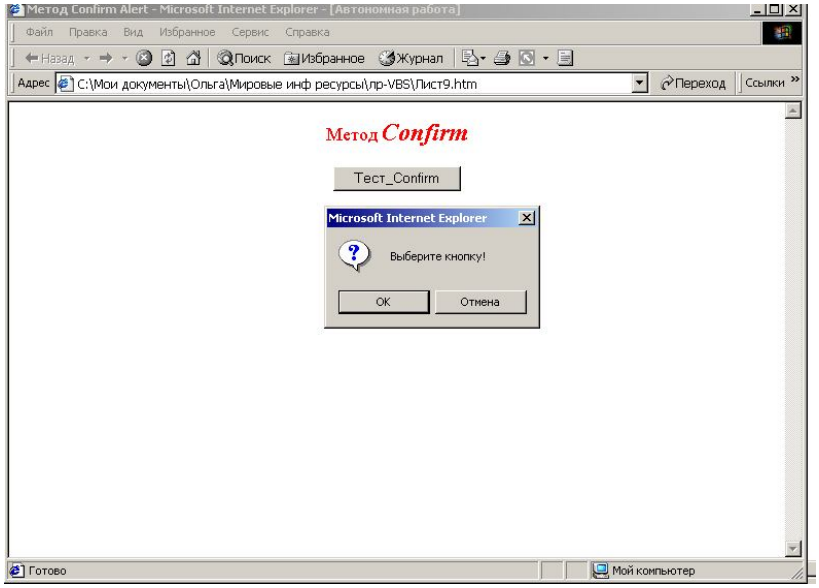

Рис.3. Работа с методом *Confirm*

Листинг 5, приведенный ниже, реализует пример, в котором пользователь должен нажать кнопку в окне браузера. После этого надо либо подтвердить свои действия, либо отказаться от них. В первом случае появляется окно с текстом "YES!!!", а во втором − окно с сообщением "NO! ! !".

Листинг 5. Работа с методом *Confirm*

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub ConfirmB_OnClick
    If Confirm ("Выберите кнопку!") Then
    Alert "YES! !"
    Else
    Alert "NO! ! ! "
    End If
End Sub
</SCRIPT>
<TITLE>Метод Confirm Alert</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
```

```
<P ALIGN="center">
<FONT COLOR="#FF0000" SIZE="4"> Метод</FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
<EM><STRONG>Confirm</STRONG></EM></FONT>
</P>
<FORM METHOD="POST">
<P ALIGN="center" ><INPUT TYPE="button"
NAME= "ConfirmB" VALUE= "Тест_Confirm" >
</P>
</FORM> </BODY> </HTML>
```

Prompt. Данный метод позволяет пользователю вывести специальное диалоговое окно, содержащее поле ввода и строку сообщения. Формат вызова метода имеет следующий вид:

```
строка__ввода = Prompt  (строка_сообшения, вариант ввода)
```

Листинг 6, приведенный ниже, реализует WEB-страницу, в которой пользователь, нажав кнопку, вызывает окно метода *Prompt*. В этом окне можно указать требуемый текст. Введенный текст будет отображен методом *Alert*. По умолчанию вводится "Я учусь" (рис.4).

Листинг 6. Работа с методом *Prompt*

```
<HTML>
<HEAD> <TITLE>Пример 10</TITLE>
<SCRIPT LANGUAGE="VBScript">
Sub TestB_OnClick
    Str = Prompt("Введите текст", "Я учусь")
    Alert Str
End Sub
</SCRIPT> </HEAD>
<BODY BGCOLOR="white">
<P  ALIGN="center" ><FONT COLOR="#FF0000" SIZE="4">Метод</FONT>
 <FONT   COLOR= "#FF0000"   SIZE="5">
 <EM><STRONG>Prompt</STRONG></EM></FONT>
</P>
<FORM  METHOD="POST">
<P ALIGN="center"><INPUT  TYPE="button"
Name = "TestB"  VALUE="Тест Prompt">
</P> </FORM> </BODY> </HTML>
```

Open. Данный метод позволяет создать новое окно браузера. Формат вызова метода имеет следующий вид:

```
Window. Open   (URL,   Имя_нового_окна,   Параметры)
```

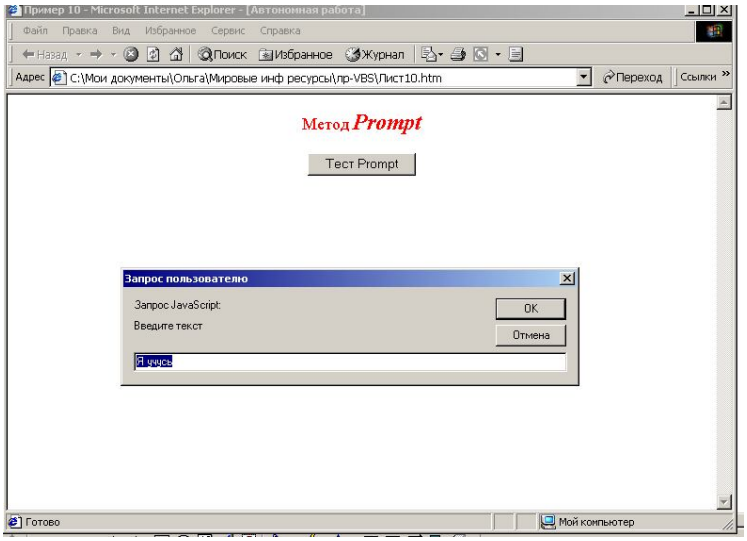

Рис.4. Работа с методом *Prompt*

Листинг 7. Работа с методом *Open*. Во время выполнения наберите такой URL - адрес, который позволит загрузить реальную Web-страницу.

```
<HTML>
<HEAD> <TITLE>Example</TITLE>
<SCRIPT LANGUAGE="VBScript">
Sub TestB_OnClick
Options = "Toolbar=Yes, Location=Yes"
Options = Options + " , Directories=Yes"
Options = Options + " , Status=Yes"
Options = Options + ", menubar=Yes"
Options = Options + ", scrollbars=No"
Options = Options + ", resizable=Yes"
Options = Options + ", width=300"
Options = Options + ", height = 400"
Window.Open "http://www.worldfide.com/index.html" , "worldfide" , Options
End Sub
</SCRIPT> </HEAD>
<BODY BGCOLOR="white">
<P ALIGN="center"><FONT  COLOR="FF0000" SIZE="4">Метод</FONT>
 <FONT   COLOR="#FF0000"   SIZE="5">
 <EM> <STRONG>Open</STRONG></EM></FONT>
</P><P ALIGN="center"><INPUT  TYPE="button" NAME="TestB"
VALUE="Нажмите кнопку для открытия">
</P>
</BODY> </HTML>
```

Navigate. Позволяет загрузить новую страницу в окно браузера. Вызов метода осуществляется посредством команды

Window.Navigate URL

Параметр *URL* задает адрес документа, который должен быть загружен в окно браузера.

В приведенном примере (листинг 8) при нажатии на кнопку появляется окно, в котором пользователю предлагается указать интересующий его *URL*. По умолчанию предложите следующий *URL:* http://www.usfeu.ru. В окно браузера будет загружена страница согласно заданному *URL* (рис.5).

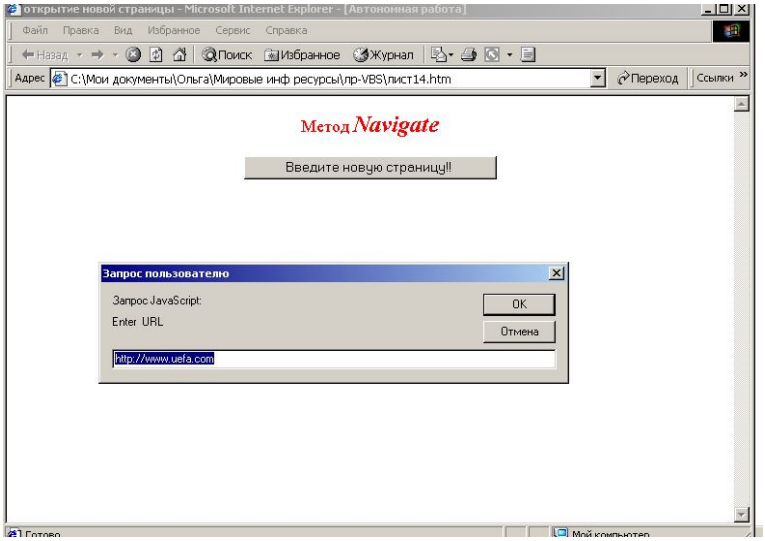

Рис.5. Работа с методом *Navigate*

Листинг 8. Работа с методом *Navigate*

```
<HTML> <HEAD> <TITLE>открытие новой страницы </TITLE>
<SCRIPT  LANGUAGE="VBScript">
Sub  TestB_OnClick
Str  =  Prompt("Enter  URL",  " http://www.usfeu.ru ")
Navigate  Str
End  Sub
 </SCRIPT> </HEAD>
<BODY BGCOLOR="white"> <P ALIGN="center">
<FONT  COLOR="#FF0000" SIZE="4">Метод</FONT>
<FONT  COLOR="#FF0000"  SIZE="5">
<EM><STRONG>Navigate</STRONG></EM></FONT>
<FORM METHOD="POST">
<P ALIGN="center"><INPUT  TYPE="button"
NAME="TestB"  VALUE="Введите новую страницу!! ">
</FORM>
</BODY>
 </HTML>
```

OnLoad. Событие возникает сразу после загрузки документа.

В приведенном ниже примере (листинг 9) показана обработка данного события. Идея примера проста. При загрузке страницы должна появляться подсказка о действиях пользователя, которые должны быть выполнены: "Нажмите и выберите вариант ответа" (рис.6).

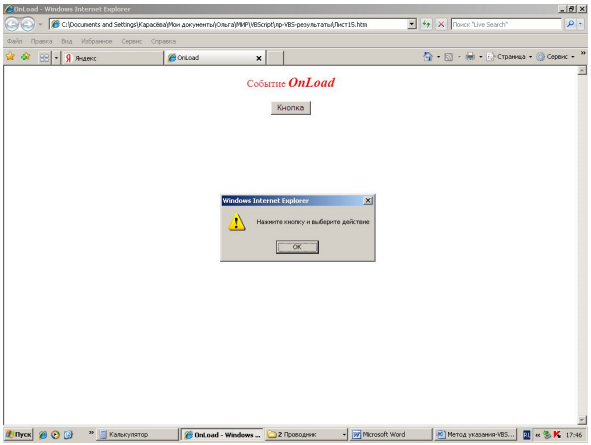

Рис. 6. Стандартная обработка события *OnLoad*

Листинг 9. *Стандартная обработка события OnLoad*

