Ташкентский государственный институт
востоковедения

Самостоятельная работа

На тему: JavaScript и VBScript

Выполнила *студентка 2 курса*
китайско-английской группы
Мухсимова Дильноза
Приняла *Салихова*
Венера

Содержание

Введение

Объектные модели языков сценариев тесно связаны с тэгами HTML. При загрузке страницы HTML в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Говорят, что браузер отражает HTML-страницу в свойствах объектов, и иногда этот процесс называют *отражением* (reflection). Созданные объекты существуют в виде иерархической структуры, отражающей структуру самой HTML-страницы. На верхнем уровне расположен объект window, представляющий собой активное окно браузера. Далее вниз по иерархической лестнице следуют объекты frame, document, location и history, представляющие соответственно фрейм, непосредственно сам документ, адрес загружаемого документа и список ранее загружавшихся документов, и т.д. Значения свойств объектов отражают значения соответствующих параметров тэгов страницы или установленных системных параметров.

Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархия объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. Поэтому невозможно обратится из сценария, расположенного ранее какой-либо формы на странице, к элементам этой формы. Всегда следует помнить о том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы.

Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображенного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

Общий обзор языка

JavaScript

Язык программирования JavaScript разработан фирмой Netscape для создания интерактивных HTML-документов. Это объектно-ориентированный язык разработки встраиваемых приложений, выполняющих как на стороне клиента, так и на стороне сервера. Синтаксис языка очень похож на синтаксис языка Java – поэтому его часто называют Java-подобным. Клиентские приложения выполняются браузером просмотра Web-документов на машине пользователя, серверные приложения выполняются на сервере.
При разработке обоих типов приложений используется общий компонент языка, называемый ядром и включающий определения стандартных объектов и конструкций (переменные, функции, основные объекты и средство LiveConnect взаимодействия с Java-апплетами), и соответствующие компоненты дополнений языка, содержащие специфические для каждого типа приложений определения объектов.

Клиентские приложения непосредственно встраиваются в HTML-страницы и интерпретируются браузером по мере отображения частей документа в его окне. Серверные приложения для увеличения производительности предварительно компилируются в промежуточный байт-код.

Основные области использования языка JavaScript при создании интерактивных HTML-страниц:

- Динамическое создание документа с помощью сценария
- Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер
- Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа
- Взаимодействие с пользователем при решении “локальных” задач, решаемых приложением JavaScript, встроенном в HTML-страницу

VBScript

Язык создания сценариев VBScript разработан фирмой Microsoft и является подмножеством достаточно распространенного в среде программистов языка Visual Basic разработки прикладных Windows-приложений. Как и его родитель, язык VBScript достаточно прост и легок в изучении.

Преимуществом его применения для создания сценариев является возможность использования, с небольшими корректировками, ранее написанных процедур на языках Visual Basic и Visual Basic for Application.
Функциональные возможности сценариев, написанных на VBScript, ничем не отличаются от возможностей сценариев JavaScript: динамическое создание документа или его частей, перехват и обработка событий и т.д.

VBScript используется для написания сценариев клиента (в этом случае браузер должен иметь встроенный интерпретатор этого языка), а также для написания сценариев на сервере (в этом случае сервер должен поддерживать язык VBScript). Для создания сценариев клиента используется набор объектов, аналогичный набору объектов JavaScript. Объекты клиента и сервера отличаются друг от друга, но существует общая часть (ядро) объектов, используемых при разработке как сценариев клиента, так и сценариев сервера.

Типы данных

JavaScript

Как и любой другой язык программирования, JavaScript использует переменные для хранения данных определенного типа. Реализация JavaScript является примером языка свободного использования типов. В нем не обязательно задавать тип переменной. Ее тип зависит от типа хранимых в ней данных, причем при изменении типа данных меняется и тип переменной.

JavaScript поддерживает четыре простых типа данных:

- Целый
- Вещественный
- Строковый
- Булевый, или логический

Для присваивания переменным значений основных типов применяются *литералы* – буквенные значения данных соответствующих типов.

Целые литералы являются последовательностью цифр и представляют обычные целые числа со знаком или без знака:

```
 123 // целое положительное число
-123 // целое отрицательное число
+123 // целое положительное число
```

Для задания вещественных литералов используется синтаксис чисел с десятичной точкой, отделяющей дробную часть числа от целой, или запись вещественных чисел в научной нотации с указанием после символа “e” или “E” порядка числа. Пример правильных вещественных чисел:

```
1.25    0.125e01        12.5E-10.0125E+2
```

Строковый литерал – последовательность алфавитно-цифровых символов, заключенная в одинарные (‘) или двойные кавычки (“), например: “Ира”, ‘ИРА’. При задании строковых переменных нельзя смешивать одинарные и двойные кавычки. Недопустимо задавать строку, например, в виде “Ира’. Двойные кавычки – это один самостоятельный символ, а не последовательность двух символов одинарных кавычек. Если в строке нужно использовать символ кавычек, то строковый литерал необходимо заключать в кавычки противоположного вида:

```
“It’s a string” // Значение строки равно It’s a string
```

Булевы литералы имеют два значения: true и false, и используются для обработки ситуаций да/нет в операторах сравнения.

VBScript

В VBScript определен только один тип данных – Variant. Это универсальный тип, в котором можно хранить информацию, предоставленную другими типами данных, применяемыми в программировании, начиная от простейшего целого и заканчивая объектами.

В своем простейшем использовании тип Variant содержит либо числовые данные, либо символьные строки – типы данных, наиболее часто встречаемые при

написании сценария. Реально содержащиеся в вариантом типе данные могут быть одного из типов, называемых *подтипами* типа Variant, представленных в табл.1.

Таблица 1. Подтипы данных, хранящихся в типе Variant

Подтип	**Описание**	**Функция преобразования**
Empty	Переменная не инициализирована	
Null	Переменная не содержит никаких допустимых данных	
Error	Содержит номер ошибки	
Boolean	Содержит значения либо True, либо False	CBool
Byte	Содержит целые числа в диапазоне от 0 до 255	CByte
Integer	Содержит целые числа в диапазоне от -32 768 до 32 767	CInt
Currency	Значения в диапазоне от -922 337 203 685 477.5808 до 922 337 203 685 477.5807	CCur
Long	Содержит целые числа в диапазоне от -2 147 483 648 до 2 147 483 647	CLng
Single	Содержит вещественные числа с плавающей точкой одинарной точности в диапазоне от -3.402823E38 до -1.401298E-45 для отрицательных значений и от 1.401298E-45 до 3.402823E38 для положительных значений	CSng
Double	Содержит вещественные числа с плавающей точкой удвоенной точности в диапазоне от -1.79769313486232E308 до -4.94065645841247E-324 для отрицательных значений и от 4.94065645841247E-324 до 1.79769313486232E308 для положительных значений	CDbl
Date(Time)	Содержит число, которое представляет дату в диапазоне от 1 января 100 года до 31 декабря 9999 года	CDate
String	Содержит строку переменной длины (до 2 миллионов символов)	CStr
Object	Содержит ссылку на объект	

Первые три подтипа, собственно говоря, не являются подтипами, а представляют значения, которые может принимать вариантный тип.

Значение Empty имеет переменная, которая была объявлена в операторе Dim (см. ниже), но ей еще не присваивали никакого значения. Это значение считается равным 0 в математических операциях и равным пустой строке (“”) в операциях со строковыми значениями.

Значение Null означает, что переменная не содержит данных. Его не следует путать со значением Empty. Вариантная переменная может получить значение Null в результате выполнения некоторых операций над ней. Это значение можно присвоить переменной, тогда как значение Empty – нельзя.

Значение Error – это специальное значение, которое используется для указания возникновения ошибки в процедуре.