```
<HTML>
<HEAD>
<SCRIPT   LANGUAGE="VBScript">
Sub Window_OnLoad
Alert "Нажмите кнопку и выберите действие"
End Sub
 Sub   TestB_OnClick
If Confirm ("Выберите ответ")   Then
Alert   "Вы выбрали ДА!"
Else
Alert   "Вы выбрали отмену !"
End   If
End Sub
 </SCRIPT>  <TITLE>  OnLoad</TITLE>
<BODY  BGCOLOR="#FFFFFF">
<P  ALIGN="center"><FONT  COLOR="#FF0000" SIZE="4">Событие</FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
<EM><STRONG>OnLoad</STRONG></EM></FONT>
<FORM  METHOD="POST">
<P ALIGN="center"><INPUT TYPE="button"
NAME="TestB"  VALUE="Кнопка">
 </FORM>
</BODY> </HTML>
```

OnUnLoad. Данное событие возникает при выгрузке документа из окна (листинг 10, рис.7).

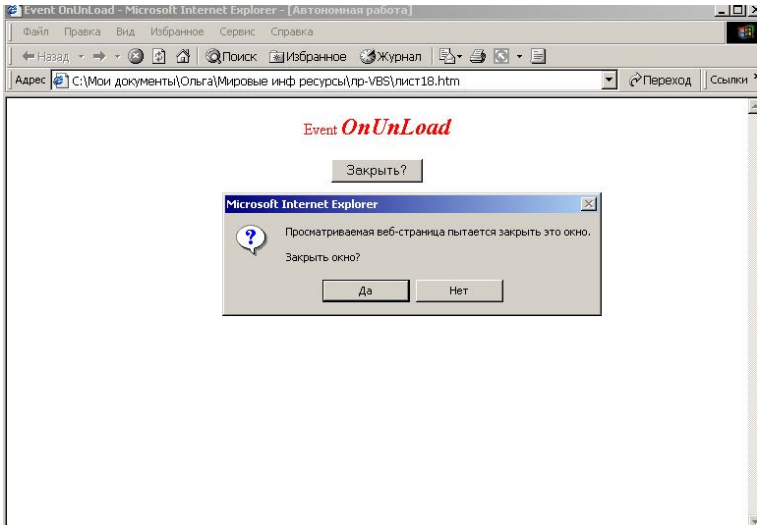

Рис. 7. Стандартная обработка события *OnUnLoad*

Листинг 10. *Стандартная обработка события OnUnLoad*

```
<HTML> <HEAD>
<SCRIPT   LANGUAGE="VBScript">
Sub Window_OnUnLoad
       Alert  "До свидания!!!!"
End  Sub
 Sub   TestB_OnClick
Window.Close
 End  Sub
</SCRIPT>
<TITLE>Event  OnUnLoad</TITLE> </HEAD>
<BODY  BGCOLOR="#FFFFFF">
 <P ALIGN="center"><FONT  COLOR="#FF0000'
SIZE="4">Event</FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
<EM><STRONG>OnUnLoad</STRONG></EM></FONT>
<FORM METHOD="POST">
<P ALIGN="center"><INPUT   TYPE="button" NAME="TestB"   VAL-
UE="Закрыть?">
</p>
</FORM>
</BODY>
</HTML>
```

Объект Frame

Объект *Frame* − это индексированный массив из фреймов (кадров) на странице. Первый кадр из списка соответствует кадру в верхнем левом углу браузера. Вы можете использовать данный объект для установки или получения различных URL-адресов разных фреймов, которые находятся на экране браузера. Фреймовый массив очень похож на объект *Window*, по-

скольку он использует другие объекты похожим способом. Например, вы можете использовать расположение объекта для получения или установки соответствующего фрейма.

Объект *Frame* подобен объекту *Window*, являясь, по существу, окном в окне. Поэтому данный объект имеет события, методы и свойства, уже описанные для объекта *Window*. Единственной темой, требующей дополнительного рассмотрения, является синтаксис ссылки из одного фрейма на другой.

Трудность заключается в том, что необходимо правильно указать порядок прохождения свойств объектов. Например, в приведенном примере (листинг 11) в окне браузера существует два фрейма − верхний и нижний. Имя верхнего − TopFrame, имя нижнего − BottomFrame. Для того чтобы грамотно сослаться из верхнего текущего фрейма на нижний, скажем, для получения доступа к одному из свойств этого фрейма, следует сначала сослаться на родительский объект верхнего фрейма – Window, а затем на нижний фрейм. Это достигается, например, такой строкой:

Top.FrameBottom… или Parent.FrameBottom. ..

Top.Frames(1)... − ссылка все на тот же нижний фрейм.

Объект Location

Данный объект хранит информацию о текущем URL.

Href. Данное свойство определяет текущий URL.

Рассмотрим пример окна в виде двух фреймов. В этом примере задача сводится к тому, чтобы при щелчке по кнопке содержимое фреймов менялось местами, т.е. содержимое верхнего фрейма помещалось в нижний и наоборот и т.д. (рис.8).

Для определения того, какой документ должен отображаться в том или ином фрейме, использована несложная составная строковая функция, приведенная ниже.

Left(Right(Top.Frames(0).Location.Href,5),1)="1"

Строка Top.Frames(0).Location.Href вернет строку текущего URL. Функция Right в приведенном примере вырезает строку из пяти символов справа: N.htm, где N – последняя цифра имени файла ("1" или "2" в зависимости от того, какой документ загружен – Frame1.htm или Frame2.htm).

Листинг 11. Установочный Index.htm

```
<HTML>
<HEAD>
<TITLE>Example</TITLE>
</HEAD>
<FRAMESET   Rows=50%,50%>
<FRAME NAME="TopFrame"  SRC="Frame1.htm">
<FRAME NAME="BottomFrame"   SRC="Frame2.htm">
</FRAMESET>
</HTML>
```

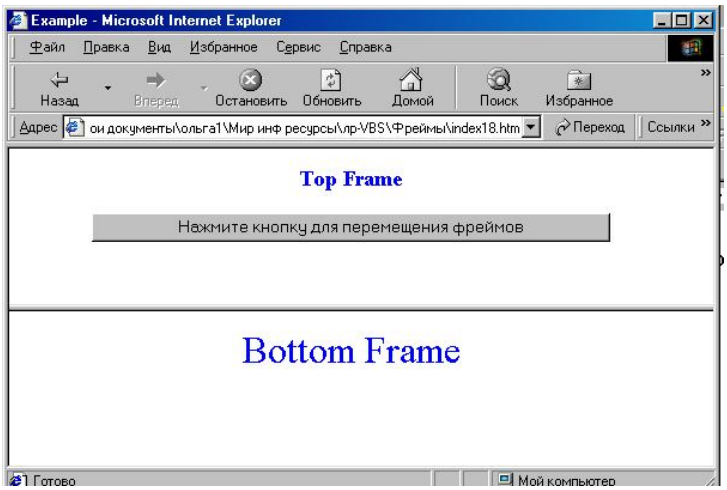

Рис. 8. Применение объектов *Frame* и *Location*

Листинг12. *Frame2.htm*

```
<html>
<HEAD>
<TITLE>Bottom Frame</TITLE> </HEAD>
<BODY  BGCOLOR="white">
<P  align="center">
<Font color="Blue"   Size="6"><Strorig>
Bottom Frame</Strong><font>
</p>
 </BODY>
</HTML>
```

Листинг 13. *Frame1.htm*

```
<html>
<HEAD>
<TITLE>Top  Frame</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<SCRIPT LANGUAGE="VBScript">
Sub TestB_OnClick
If Left(Right(Top.Frames(0).Location.Href,5),1)="1" Then
Top.BottomFrame.Location.Href ="Frame1.htm"
Top.TopFrame.Location.Href = "Frame2.htm"
Else
      Top.BottomFrame.Location.Href ="Frame2.htm"
      Top.TopFrame.Location.Href ="Frame1.htm"
   End  If
End Sub
 </SCRIPT>
<P ALIGN="center"> <FONT   COLOR="Blue"   SIZE="4"><STRONG>
Top Frame
</STRONG></FONT>
</p>
<FORM>
<P ALIGN="center">
<INPUT  TYPE="button"  NAME="TestB"
VALUE="Нажмите кнопку для перемещения фреймов">
</p>
</FORM>
</BODY>
</HTML>
```

Объект History

Основной целью объекта *History* является доступ к списку введенных адресов в браузере. Существуют три метода, используемые для навигации по папке *History*. Наиболее важные методы перечислены в табл. 5. Объект *History* не генерирует никаких событий.

Таблица 5

Некоторые методы объекта *History*

Метод	Назначение
Go	Используется для того, чтобы определить, сколько раз браузер должен пролистать вперед историю введенных адресов. Формат вызова: history.go(n), где n - номер файла истории, на который нужно перейти

Окончание таблицы 5

Forward	Используется для определения того, сколько раз браузер должен пролистать вперед историю введенных адресов. Формат вызова: history.forward(n), где n - число раз, которое нужно "идти вперед"
Back	Используется, чтобы определить, сколько раз браузер должен "идти назад" по текущей истории вашего браузера. Формат вызова: history.back(n), где n - число раз, которое браузер должен "идти назад"

Пример работы с фреймами и объектом *History*.

При нажатии кнопки «*Новая Web-страница*» на экране появляется диалоговое окно (рис. 9), в котором необходимо ввести адрес страницы (в нашем случае страницы находятся в одном каталоге). Содержание загруженной страницы помещается в нижний фрейм. Кнопки *Вперед*, *Назад* используются для навигации по папке *History* (рис. 10).

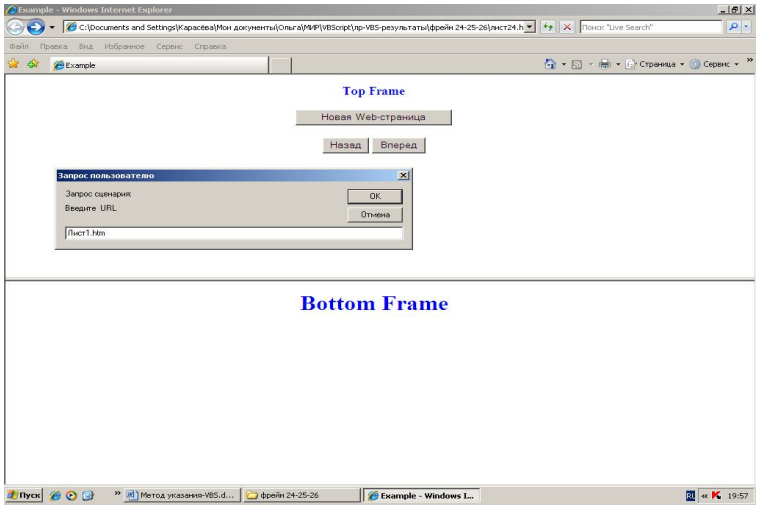

Рис. 9. Диалоговое окно после нажатия кнопки *Новая Web-страница*

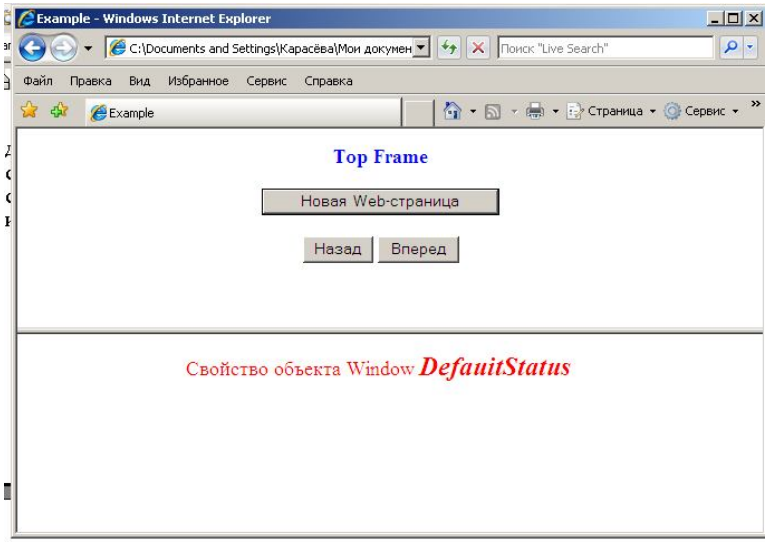

Рис.10. Результат выполнения листинга 14

Листинг 14. Индексный файл (*index.htm*)