Каждый подтип данных задается с помощью литералов. Числовые литералы представляют собой целые числа, действительные числа с плавающей или фиксированной точкой. Примеры числовых литералов приведены ниже:

```
 23              ‘Целое число
-23.78  ‘Действительное число с фиксированной точкой
-237.8E-1       ‘Действительное число с плавающей точкой
```

Строковые литералы задаются в виде последовательности символов, заключенных в двойные кавычки (“):

“Это строковый литерал”.

Литералы даты и времени заключаются между символами числовых знаков (#). VBScript поддерживает большое число форматов даты и времени. Следующие примеры показывают правильные литералы даты и времени, соответствующие дате 10 июня 1999 года:

#10-6-99 22:20#
#10/6/99#
#10/6/99 10:20pm#

Внутренне литералы даты и времени представляются в виде действительных чисел удвоенной точности. Целая часть представляет количество дней, прошедших от даты 30 декабря 1899 года, а дробная часть – время суток.

Булевы литералы True и False являются константами целого типа, принимающими соответственно значения 1 и 0. Любое числовое значение, не равное нулю, преобразуется функцией CBool в True, а нулевое значение (целое или действительное) – в False.

Вариантный тип данных при использовании в выражениях в качестве операндов разнообразных операторов языка обрабатывается в зависимости от подтипа содержащихся в нем данных. Например, при использовании переменных этого типа данных в операторе сложения (+) результат зависит от того, какие подтипы данных в них содержатся. Если хотя бы один из операндов содержит число, то результат будет сумма значений двух переменных (содержимое второго операнда преобразуется к числовому подтипу), если оба операнда содержат строковые данные, тот результатом будет конкатенация строк.

Вариативный тип данных предоставляет программисту более эффективный способ обработки и хранения данных, не заботясь о типе хранимых данных. Если, например, при вычислениях первоначально в переменной вариантного типа хранилось значение типа Byte (число в диапазоне от 0 до 255), и в результате выполнения некоторых действий это значение стало отрицательным, то просто изменится представление этого числа в переменной (оно станет Integer) и не возникает никакой ошибки. Правда, за это удобство приходится платить используемой памятью: для вариантного типа данных вне зависимости от хранимого подтипа нужно 16 байт памяти.

Иногда в некоторых вычислениях необходимо явно преобразовать содержащийся в переменной подтип в другой. Для этого в VBScript имеется ряд функций преобразования в соответствующие типы. В табл. 1 последний столбец содержит имена функций преобразования в соответствующий подтип. Эти функции в качестве параметра принимают литералы, переменные и выражения.

Переменные

JavaScript

Каждая переменная имеет имя, которое должно начинаться с буквы латинского алфавита, либо символа подчеркивания “_”, за которым следует любая комбинация алфавитно-цифровых символов или символов подчеркивания. Следующие имена являются допустимыми именами переменных

```
Temp1
MyFunction
_my_Method
```

Язык JavaScript чувствителен к регистру. Это означает, что строчные и прописные буквы алфавита считаются разными символами.

Определить переменную можно двумя способами:

- Оператором var
- Оператором присваивания (=)

Оператор var используется как для задания, так и для инициализации переменной и имеет синтаксис:

```
var имя_переменной [= начальное_значение];
```

Необязательный оператор присваивания задает данные, которые содержит переменная. Их тип определяет и тип переменной. Например, следующий оператор

```
var weekDay = “Пятница”;
```

задает переменную weekDay, присваивает ей строковое значение “Пятница”, и тем самым определяет ее тип как строковый.

Если при определении переменной ей не присвоено никакого значения, то ее тип не определен. Ее тип будет определен только после того, как ей будет присвоено некоторое значение оператором присваивания =.