```
<HTML>
<HEAD>
<TITLE>Example</TITLE>
</HEAD>
<FRAMESET   Rows=50%,50%>
<FRAME NAME="BottomFrame"  SRC="FrameControl.htm">
<FRAME NAME="TopFrame"   SRC="FrameWindow.htm">
</FRAMESET>
</HTML>
```

Листинг 14-1.Верхний фрейм (*FrameWindow.htm*)

```
<HTML>
<HEAD>
<TITLE>Top Frame</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<SCRIPT LANGUAGE="VBScript">
Sub NewB_OnClick
Str = Prompt("Введите  URL","Timer.htm")
if Str <> Empty Then
      Top.BottomFrame.Location.Href = Str
End If
End Sub
Sub ForwardB_OnClick
Top.BottomFrame.History.Forward 1
End Sub
 Sub BackB_OnClick
Top.BottomFrame.History.Back 1
End  Sub
</SCRIPT>
 <P  ALIGN="center">
<FONT  COLOR="Blue"   SIZE="4"><STRONG>
Top  Frame
</STRONG></FONT>
</P>
<FORM>
<P ALIGN="center">
<INPUT   TYPE="button" NAME="NewB" VALUE="Новая  Web-страница">
</P>
<P ALIGN="center">
<INPUT TYPE= "button" NAME="BackB" VALUE="Назад">
<INPUT TYPE="button" NAME="ForwardB" VALUE="Вперед">
</P>
</FORM>  </BODY> </HTML>
```

Листинг 14-2. Нижний фрейм(*FrameControl.htm*). В этот фрейм загружается страница с указанным адресом URL.

```
<HTML>
<HEAD>
<TITLE>Нижний фрейм</TITLE>
</HEAD>
```

```
<BODY BGCOLOR="white">
<P ALIGN="center">
<FONT  COLOR="Blue"  SIZE="6"><STRONG>
Bottom  Frame</STRONG></FONT>
</P>
</BODY> </HTML>
```

Объект Document

Объект *Document* имеет дело прежде всего с телом HTML-страницы. Он имеет три дочерних объекта: *Link, Anchor и Form*, каждый из которых является индексированным массивом объектов *Link, Anchor и Form*. Кроме того, объект *Form* содержит подобъект *Element*, который является индексированным массивом всех объектов и элементов управления на странице. Некоторые важные свойства и методы перечислены в табл. 6. Для *Document*, заметим, не существует никаких событий.

Таблица 6

Некоторые методы объекта *Document*

Метод	Назначение
BgColor	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление #rrggbb или соответствующее название
FgColor	Устанавливает цвет текста документа. Аналогичен по функциям свойству BgColor
Referrer	Указывает URL документа, на который ссылается пользователь в настоящее время. Например, если кто-то обратился по адресу: http://www.nm.org/welcome.htm с сервера http://www.someplace.com, то свойством Referrer будет: http://www.someplace.com, если это свойство было в странице вышеупомянутого расположения; в противном случае оно обращается в Null
LastModified	Показывает дату последней модификации документа
Open	Открывает документ для записи дополнительных строк в формате HTML. Синтаксис: document.open()
Write	Записывает HTML-текст в текущий документ и должен вызываться, когда документ открывается для записи. Синтаксис: document.write(somestring), где somestring может быть одной строкой, переменной или же несколькими связанными строками в формате HTML, которые выводятся на экран
Close	Закрывает документ после того, как имели место следующие вызовы: document.write, document.close

Прежде чем перейти к рассмотрению свойств и методов объекта *Document*, следует отметить тот факт, что окно, как и фрейм, может содержать лишь один данный объект.

Для объекта *Document* определен довольно большой набор свойств, приведенных ниже.

Anchors. Данное свойство позволяет обратиться к объекту *Anchor,* являющемуся массивом гиперссылок, присутствующих в документе. Объект *Anchor* будет рассмотрен подробнее. Аналогичные функции ссылки на объекты выполняют свойства *Links* и *Forms*.

LastModified. Возвращает дату последнего изменения документа. Данное свойство предназначено только для чтения.

Location. Данное свойство позволяет получить URL текущего документа. По функциональным возможностям это свойство аналогично свойству *Href* объекта *Location*. Это свойство предназначено также только для чтения.

Title. Позволяет получить заглавие страницы, задаваемое тегом <TITLE> и отображаемое в строке заголовка окна браузера. Свойство может быть использовано только для чтения.

Объект *Document* обладает свойствами, отвечающими за цветовое отображение тех или иных элементов страницы.

В VBScript определены следующие стандартные цветовые константы: Aqua, Black, Blue, Fuchsia, Gray, Green, Lime, Maroon, Navy, Olive, Purple, Red, Silver, Teal, White, Yellow.

Следует отметить, что какой-либо цвет или оттенок цвета может быть задан при помощи специального RGB-формата. Название этого формата происходит от названий трех цветов Red, Green, Blue. Каждому из этих цветов должна быть поставлена в соответствие требуемая интенсивность. При задании интенсивности следует учитывать, что нулю соответствует

самая низкая интенсивность, а числу 255 − самая высокая. Интенсивность каждого из трех цветов указывается в шестнадцатеричном коде. Последовательно указав цветовые интенсивности каждого цвета, можно получить желаемый цвет или оттенок.

Например, "004A51". В данном примере цвету Red соответствует нулевая интенсивность, цвету Green - "4A", а цвету Blue -"51" в шестнадцатеричном коде.

BgColor. Данное свойство позволяет задать цвет фона документа.

FgColor. Позволяет задать цвет текста документа.

Необходимо отметить, что эти свойства дают возможность немедленно изменить цвет согласно вновь определенному. В примере, приведенном ниже (листинг 15, рис. 11), рассматривается страница с двумя кнопками. Одна кнопка изменяет цвет фона, а другая − цвет текста.

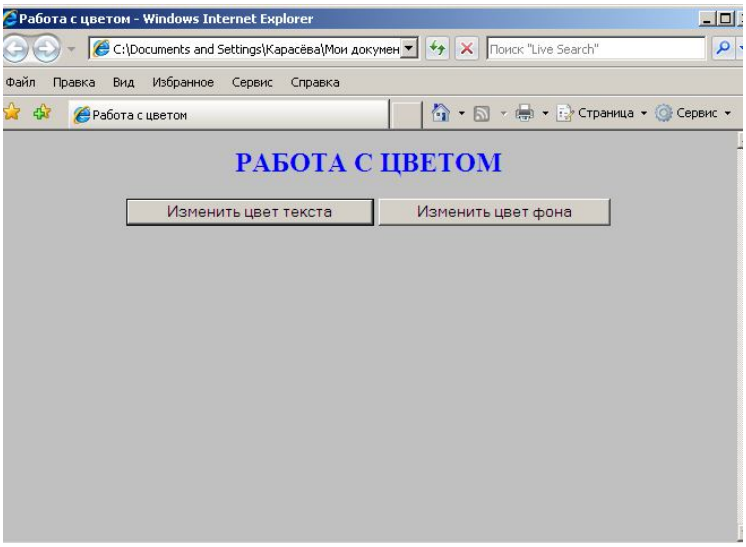

Рис. 11. Работа с цветом

Листинг 15. Использование свойств объекта *Document,* отвечающих за работу с цветом

```
<HTML>
<HEAD>
<TITLE> Работа с цветом </TITLE>
<SCRIPT LANGUAGE="VBSCRIPT">
DIM A(15)
A(1) = "WHITE"
A(2) = "BLACK"
```

```
   A(3) = "BLUE"
   A(4) = "RED"
   A(5) = "YELLOW"
   A(6) = "AQUA"
   A(7) = "FUCHSIA"
   A(8) = "GRAY"
   A(9) = "LIME"
   A(10) = "MAROON"
   A(11)= "NAVY"
   A(12)= "OLIVE"
   A(13)= "PURPLE"
   A(14)= "TEAL"
   A(14)= "Silver"
   I=1
   J=2
   DOCUMENT.BGCOLOR = A(I)
   DOCUMENT.FGCOLOR = A(J)
   SUB FOREB_ONCLICK
   IF I = 15 THEN I=0
   I = I + 1
   DOCUMENT.FGCOLOR = A(I)
   END SUB
   SUB GROUNDB_ONCLICK
   IF J=15 THEN J=0
   J = J + 1
   DOCUMENT.BGCOLOR = A(J)
   END SUB
   </SCRIPT>
    </HEAD>
    <BODY>
    <CENTER>
   <H2> РАБОТА С ЦВЕТОМ </H2>
    <FORM NAME="SET_COLOR">
    <INPUT TYPE="BUTTON" NAME="FOREB" VALUE="Изменить цвет
текста">
   <INPUT TYPE="BUTTON" NAME="GROUNDB" VALUE="Изменить цвет
фона">
   </P>
   </FORM>
   </CENTER>
   </BODY>
   </HTML>
```

Продолжим знакомство со свойствами объекта *Document.*

LinkColor. Служит для определения цвета гиперссылок документа.

ALinkColor. Определяет цвет, используемый для отображения активной гиперссылки. Активной считается та ссылка, на которую установлен указатель мыши, нажата левая кнопка мыши, однако еще не отпущена.

VlinkColor. Позволяет определитъ цвет для тех гиперссылок, которые уже выбраны в ходе работы.

Листинг 16 реализует простейший пример, отображающий значения свойств объекта *Document* no работе с цветами гиперссылок.

Листинг 16. Работа с цветами гиперссылок

```
<HTML> <HEAD>
<SCRIPT   LANGUAGE="VBScript">
Sub AlertB_OnClick
Alert   "LinkColor  =   "   + Document.LinkColor + chr(13) + chr(10) + _
"ALinkColor = " + Document.ALinkColor +  chr(13)   +   chr(10)   + "VLinkColor = " -
+ Document . VlinkColor
End  Sub
 </SCRIPT>
<TITLE>Color   Properties</TITLE>
 </HEAD>
<BODY  BGCOLOR="#FFFFFF">
<P ALIGN="center"><FONT  COLOR="#FF0000" SIZE="4">
Color  Link</FONT>
<FONT   COLOR="#FF0000"
SIZE="5"><EM><STRONG>
Proper ties</STRONG></EM></FONT>
<FORM METHOD="POST">
<P ALIGN=”center”><INPUT _
TYPE="button"  NAME=" AlertB "
VALUE="Показать значения">
</P>
</FORM>
 </BODY>
</HTML>
```

Помимо перечисленных свойств объекта *Document,* существуют возможности работы со специальным текстовым файлом, находящимся на стороне клиента и предназначенным для хранения небольших объемов справочной информации. За выполнение действий с этим файлом отвечает свойство *Cookie* рассматриваемого объекта.

Свойство *Referrer* позволяет получить URL, содержащийся в гиперссылке, выбранной пользователем для перехода на текущий документ. Данное свойство возвращает пустую строку, если переход к документу осуществляется не по гиперссылке, а при помощи URL, указанного в адресной строке.

Помимо свойств, обзор которых был представлен, объект *Document* обладает набором методов, которые удобно использовать для создания динамических документов, то есть создаваемых по ходу выполнения сценария. Это позволяет формировать более "живые" страницы, содержимое которых может выглядеть различным образом в зависимости от выполнения некоторых конкретных условий.

Методы объекта *Document* приведены ниже.

Open. Данный метод открывает буфер объекта *Document* для заполнения. Если метод *Open* не будет вызван, то метод *Write* будет формировать HTML-документ наряду с остальным существующим HTML-кодом. В том случае, если метод *Open* будет вызван, вся уже содержащаяся информация в документе будет перезаписываться при использовании метода *Write*.

Write. Метод позволяет добавить строку в документ. Поскольку добавляемая строка заносится без какого бы то ни было форматирования, в нее могут быть включены теги языка HTML. Например:

```
Document. Write  "<H2> Пример  </H2>"
```

Writeln. Данный метод позволяет выполнить действия, аналогичные действиям, выполняемым при вызове метода *Write*, с той разницей, что в конец строки будет добавлен символ возврата каретки. Однако почти всегда на практике HTML не различает символы перевода строки, а потому действия, выполняемые методами *Write* и *Writeln* можно считать аналогичными.

Close. Метод позволяет окончить запись в объект *Document*. Непосредственно после вызова данного метода сформированный документ будет отображен браузером.

Листинг 17 содержит код страницы, иллюстрирующей использование метода *Write*.

Листинг 17. Работа с методом *Write* объекта *Document*

```
<HTML>
<HEAD>
<TITLE>Example</TITLE>
<SCRIPT  LANGUAGE="vbscript">
Document.Write   "<P ALIGN=center>"
Document.Write   "<FONT   COLOR='Blue'"
Document.Write   “SIZE= '7 ' ><STRONG>NEW TEXT”
Document.Write   "</STRONG></FONT>"
Document.Write   "</P>"
Document. Close
</SCRIPT>
</HEAD>
<BODY  BGCOLOR="white">
<P  ALIGN="center"><FONT   COLOR="#FF0000"
SIZE="4">Methods   of  object  </FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
<EM>< STRONG>Document</STRONG></EM></FONT>
</P>
</BODY>
</HTML>
```

В отличие от приведенного листинга, описывающего включение метода *Write* в формирование Web-страницы, следующий пример (листинг 18, рис. 12) позволяет создать новую страницу. Это достигается посредством использования метода *Open*. Чтобы браузер отобразил новую страницу, достаточно лишь нажать на приведенную кнопку.

Листинг 18. Использование метода *Open* в формировании новой страницы

```
<HTML>
 <HEAD>
<TITLE>Example</TITLE>
<SCRIPT  LANGUAGE="vbscript">
 Sub  TestB_OnClick
Document.Open
Document.Write   "<P ALIGN= center >"
```

```
Document.Write   "<FONT  COLOR= ' Blue ' "
Document.Write   "SIZE= '7' ><STRONG>А вот и текст!"
Document.Write   "</STRONG>"
Document.Write   "</FONT>"
Document.Write   "</P>"
Document.Close
 End  Sub
</SCRIPT>
</HEAD>
<BODY  BGCOLOR="white">
<P ALIGN="center"><FONT COLOR="#FF0000"
SIZE="4">Метод объекта  </FONT>
<FONT   COLOR="#FF0000"   SIZE="5">
 <EM><STRONG>Document</STRONG></EM></FONT>
</P>
<P ALIGN="center"><INPUT TYPE="button"
 NAME="TestB" VALUE="Нажмите кнопку для демонстрации метода">
</BODY>
</HTML>>
```

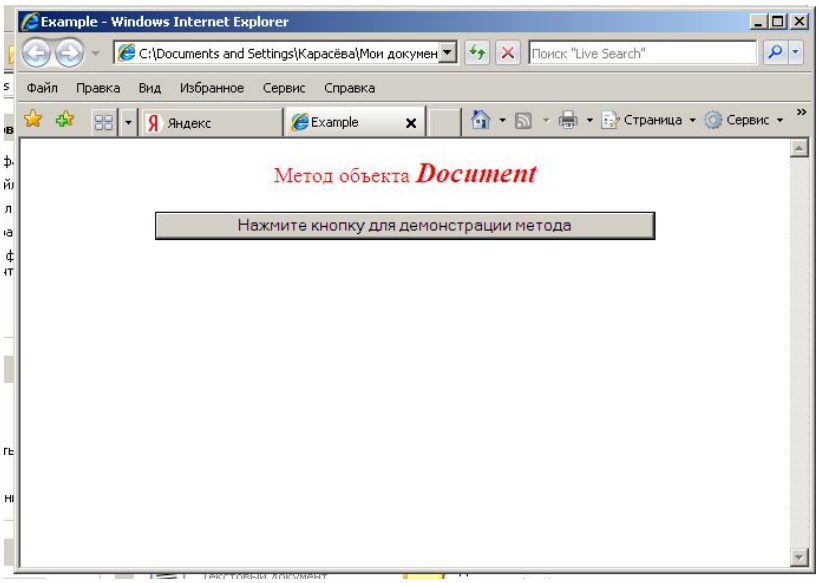

Рис. 12. Начальное состояние страницы

Объект Element

Объект *Element* включает встроенные элементы управления HTML. Данный объект обычно является дочерним по отношению к объекту *Form*. В принципе, размещение встроенных элементов управления возможно и вне формы. Однако в этом случае свойства элементов будут недоступны, несмотря на тот факт, что браузер корректно отобразит каждый элемент. Таким образом, обычно на практике объект *Element* имеет родительский

объект *Form,* что позволяет получить доступ к свойствам встроенных элементов управления HTML. Следует отметить, что элементы управления ActiveX не имеют ограничения на расположение. Это означает, что их свойства доступны независимо от места расположения элементов.

К объекту *Element* можно обращаться, используя имя или по порядковому номеру элемента в массиве *Elements*. Использование обращения к объектам по их номерам может быть удобным при переборе большого количества однотипных элементов. В этом случае целесообразно воспользоваться циклом, в котором для обращения к элементам достаточно лишь перебирать их порядковые номера.

Чтобы определить количество элементов в массиве *Elements* достаточно воспользоваться свойством *Count*. В этом случае обращение к свойству *Count* может выглядеть, например, таким образом:

```
Count_of_Elements = Document.Form(0).Elements.Count
```

Как уже говорилось выше, объект *Element* представляет собой массив встроенных элементов управления. Порядок следования элементов в массиве соответствует порядку загрузки элементов в браузер, то есть очередности объявлений в HTML-коде.

Element является дочерним объектом по отношению к объекту *Form*. Каждая форма имеет лишь один объект *Element.* Для обращения к этому объекту следует задать строку, аналогичную приведенной ниже:

```
Window.Document.TestForm.Elements...
```

Следует учитывать, что к каждому элементу массива встроенных объектов можно обращаться непосредственно по имени. Например:

```
Window.Document.TestForm.TestButton...
```

Объект *Element* обладает двумя свойствами. Свойство *Length* позволяет определить количество элементов в массиве объекта *Element*. Нумерация элементов начинается с нуля.

Свойство *Count* полностью идентично свойству *Length*.

Всего существует десять встроенных элементов управления HTML. Изучению работы с данными элементами посвящен этот раздел.

HTML поддерживает использование кнопок, зависимых и независимых переключателей, полей ввода, списков выбора. Почти все они определяются посредством тега <INPUT>.

Кнопки

Язык HTML поддерживает кнопки трех разновидностей, которые, тем не менее, обладают одинаковыми свойствами, методами и событиями.

Различают следующие виды кнопок:

- обычная кнопка (*Button*);
- кнопка сброса значений элементов формы (*Reset*);
- кнопка инициации передачи данных из формы на сервер (*Submit*).

Все кнопки поддерживают свойства, приведенные ниже.

Form. Очень гибкое свойство, позволяющее сослаться на родительскую форму и получить доступ к ее свойствам. Например:

```
Window.Document.Testform.TestButton.Form.Method = "Get"
```

Name. Данное свойство позволяет определить имя объекта, заданное посредством атрибута *Name*. В случае, если при описании кнопки имя не было определено, данное свойство возвращает пустую строку.

Здесь необходимо отметить следующую особенность. Хотя свойство *Name* и предназначено для чтения, существует возможность его программного изменения. Однако если имя элемента изменить в сценарии, то все обработки событий останутся привязанными к старому имени. Здесь и проявляется ограниченность действий со свойством *Name*. Интерпретатор уже откомпилировал программу, а это означает, что вызываться будет один и тот же обработчик события, несмотря на возможные изменения свойства *Name*.

Value. Данное свойство определяет надпись, отображаемую на кнопке. Это свойство доступно как для чтения, так и для редактирования.

Рассматриваемые кнопки поддерживают лишь одно событие *OnClick*, которое генерируется при щелчке пользователя на кнопке, и один метод *Click*, позволяющий программно имитировать щелчок. Очень важной особенностью является тот факт, что вызов метода *Click* не генерирует события *OnClick*, а значит, обработчик данного события не вызывается. Таким образом, если существует необходимость программно имитировать щелчок на кнопке с исполнением обработчика события *OnClick*, нужно указывать следующий код:

```
TestButton_OnClick
Document.TestForm.TestButton.Click
```

Если же, например, для кнопки типа *Submit* вызывать только метод *Click*, то данные из формы будут переданы на сервер, но обработчик события *OnClick* запускаться не будет.

В примере (листинг 19, рис. 13) рассматривается Web-страница, содержащая три кнопки и одно поле ввода.

Листинг 19. Работа с кнопками.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
 Sub TestB_OnClick
      NL = chr(13) + chr(10)
      Str = "Имя кнопки: "
      Str = Str & Document.TestForm.Tests. Name & NL
      Str = Str & " Элементов формы: "
      Str = Str & Document.TestForm.Tests.Form.Elements.Length & NL
      Str = Str & "Надпись: "
      Str = Str & Document.TestForm.TestS.Value
      x = MsgBox(str,0, "Test Button Properties")
End Sub
 Sub TestBV_OnClick
      Document.TestForm.Tests.Value = Document.TestForm.CaptionT.Value
End Sub
Sub TestS_OnClick
      Alert "Вы нажали на кнопку Submit"
End Sub
</SCRIPT>
</HEAD>
<BODY BGCOLOR="White">
<FORM NAME="TestForm">
```

```
<P ALIGN="center">
<INPUT TYPE="text"NAME="CaptionT"Value="">
<INPUT TYPE="button"NAME="TestBV"
Value="Изменить надпись на кнопке Submit">
<P ALIGN="center">
<INPUT TYPE="button"NAME="TestB"
Value= "Показать свойства кнопки и формы">
</P>
<P ALIGN="center">
<INPUT TYPE="submit"NAME="TestS"Value="Submit">
</P>
</FORM>
 </BODY>
 </HTML>
```