VBScript

Переменные используются для хранения данных приложения. Прежде чем переменную можно будет использовать, ее необходимо объявить. Это можно осуществить явным способом с помощью оператора Dim, или неявным – просто использовать имя переменной в операторе присваивания. Синтаксис оператора явного объявления переменной следующий:

```
Dim имя_переменной
```

Параметр имя_переменной – имя объявляемой переменной. Оно должно начинаться с буквы, не содержать пробелов, точку (.), восклицательный знак (!), а также символов (@), (&), ($), (#) и не превышать длину в 255 символов.

Язык VBScript не чувствителен к регистру. Это означает, что в нем не различаются строчные и прописные буквы. Поэтому, например, и m, и M будут ссылаться на

одну и ту же переменную, если используются в качестве идентификатора переменной.

Иногда в программе необходимо задавать переменные, значения которых нельзя изменять. Такие переменные называются *именованными константами*. В VBScript для задания констант существует оператор Const, имеющий следующий синтаксис:

```
Const conName = “Дмитрий”    ‘Строковая константа
Const conPi = 3.1416         ‘Числовая константа
Const conBirthDay = #1-8-53# ‘Константа даты
```

Операторы

JavaScript

Оператор *присваивания* рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной, заданное в левой части оператора.

Арифметические выражения создаются *арифметическими* операторами (табл. 2).

Таблица 2. Арифметические операторы

Оператор	Название
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения переменной на единицу
--	Уменьшение значения переменной на единицу

Кроме простого оператора присваивания (=) существуют сокращенные формы операторов присваивания, совмещенных с арифметическими операторами, в которых производятся арифметические действия над левыми и правыми операндами и результат присваивается переменной, заданной левым операндом. Все они перечислены в табл. 3.

Таблица 3. Сокращенные операторы присваивания

Оператор	Значение
X * = Y	X = X * Y
X / = Y	X = X / Y
X + = Y	X = X + Y
X - = Y	X = X – Y
X % = Y	X = X % Y

Для создания *логических* выражений используются операторы сравнения и логические операторы, применяемые к переменным любого типа.

Операторы *сравнения* аналогичны таковым в других языках программирования. Их список представлен в табл. 4.

Таблица 4. Операторы сравнения

Оператор	Название
==	Равно
!=	Не равно
>=	Больше или равно
<=	Меньше или равно
>	Строго больше
<	Строго меньше

При использовании этих операторов в выражении оно вычисляется равным true, если соответствующее сравнение верно, в противном случае значение выражения равно false.

Логические операторы представлены в табл. 5. В примерах предполагается, что переменная var1 = ‘Кит’, var2 = ‘Кот’, var3 = false.

Таблица 5. Логические операторы

Оператор	Синтаксис	Описание	Пример
&& (логическое И)	выраж1 && выраж2	Возвращает выраж1, если оно преобразуется или равно false, иначе выраж2	var1 && var2 (равно ‘Кот’) var2 && var3 (равно false)
|| (логическое ИЛИ)	выраж1 || выраж2	Возвращает выраж1, если оно преобразуется или равно true, иначе выраж2	var1 || var2 (равно ‘Кит’) var3 || var1 (равно ‘Кит’) var3 || false (равно false)
! (логическое НЕ)	!выраж	Если выраж равно true, возвращает false; если выраж равно false, возвращает true	!var1 (равно false) !var3 (равно true)

Логические операторы и операторы сравнения используются в операторах цикла и условия для проверки завершения цикла или выполнения определенной группы операторов.

Строковые операторы используются для создания строковых выражений. В JavaScript, собственно говоря, существует только один строковый оператор – оператор конкатенации (соединения) строк (+), если не считать сокращенной формы оператора присваивания со сложением (+=). Этот оператор присоединяет к строковому значению первого операнда строковое значение второго, получая результат, равный соединению строк:

```
string = “Моя”+”строка”; // Значение переменной string равно “Моястрока”
```

Условный оператор является единственным оператором, использующим три операнда. Его значением является один из двух операндов, определяемый из условия истинности третьего. Его синтаксис таков:

```
(условие) ? знач1 : знач2;
```

Если операнд условие имеет значение true, то результатом вычисления условного оператора будет знач1, в противном случае – знач2. Например, оператор

```
range = (mark <= 2) ? “Пересдача” : “Зачтено”;
```

присваивает переменной range значение “Пересдача”, если переменная mark меньше либо равно 2, в противном случае ей присваивается значение “Зачтено”.