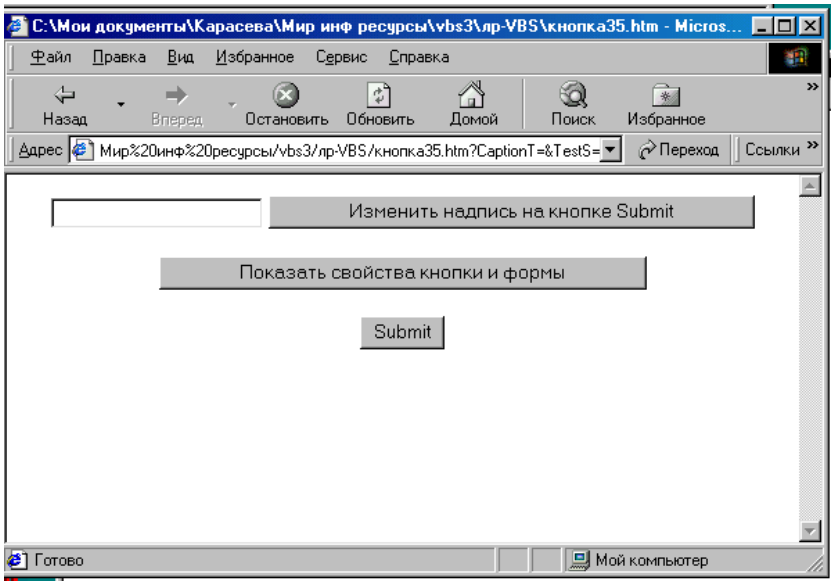

Рис. 13. Листинг 19. Работа с кнопками

Переключатели

Различают два вида переключателей − зависимые *Radio Buttons* и независимые *Checkbox*. Независимый переключатель, который также называется флажком, представляет собой небольшой квадратный элемент, внутри которого может быть помещена галочка, что соответствует состоянию *True* и означает, что флажок установлен. Отсутствие галочки говорит о том, что флажок сброшен, и соответствует значению *False*.

Независимые переключатели имеют свойства, приведенные ниже.

Form. Позволяет обратиться к родительской по отношению к переключателю форме, что дает возможность получить доступ к ее свойствам.

Name. Данное свойство возвращает имя, определенное атрибутом NAME тега <INPUT>. Если атрибут *Name* не задан, то при обращении к свойству *Name* будет возвращена пустая строка. Данное свойство предназначено только для чтения.

Value. Это свойство возвращает строку, определенную атрибутом VALUE тега <INPUT> и доступно как для чтения, так и для редактирования.

Checked. Данное свойство является ключевым, поскольку задает состояние флажка. Значение свойства имеет тип *Boolean*. Установленному флажку соответствует *True*, а сброшенному − *False*. *Checked* можно использовать не только для чтения. Установленное программно, оно является эквивалентом щелчка пользователя на флажке.

Как и кнопочные встроенные элементы, флажок поддерживает одно событие − *OnClick* и один метод − *Click*. Событие *OnClick* возникает при щелчке пользователя на флажке. При вызове метода *Click* это событие не генерируется. Сам флажок меняет свое положение на противоположное. Например, приведенный ниже фрагмент кода меняет состояние флажка на обратное без вызова обработчика события *OnClick*:

```
Document.TestForm.TestCheck.Click
```

В примере (листинг 20, рис. 14) продемонстрирована работа с независимыми переключателями. На Web-странице приведены семь флажков и три кнопки. При выборе первой кнопки *Set Check Boxes in True* все переключатели примут значение *True*. Если пользователь нажмет вторую кнопку *Set Check Boxes in False,* флажки сбросят свои значения. В случае если пользователь установит лишь некоторые из переключателей в положение "включен", нажатие третьей кнопки *Find selected Check Boxes* позволит вывести окно сообщения, отображающее все выбранные пользователем переключатели.

Листинг 20. *Работа с независимыми переключателями*

```
<HTML>
<HEAD>
   <SCRIPT LANGUAGE="VBScript">
   Sub TestS_OnClick
      For x = 0 to 6
      Document.TestForm.Elements(x).Checked = True
      Next
     End Sub
      Sub TestC_OnClick
            For x = 0 to 6
            Document.TestForm.Elements(x).Checked  = False
            Next
      End Sub
      Sub TestF_OnClick
            NL = chr(13) + chr(10)
            Str = " Выбраны переключатели:" + NL
            For x=0 to 6
                  If  Document.TestForm.Elements(x).Checked  Then
                  Str = Str + Document.TestForm.Elements(x). Value + NL
                  End  If
            Next
      Alert  Str
      End Sub
       </SCRIPT> </HEAD>
      <BODY BGCOLOR="white"> <FORMNAME="TestForm">
       <P ALIGN="center">
      <TABLE>
      <TD><Option One<TD><INPUTTYPE ="checkbox"
      NAME="cbl"  VALUE="One"><TR>
       <TD><Option   Two<TD><INPUT TYPE ="checkbox"
       NAME="cb2" VALUE="Two"><TR>
       <TD><Option  Three<TD><INPUT  TYPE="checkbox”
      NAME="cb3"  VALUE="Three"><TR>
      <TD><Option Four<TD><INPUT TYPE ="checkbox"
      NAME="cb4"  VALUB="Four"><TR>
       <TD><Option  Five<TD><INPUT TYPE ="checkbox"
      NAME="cb5"  VALUE="Five"><TR>
      <TD>Option  Six<TD><INPUT TYPE ="checkbox"
      NAME="cb6"  VALOE="Six"><TR>
      <TD><Option   Seven<TD><INPUT TYPE = "checkbox"
       NAME="ob7" VALUE="Seven"><TR>
       </TABLE>
       </P>
      <P ALIGN="center">
      <INPUT TYPE ="button"NAME="TestS"
      VALUE=" Установить все переключатели в положение True">
      <INPUT TYPE="button" NAME="TestC"
```

```
VALUE=" Установить все переключатели в положение in
False">
<INPUT TYPE ="button"NAME="TestF"
VALUE="Показать выбранные переключатели ">
</P>
</FORM>
 </BODY>
</HTML>
```

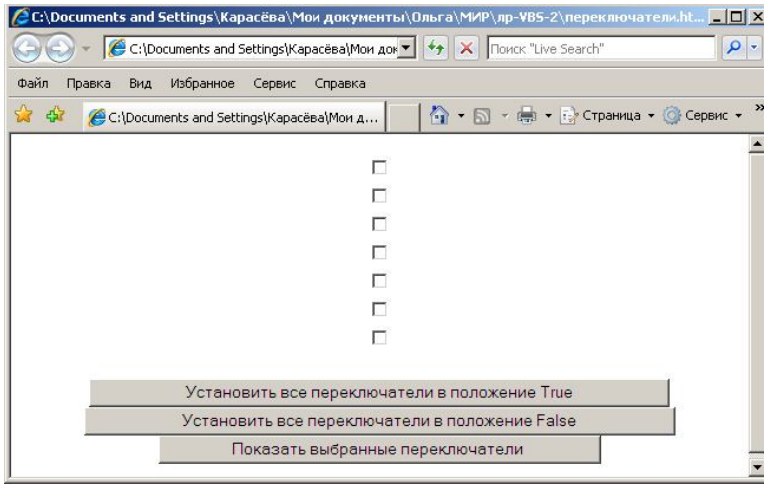

Рис. 14. Независимые переключатели

Под ***зависимыми*** переключателями понимают группу небольших элементов в виде окружностей, лишь один из которых может быть выбран. Это означает, что, выбирая тот или иной элемент из группы зависимого переключателя, пользователь исключает все другие. В случае, когда пользователю должна быть предоставлена возможность выбора нескольких вариантов, следует использовать флажки.

Чтобы предоставить браузеру информацию о том, какие элементы входят в состав группы зависимых переключателей, им следует задать одинаковые имена. Выбранный пользователем элемент сразу становится активным, а переключатель, который был активным прежде, также изменит свое состояние − станет пассивным.

Следует отметить, что, ввиду наличия нескольких одинаковых имен у зависимых переключателей, к ним невозможно обращаться как к элементам, рассмотренным выше. Это привело бы к неоднозначности, связанной

с неконкретностью обращения. Поэтому при работе с зависимыми переключателями с ними нужно оперировать как с элементами массива.

Чтобы определить, какой из переключателей установлен, необходимо их проверить поочередно.

Зависимые переключатели имеют те же свойства, что и независимые.

Checked. Дает возможность определить, выбран ли переключатель. Это свойство имеет тип *Boolean*, оно доступно как для чтения, так и для изменения. Необходимо отметить, что для программного выбора другого переключателя ему необходимо установить значение *True* в свойство *Checked*. Это приведет к автоматическому присваиванию значения *False* свойству *Checked* всех других переключателей. Однако при изменении значения переключателя возможно возникновение ошибки, связанное с его непреднамеренным включением. Дело в том, что *VBScript* устанавливает переключатель в состояние *True* при любой попытке присвоить свойству *Checked* какое-либо значение. Эту особенность следует учитывать при изменении положения переключателей. В то же время определение значения свойства *Checked* не отражается на состоянии переключателя. Это означает, что строка

```
Document.TestForm.Elements(0).Checked=False
```

приводит к включению переключателя в отличие от строки

```
str=Document.TestForm.Elements(0).Checked
```

По аналогии с независимыми переключателями зависимые поддерживают только одно событие − *OnClick* и один метод − *Click*.

При программном вызове метода *Click* событие *OnClick* не генерируется. Это событие возникает при щелчке на переключателе.

Следует отметить, что при работе с зависимыми переключателями обработчик события *OnClick* нельзя называть по совокупности имени элемента и названия события. Это связано с тем, что зависимые переключатели одной группы имеют общее имя. Вместо этого обработчик события

OnClick должен быть выполнен по аналогии с примером, приведенным в листинге 21. При выборе переключателя происходит смена фонового цвета. Нажатие кнопки *Default* приводит к возврату страницы к исходному виду. Параметр IndexR передается обработчику и позволяет определить, на каком переключателе был выполнен щелчок.

Листинг 21. Работа с зависимыми переключателями. Смена фонового цвета (рис.15).

```
<HTML> <HEAD>
<SCRIPT LANGUAGE="VBScript">
 Sub RadioColor(indexR)
      If  indexR =  0then Document.BgColor = "Red"
      if  indexR = 1   then Document.BgColor = "Silver"
      if  indexR = 2 then Document.BgColor =  "Green"
End Sub
Sub DefaultB_OnClick
      Document.BgColor =   "White"
      Document.Forms(0).Elements(0).Checked =  True
 End Sub
</SCRIPT>
<TITLE> Radio Buttons </TITLE>
</HEAD>
<BODY BGCOLOR="White">
<FORMMETHOD="POST">
<P ALIGN="center">
<FONT COLOR="Blue"SIZE="6">  Color of background
</FONT>
<P ALIGN="center"> <INPUT TYPE="radio" ONCLICK="RadioColor(0)"
CHECKED NAME="ColorR"VALUE="V1">
<FONT SIZE="4">Red</FONT>
<BR> <INPUT TYPE="radio" ONCLICK="RadioColor (1)"
NAME="ColorR"VALUE="V2">
<FONT SIZE="4">Silver</FONT>
 <BR> <INPUT TYPE="radio"ONCLICK="RadioColor(2)"
NAME="ColorR"VALUE="V3">
<FONT SIZE="4">Green</FONT>
<P ALIGN="center">
<INPUT  TYPE="button"NAME= "DefaultB"VALUE= "Default">
</P>
</FORM>
 </BODY>
 </HTML>
```

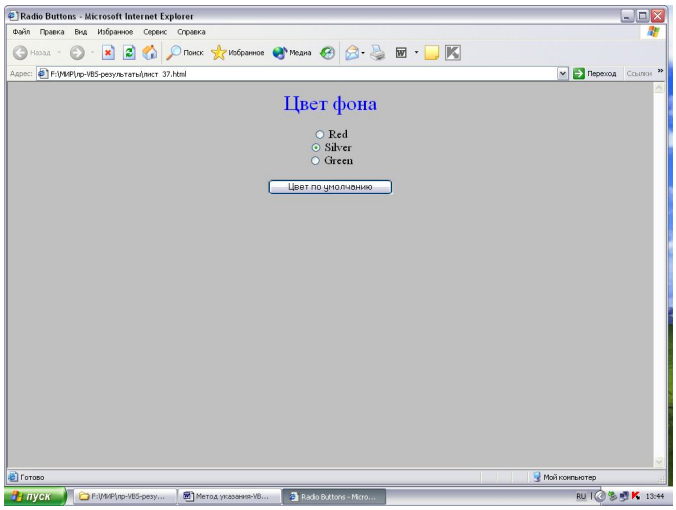

Рис. 15. Смена фонового цвета

Поля ввода

Свойства элементов управления, предназначенных для ввода текста, приведены ниже.

Form. Данное свойство позволяет обратиться к родительской форме, что дает возможность использования свойств объекта *Form.*

Name. Возвращает имя, определенное атрибутом *Name* тега <INPUT>. Свойство предназначено для чтения.

Value. Значение данного свойства доступно как для чтения, так и для редактирования. Оно содержит текст, заключенный в поле ввода текста. Следует отметить, что начальное положение области редактирования определяет тег <TEXTAREA> в отличие от полей редактирования и пароля, начальное значение которых задается посредством использования тега <INPUT>, а именно его атрибута *TEXT*. Поскольку данное свойство доступно для модификации, пользователь имеет возможность программно влиять на содержимое элемента ввода текста. При работе с полем пароля следует иметь в виду тот факт, что свойство *Value* будет хранить введенный текст, а не звездочки, которыми заменяются вводимые символы.

С рассматриваемыми элементами управления ввода текста связаны события *OnBlur*, *OnChange*, *OnFocus*, *OnSelect.*

OnBlur. Данное событие генерируется при потере фокуса элементом управления, то есть тогда, когда он перестает быть активным.

OnChange. Генерирование этого события происходит в момент потери фокуса элементом, если значение свойства *Value* было изменено. Следует отметить, что при программной модификации этого свойства событие *OnChange* не возникает. *OnChange* удобно использовать для проверки ввода пользователя.

OnFocus. Данное событие генерируется при активизации элемента, то есть в момент получения им фокуса ввода. При программной активизации событие *OnFocus* не возникает, оно является реакцией на действия пользователя.

OnSelect. Это событие возникает при выделении текста в элементе управления посредством метода *Select*. События не генерируются при выделении текста пользователем.

Следует особо отметить тот факт, что поле пароля не поддерживает никакое из перечисленных событий.

Элементы управления, предназначенные для ввода текста, поддерживают применение трех методов, приведенных ниже.

Blur. Вызов этого метода приводит к потере элементом фокуса. Однако при этом никоим образом не отслеживается элемент, который его приобретает. Поэтому с практической точки зрения легче сделать активным какой-либо другой элемент, чтобы текущий утратил фокус.

Focus. Передает фокус неактивному элементу без генерирования события *OnFocus*.

Select. Метод позволяет выделить текст, который содержится в поле ввода встроенного элемента управления, если тот является активным. Если элемент не имеет фокуса ввода или не содержит текстовой информации, то данный метод игнорируется.

В приведенном примере (листинг 22, рис. 16) реализуется следующая Web-страница. Если пользователь вводит текст в поле *Область ввода текста,* а затем нажимает кнопку *Добавить в текстовую область,* введенный текст заносится в поле ввода, расположенное в верхней части страницы.

При нажатии кнопки *Новая страница* происходит переход в области ввода текста на следующую строку. Нажатие кнопки *Выделить текст* приводит к выделению всего введенного текста. Поле ввода пароля *Область скрытого текста* дает возможность ввести текст, который отображается в области редактирования текста при возникновении события *OnChange*.

Листинг 22. Работа с полями ввода текста.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="VBScript">
Sub TestB1_OnClick
Document.TestForm.TestTA.Value = Document.TestForm.TestTA.Value + -
Document.TestForm.TestT1.Value
Document.TestForm.TestT1.Value = " "
End Sub
Sub TestB2_OnClick
Document.TestForm.TestTA.Value = Document.TestForm.TestTA.Value + -
chr(13) + chr(10)
End Sub
Sub TestB3_OnClick
Document.TestForm.TestTA.Select
End Sub
Sub TestT2_OnChange
Document.Testporm.TestTA.Value = Document.TestForm.TestTA.Value + -
Document.TestForm.TestT2.Value
Document.TestForm.TeatT2.Value = " "
End Sub
 </SCRIPT>
 </HEAD>
<BODY BGCOLOR="white">
 <FORM NAME="TestForm"> <P ALIGN= "center">
<TEXTAREA ROWS="10"COLS="30"NAME="TestTA">
</TEXTAREA>
</P>
<P ALIGN="center">
<INPUT TYPE="text" NAME="TestT1">
<FONT COLOR="Blue"SIZE="3">
 <STRONG>Область ввода текста</STRONG></FONT><BR>
<INPUT TYPE="password" NAME="TestT2">
```

```
<FONT COLOR="Blue" SIZE="3">
 <STRONG>Область скрытого текста</STRONG> </FONT> <BR>
</P>
<P ALIGN="center">
<INPUT TYPE="button" NAME="TestB1" Value="Добавить в текстовую
область">
<INPUT TYPE="button" NAME="TestB2" Value="Новая строка">
<INPUT TYPE="button" NAME="TestB3" Value="Выделить текстовую
область">
</P>
</FORM>
</BODY>
</HTML>
```

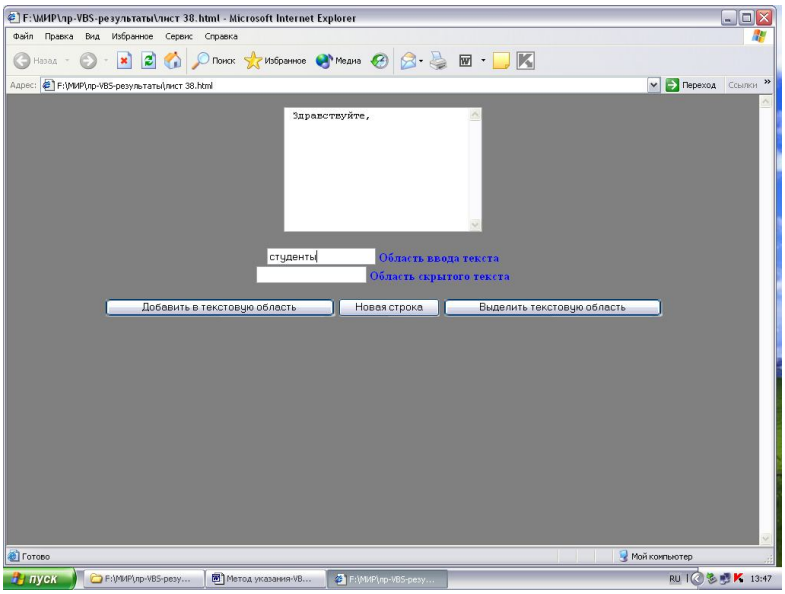

Рис. 16. Работа с полями ввода текста

Список выбора

Ниспадающий список выбора (*Select*) описывается посредством использования тега <SELECT>, причем каждый из доступных предлагаемых вариантов должен быть определен тегом <OPTION>.

Существует возможность создания списков, позволяющих выбирать несколько вариантов. Для этих целей следует использовать атрибут *Multiple* тега <SELECT>.

Еще одной особенностью, которую следует учитывать при работе с элементом *Select*, является факт статичности вариантов выбора. Это означает, что начальное определение элемента остается неизменным на протя-

жении всего сеанса работы. Невозможно вносить коррективы или изменять число предлагаемых вариантов.

Элемент управления *Select* имеет большой набор свойств, приведенных ниже. *Length.*

Form. Данное стандартное свойство элементов управления позволяет сослаться на родительскую форму, что обеспечивает доступ к ее свойствам.

Length. Свойство позволяет определить количество элементов, отображенных в списке выбора, совпадающее с числом тегов <OPTION> и количеством элементов в массиве *Options*, речь о котором пойдет ниже.

Данное свойство аналогично одноименному свойству объекта *Select*.

Name. Свойство возвращает имя элемента управления *Select*, заданное посредством атрибута *Name* его одноименного тега. Данное свойство предназначено для чтения и возвращает пустую строку, если атрибут *Name* отсутствует.

SelectIndex. Это свойство позволяет установить индекс выбранного элемента массива *Options*. Следует учитывать, что нумерация элементов начинается с нуля.

Options. Данное свойство позволяет сослаться на набор *Options*, который содержит массив элементов списка выбора. Иначе говоря, элементами массива являются предлагаемые варианты, каждый из которых заключен в теге <OPTION>.

Объект *Options*, в свою очередь, также имеет свои свойства.

DefaultSelected. Свойство служит для определения элемента, выбранного по умолчанию. HTML-код позволяет создать вариант выбора по умолчанию посредством атрибута SELECTED тега <OPT1ON>. Если при описании SELECTED не использовался, значит свойство *DefaultSelected* возвращает *False*, в противном случае *True*.

Selected. Значение этого свойства − нуль либо единица в зависимости от того, выбран данный элемент или нет. Это свойство очень удобно использовать при работе с возможностью множественного выбора. В этом случае, последовательно анализируя свойство *Selected* для каждого из элементов массива, можно выявить все отмеченные варианты.

В операциях сравнения значение свойства *Selected* можно рассматривать как *Boolean*, если только не прибегать к явному указанию значений *True* и *False*.

SelectedIndex. Данное свойство набора *Options* идентично одноименному свойству объекта *Select*. Однако необходимо учитывать, что при множественном выборе *SelectedIndex* возвращает лишь индекс первого элемента из выбранных.

Text. Свойство позволяет определить строковое значение, установленное тегом <OPTIONS>. Вообще говоря, именно из этих строк и состоит ниспадающий список выбора.

Два события, определенных над встроенным элементом управления *Select*, приведены ниже.

OnFocus. Генерируется при активизации списка выбора. Вызов метода *Focus* не приводит к возникновению события *OnFocus*.

OnChange. Данное событие генерируется при выборе любого другого элемента из ниспадающего списка.

Также элемент управления *Select* поддерживает два метода.

Blur. Позволяет деактивизировать элемент, то есть лишить его фокуса. Однако неизвестно, какой элемент станет активным после вызова данного метода. В связи с этим считается более логичным и последовательным передать фокус явно одному из неактивных элементов управления.

Focus. Данный метод позволяет передать фокус элементу *Select* без генерирования события *OnFocus*.

Приведенный ниже пример (листинг 23, рис.17) иллюстрирует использование списка выбора при работе простейшего калькулятора. Для работы калькулятора заведены три поля ввода, один список выбора и две кнопки. Два первых поля ввода позволяют задать необходимые операнды. Список выбора дает возможность указать требуемую операцию. Первая кнопка, *Calculate*, обеспечивает вывод результата вычислений в третьем поле ввода. Кнопка *Clear* приводит форму в исходное состояние.

Листинг 23. *Использование списка выбора (Калькулятор)*