Кроме перечисленных выше операторов в JavaScript существует большая группа операторов для *поразрядных действий* с данными. В них содержимое каждого оператора рассматривается как последовательность битов, а не как данные строкового, числового или булевого типов.

VBScript

При вычислении выражений необходимо производить разнообразные действия с переменными и литералами. Для этих целей в VBScript предусмотрен ряд встроенных операторов, выполняющих арифметические операции, операции сравнения, конкатенацию (соединение) строк и логические операции над данными, хранящимися в переменных, или представленными литералами.

В VBScript каждый оператор размещается на отдельной строке и не завершается никаким разделителем. Однако, если возникает необходимость задания нескольких операторов в одной строке, то они разделяются двоеточием (:).

Если оператор достаточно длинный, или из соображений удобства чтения исходного текста необходимо расположить его в нескольких строках, то следует использовать символы продолжения – пробел со знаком подчеркивания (_).

Комментарии в языке VBScript вводятся в текст программы одинарной кавычкой (‘). Любой текст, расположенный в строке за одинарной кавычкой, трактуется интерпретатором как комментарий, и, естественно, не обрабатывается им.

Большую группу представляют *арифметические* операторы, выполняющие основные арифметические действия над числовыми данными: возведение в степень (^), умножение (*), деление (/), целочисленное деление (\), сложение (+) и вычитание (-). Они подчиняются принятым в математике правилам старшинства операций: сначала выполняется возведение в степень, затем умножение или деление, далее сложение или вычитание. Скобки изменяют последовательность вычисления операций.

Для сравнения данных используются операторы *сравнения*: равенство (=), неравенство (<>), меньше (<), больше (>), меньше или равно (<=), больше или равно (>=). Объекты сравниваются с помощью специального оператора Is.

В VBScript нет специального знака для операции присваивания. Один и тот же знака равенства используется как для операции присваивания значения переменной (см. выше), так и для операции сравнения на равенство. Смысл операции, представляемой этим символом, зависит от контекста, в котором она применена.

Существует ряд операторов, выполняющих действия над *логическими* (булевыми) данными: отрицание (Not), конъюнкция (And), дизъюнкция (Or), исключающее ИЛИ (Xor), эквивалентность (Eqv) и импликация (Imp).

Оператор (&) производит конкатенацию (соединение) двух строк. При его выполнении данные, содержащиеся в операндах, преобразуются при необходимости к строковому подтипу, и осуществляется сцепление двух строк.

Операторы управления

JavaScript

Весь набор управления языка можно разбить на три группы:

- Операторы выбора, или условные
- Операторы цикла
- Операторы манипулирования с объектами

Операторы выбора

К этой группе операторов относятся операторы, которые выполняют определенные блоки операторов в зависимости от истинности некоторого булевского выражения. Этот оператор условия if…else и переключатель switch.

Оператор условия if применяется, если необходимо вычислить некоторый блок операторов в зависимости от истинности заданного условия, и имеет следующий синтаксис:

```
if (условие) {
        операторы1
}
[else {
        операторы2
}]
```

Первая группа операторов операторы1 выполняется при условии истинности выражения условие. Необязательный блок else задает группу операторов операторы2, которая будет выполнена в случае ложности условия, заданного в блоке if.

Внутри группы выполняемых операторов могут использоваться любые операторы JavaScript, в том числе и операторы условия. Таким образом, можно создавать группу вложенных операторов условия и реализовывать сложные алгоритмы проверки.