```
<HTML>
<HEAD>
<TITLE> пример использования поля для выбора</TITLE>
<SCRIPT Language="VBScript">
 Sub Calculate_OnClick
znak = Document.Calc.Sign.Options _(Document.Calc.Sign.SelectedIndex).Text
elem1 = Document.Calc.element_1.Value
elem2 = Document.Calc.element_2.Value
elem1 = CDbl(elem1)
elem2 = CDbl(elem2)
Select Case znak
 Case "+"
res = elem1+elem2
Case "-"
res = elem1-elem2
Case "*"
res = elem1*elem2
Case "/"
res = elem1/elem2
Case "^"
res = elem1^elem2
Case "ln"
res = LOG(elem1)
End Select
Document.Calc.Result.Value=res
End Sub
</SCRIPT>
 </HEAD>
<BODY BGCOLOR="#FFFFFF">
 <P ALIGN="center">
<FONT COLOR="Red" SIZE="5">
Калькулятор </FONT>
</P>
<FORM NAME="Calc">
 <P ALIGN="center">
 <INPUT TYPE="text" NAME="element_1">
```

```
<BR>
<SELECT NAME="Sign" size="1">
<OPTION>+
<OPTION>-
<OPTION>*
<OPTION>/
<OPTION>^
<OPTION>ln
</SELECT> <BR>
<INPUT TYPE="text" SIZE="11"
NAME="element_2">
<BR>
=
<BR>
<INPUT TYPE="text" SIZE=”20” NAME="result">
</P>
<P ALIGN="center">
<INPUT   TYPE="button"  NAME ="Calculate"VALUE="Вычислить">
</P>
<P ALIGN="center">
<INPUT  TYPE="reset"   NAME="Сброс"
</P>
</FORM>
</BODY>
</HTML>
```

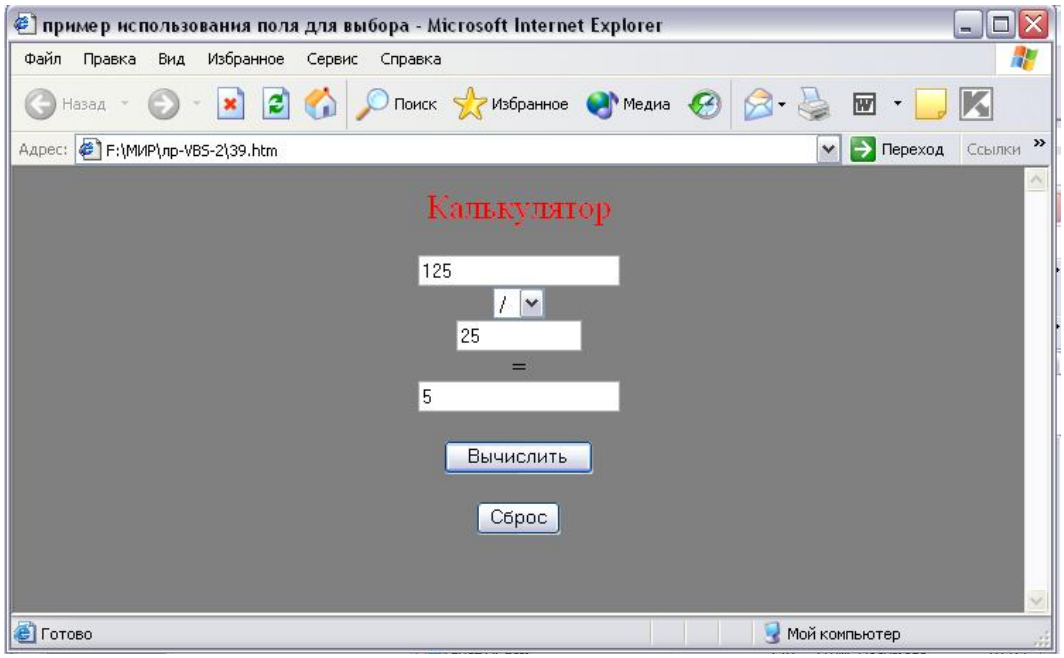

Рис. 17. Список выбора. Калькулятор

Элемент управления Hidden

Элемент управления *Hidden* позволяет скрыть содержимое объекта формы. Обычно это данные, которые должны быть защищены до их отправки на сервер. Таким образом, элемент управления *Hidden* предоставляет возможность не отображать на Web-странице некоторые данные, предназначенные, тем не менее, для отсылки серверу.

Два свойства элемента *Hidden* представлены ниже.

Name. Данное свойство определяет имя элемента управления, который должен быть скрыт. Атрибут *NAME* тега <INPUT> задает значение этого предназначенного только для чтения свойства.

Value. Это свойство, задаваемое одноименным атрибутом тега <INPUT>, содержит значение элемента *Hidden* и предназначено как для чтения, так и для программного изменения.

Динамические Web-страницы

Самым существенным достижением клиентских сценариев является возможность создания динамических Web-страниц, для формирования которых не требуется обращение к серверу. Таким образом, пользователь имеет дело со страницей, генерирующей другие, возможно, такие же интерактивные, страницы.

Достигнутая таким образом разгрузка сервера не только экономит время, снижает трафик, но и позволяет серверу более эффективно использовать свои ресурсы.

Помимо этого удается решить проблему, возникающую, например, при работе с датой и временем. При передаче данных с сервера информация могла устареть. Да и разницу во времени между различными часовыми поясами нельзя сбрасывать со счетов. Если же страница генерируется прямо в браузере, эта проблема решается сама собой.

Но и это еще не полный перечень достоинств динамических Web-страниц. Ведь VBScript позволяет не только генерировать новые, но и дополнять уже существующие Web-страницы, состоящие из HTML-кода.

Динамические страницы существуют в памяти компьютера, ведь мы знаем, что VBScripl не имеет возможностей сохранения информации на диск за исключением файлов *Cookie*.

Для работы с динамическими страницами используются методы объекта *Document Open, Write, Writeln, Close*.

Пример, приведенный ниже (листинг 24, рис.18), иллюстрирует работу по созданию динамической страницы, которая отображается затем в правом фрейме окна. Пользователь заполняет несложную форму в левом фрейме (листинг 25), а правый фрейм (листинг 26) в это время представляет собой пустую страницу. После нажатия кнопки *Выполнить* сгенерированная страница отобразится в правом фрейме окна.

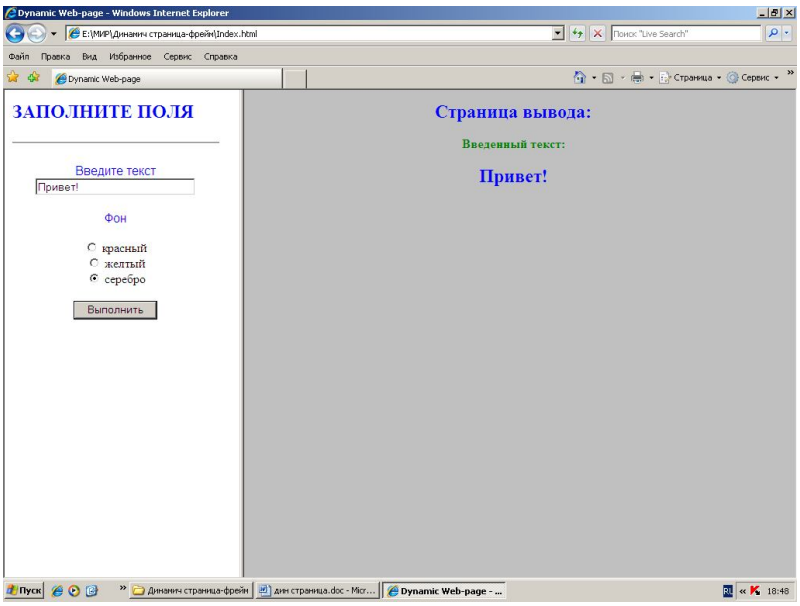

Рис. 18. Динамическая страница после выполнения кода (листинг 24,25,26)

Листинг 24. HTML-код документа

```
<HTML>
<HEAD>
<TITLE> Dynamic Web-page </TITLE> </HEAD>
<FRAMESET COLS=30%,70%>
<FRAME NAME="FrameInput"SRC="inputfr.htm">
```

```
<FRAME NAME="FrameOutput"SRC="outputfr.htm">
</FRAMESET>
<BODY BGCOLOR="#FFFFFF">
</body>
</html>
```

Листинг 25. HTML-код документа inputfr.htm, расположенного в левом фрейме

```
<HTML>
<HEAD>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1251">
  <META NAME="Author" CONTENT="Anonimus">
  <META NAME="GENERATOR" CONTENT="Lesteh">
  <TITLE></TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="VBScript">
Sub GenerateB_OnClick
Top.FrameOutput.Document.Open
If Top.FrameInput.Document.TestForm.Elements(1).Checked= True Then
Top.FrameOutput.Document.BgColor="Red"
End If
If Top.FrameInput.Document.TestForm.Elements(2).Checked Then
Top.FrameOutput.Document.BgColor="Yellow"
End If
If Top.FrameInput.Document.TestForm.Elements(3).Checked Then
Top.FrameOutput.Document.BgColor="Silver"
End If
NL=chr(13)+chr(10)
Top.FrameOutput.Document.Write "<HTML>" & NL
Top.FrameOutput.Document.Write "" & NL
Top.FrameOutput.Document.Write "<HEAD>" & NL
Top.FrameOutput.Document.Write "<TITLE>" & NL
Top.FrameOutput.Document.Write "Страница вывода:" & NL
Top.FrameOutput.Document.Write "</TITLE>" & NL
Top.FrameOutput.Document.Write "</HEAD>" & NL
Top.FrameOutput.Document.Write "" & NL
Top.FrameOutput.Document.Write "<BODY Bgcolor='White'>" & NL
Top.FrameOutput.Document.Write "" & NL
Top.FrameOutput.Document.Write "<p align='center'>" & NL
Top.FrameOutput.Document.Write "<font color='Blue' size='5'>" & NL
Top.FrameOutput.Document.Write "<strong> Страница вывода:" & NL
Top.FrameOutput.Document.Write "</strong></font></p>" & NL
Top.FrameOutput.Document.Write "<p align='center'><Font color='Green' size='3'>" &
NL
Top.FrameOutput.Document.Write "<strong>Введенный текст: " & NL
Top.FrameOutput.Document.Write "</strong></font></p>" & NL
Top.FrameOutput.Document.Write "<p align='center'>" & NL
Top.FrameOutput.Document.Write "<font color='Blue' size='5'><strong>" & NL
```

```
Top.FrameOutput.Document.Write Self.Document.TestForm.TestText.Value & NL
Top.FrameOutput.Document.Write "</strong>" & NL
Top.FrameOutput.Document.Write "</BODY>" & NL
Top.FrameOutput.Document.Write "</HTML>" & NL
Top.FrameOutput.Document.Close
End Sub
</SCRIPT>
<h3><font color="#0000FF" size="5">
ЗАПОЛНИТЕ ПОЛЯ </Font></h3>
<hr>
<form name="TestForm">
 <p align="center">
<font color="#0000FF" size="3"face="Arial"> Введите текст </font>
<font face="Arial">
<input type="text" size="30" name="TestText">
</font>
</p>
<p align="center">
<font color="#0000FF" Size="3"face="Arial"> Фон </font>
<p align="center">
<input type="radio" name="ColorR" value="V1"> красный <br>
<input type="radio" name="ColorR" value="V2"> желтый <br>
<input type="radio" name="ColorR" value="V3"> серебро </p>
<p align="center">
<font face="Arial">
<input type="button" name="GenerateB" value="Выполнить"> </font>
</p>
</form>
</body>
</html>
```

Листинг 26. HTML-код документа Output.htm, расположенного в правом фрейме.

```
<HTML>
<HEAD>
 <TITLE> Правый фрейм:</TITLE>
</HEAD>
<BODY bgcolor="white">
Страница для заполнения
</BODY>
</HTML>
```

ПРИМЕРЫ НА ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ

Работа с датой и временем

VBScript располагает большим набором разнообразных гибких функций для работы с датой и временем. Порой назначение отдельных функций оказывается нетривиальным, но при ближайшем рассмотрении можно разглядеть в них глубокий смысл и простоту в использовании.

Рассмотрим, к примеру, вопрос о формате представления даты. Ведь в тех же Германии и США используются разные форматы дат. Функция VBScript *IsDate* обращается к установкам *Панели управления*, чтобы определить формат даты, используемый пользователем. Это дает возможность вводить данные так, как привык пользователь, независимо от страны проживания.

Формат использования функции *IsDate* имеет следующий вид:

```
Переменная = IsDate (строка)
```

Переменная принимает значение типа *Boolean*, определяющее правильность введенного формата даты. Если *Переменная* = *True*, то дата введена верно.

Эту же функцию можно использовать для проверки введенного времени. Однако для строки, содержащей время, можно использовать лишь два разделителя " :" и ".".

Исполняемые VBScript форматы Windows *Short Date* и *Long Date* имеют некоторые различия в представлении. Так, *Short Date* не включает ведущие нули перед значением месяца или числа, а значение года не содержит столетия. *Long Date*, помимо этих данных, предлагает информацию о дне недели.

Наиболее распространенной задачей при работе с датой и временем является определение текущих данных даты и времени.

Для их определения можно использовать функции, приведенные ниже.

Date − возвращает значение даты на клиенте в формате *Short Date* согласно установкам, заданным в *Панели управления*:

```
Переменная = Date ()
```

Now − возвращает значения даты и времени на клиенте в формате *Short Date* согласно установкам, заданным в *Панели управления*:

```
Переменная = Now()
```

Time − возвращает значение времени на клиенте в двенадцатичасовом формате согласно установкам, заданным в *Панели управления*:

```
Переменная = Time ()
```

Для выделения составляющей из значения времени или даты применяются функции *Hour, Minute, Second, Year, Month, Day*. Все они имеют сходный формат вызова. Например,

```
Переменная  = Month(Дата)
```

или

```
Переменная = Minute(Дата)
```

Для преобразования строки в подтип *Date* с целью получения даты следует использовать функцию *DateValue*:

```
Date_Переменная = DateValue(строка)
```

Значение даты формируется в формате *Short Date*. Для получения времени следует использовать функцию *TimeValue*. Ее синтаксис имеет следующий вид:

```
Date_Переменная = TimeValue (строка)
```

Функция *Weekday* позволяет определять день недели по дате. Эта функция имеет следующий синтаксис:

```
Integer_Переменная = Weekday(дата)
```

Надо отметить, что нумерация дней начинается с воскресенья и заканчивается субботой.

В приведенном примере (листинг 27, рис. 19) пользователь имеет возможность познакомиться с функциями даты и времени на практике.

Листинг 27. *Работа с функциями даты и времени*

```
<html>
<head>
<title>Time & Date</title>
<script language="VBScript">
Sub TimeB_OnClick
NL = chr(13) + chr(10)
If IsDate (Document.TestForm.TimeT.Value) Then
t = TimeValue(Document.TestForm.TimeT.Value)
h = Hour(t)
m = Minute(t)
s = Second(t)
Strt = "Time: " + CStr(t) + NL + "Hour: " + CStr(h) + NL + -
"Minute: " + CStr(m) + NL + "Second: " + CStr(s)
Alert Strt
Else
Alert "Неверное время!"
End If
End Sub
Sub DateB_OnClick
        Dim days(7)
days(1)="Sunday"
days(2)="Monday"
days(3)="Tuesday"
days(4)="Wednesday"
days(5)="Thursday"
days(6)="Friday"
days(7)="Saturday"
Dim months(12)
months(1)="January"
months(2)="February"
months(3)="March"
months(4)="April"
months(5)="May"
months(6)="June"
months(7)="July"
months(8)="August"
months(9)="September"
months(10)="October"
months(11)="November"
months(12)="December"
If IsDate(Document.TestForm.DateT.Value) Then
d = DateValue(Document.TestForm.DateT.Value)
y = year(d)
m = month(d)
x = day(d)
Strd = CStr(x) + " " + months(m) + " " + CStr(y) + chr(13) + chr(10)
Strd = Strd + days(Weekday(d))
Alert Strd
Else
Alert "Неверная дата!"
```

```
End If
        End Sub
</script>
</head>
<body bgcolor="white">
        <form name="testForm">
<P Align="center">
<font color="Red" size="4"><strong><em>
Введите время</em></strong></font>
<input type="text" name="TimeT">
<P Align="center">
</P>
<p align="center">
<input type="button" name="TimeB"
value="Время">
</P>
<p Align="center">
<font color="blue" size="4"><strong><em>
Введите дату</em></strong></font>
        <input type="text" name="DateT">
<br></p>
<p align="center">
<input type="button" name="dateB"
value="Дата">
</p>
</form>
</body>
</html>
```

При вводе времени и нажатии на кнопку *Time Value* отобразятся значения функций по работе со временем. Ввод даты позволяет также определить день недели по заданному значению и улучшить формат выдачи даты посредством работы с массивом (рис. 19).

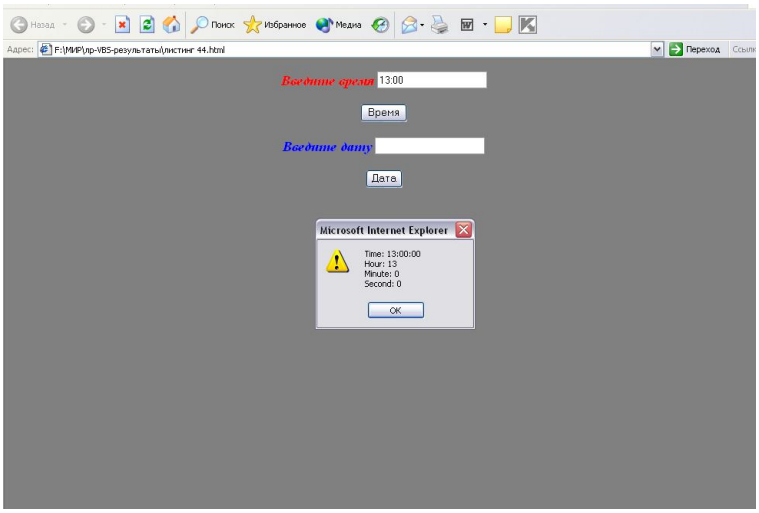

Рис.19. Страница к листингу 27

Пример преобразования текста в целочисленный вид

Поставим задачу написать программу, которая бы случайным образом загадывала число от 1 до 100 и предлагала вам его угадать. При неправильном предположении программа должна выводить сообщение о том, больше загаданное число или меньше. Ведется подсчет попыток. В случае победы выводится поздравление (рис. 20).

Листинг 28. Код программы «Угадай»

```
<html>
<head>
<title>Угадай число</title>
<script language="VBscript">
dim a                    ' определяем переменные
dim p
dim v
sub begin_onclick
 randomize           ' включение генератора случайных чисел
 a=int(rnd(1)*100+1)        ' присваивание переменной (a) случайного  значения
 p = 1                      ' обнуление счетчика
 alert "число загадано" ' вывод сообщения в отдельном окне
end sub                     ' конец процедуры
sub but_onclick
 v = document.ugadai.chislo.value
' переменной (v) присваивается значение, содержащееся в поле ввода формы
' путь: документ (просто эта html страница).форма с именем ugadai.
' объект- поле для ввода с именем chislo , свойство value
 v=cint(v)    ' поскольку поле для ввода содержит текстовую
              ' информацию, то переменную (v) надо преобра-
              ' зовать в целочисленный тип (функция cint)
 if a > v then
   alert "загаданное число больше, попробуйте еще"
   p = p + 1
 end if
 if a < v then
   alert "загаданное число меньше, попробуйте еще"
   p = p + 1
 end if
    if a = v then
   document.write"<center>Победа за "&p&" ходов.</center>"
 end if
              ' оператор document.write выводит содержащуюся
              ' в кавычках информацию в новое окно браузера
              ' как html код.
end sub
</script>
</head>
```

```
<body bgcolor="White" text="black">
<FORM NAME="ugadai">
<INPUT TYPE="button" NAME="begin" VALUE="Загадать число"><BR><BR>
Ваш вариант:<BR>
<INPUT TYPE="text" NAME="chislo">
<BR>
<INPUT TYPE="button" NAME="but" VALUE="Введите">
</FORM>
</body>
</html>
```

Рис.20. Страница к листингу 28. Вид после нажатия кнопки «Загадать число»

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Глушаков, С.В. Программирование в среде Windows. Учебный курс [Текст] / С.В. Глушаков, И.В. Мельников, А.С. Суряный. Харьков: Фолис; М: АСТ, 2000. 485 с.

Интернет-источники

2. http://azbukavb.narod.ru/vbsprog/vbsprog1.html

СОДЕРЖАНИЕ