В операторе switch вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответствующего блока case. Синтаксис этого оператора следующий:

```
switch (выражение) {
        case значение1 :
                [операторы1]
                break;
        case значение2 :
                [операторы2]
                break;
        …
        default :
                [операторы]
}
```

Если значение выражения в блоке switch равно значение1, то выполняется группа операторов операторы1, если равно значение2, то выполняется группа операторов операторы2 и т.д. Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch.

Необязательный оператор break, задаваемый в каждом из блоков case, выполняет безусловный выход из оператора switch. Если он не задан, то продолжается выполнение операторов в следующих блоках case до первого оператора break или до конца тела оператора switch.

Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполнится некоторое заданное условие. В языке существует два оператора цикла: for и while. Они отличаются механизмом организации цикла.

Оператор цикла for позволяет организовать выполнение блока операторов заданное число раз. Он определяет переменную, называемую *переменной цикла*, которая изменяет свое значение во время выполнения цикла. Условие завершения цикла зависит от значения этой переменной. Оператор имеет следующий синтаксис:

```
for ([инициал_выражение];[условие];[изменяющее_выражение]) {
        [операторы]
}
```

Параметром инициал_выражение задается и инициализируется переменная цикла. Это выражение вычисляется один раз в начале выполнения цикла. После этого проверяется истинность выражения условие. Если оно истинно, то выполняется блок операторов тела цикла, ограниченного фигурными скобками; вычисляется изменяющее_выражение, содержащее переменную цикла, и снова проверяется истинность выражения условие. Если оно истинно, то повторяется цикл вычислений, если нет, то оператор цикла завершает свое выполнение.

Цикл while выполняется пока истинно выражение, задающее условие выполнения цикла. Его синтаксис следующий:

```
while (условие) {
        [операторы]
}
```

Сначала проверяется истинность условия, заданного в заголовке цикла, а затем выполняются (или не выполняются) операторы тела цикла. Проверка истинности условия осуществляется на каждом шаге цикла. Использование этого цикла предполагает, что условное выражение окончания цикла меняется в зависимости от вычисленных значений переменных и выражений в теле цикла.

Иногда необходимо завершить цикл не по условию, задаваемому в заголовке цикла, а в результате вычисления некоторого условия в теле цикла. Для этой цели в JavaScript существуют операторы break и continue.

Оператор break завершает выполнение цикла и передает управление оператору, непосредственно следующим за оператором цикла. Оператор continue прекращает выполнение текущей итерации и начинает выполнение следующей, т.е. в цикле while он передает управление на проверку выражения условие цикла, а в цикле for – на вычисление выражения изменяющее_выражение.

Манипулирование объектами
Четыре оператора JavaScript предназначены для работы с объектами. Это оператор new, создающий новый объект, операторы for…in и with и ключевое слово this.

Оператор for…in позволяет организовать цикл по свойствам объекта JavaScript. Синтаксис его следующий:

```
for (переменная_цикла in объект) {
        [операторы]
}
```

Этот цикл производит перебор свойств объекта. В переменной цикла на каждой итерации сохраняется значение свойства объекта. Количество итераций равно количеству свойств, существующих у заданного в заголовке цикла объекта.

Оператор with задает объект по умолчанию для блока операторов, определенных в его теле. Это означает, что все встречаемые в операторах этого блока свойства и методы, являются свойствами и методами указанного объекта. Применение данного оператора избавляет от необходимости указывать иерархию принадлежности объекта и сокращает исходный текст программы.

VBScript
Операторы сценария выполняются последовательно в том порядке, как они записаны. Изменить порядок выполнения операторов в VBScript можно операторами условия и цикла.

Операторы условия
Операторы принятия решения (условные операторы) выполняют определенные блоки операторов в зависимости от результатов проверки некоторого выражения или выражений. VBScript поддерживает следующие конструкции операторов принятия решения:

- if…then
- if…then…else
- select case

Конструкция if…then применяется, когда необходимо выполнить группу операторов или один оператор в зависимости от значения выражения, задаваемого в качестве параметра условия конструкции.

Ее первая форма

```
if условие then оператор
```

позволяет вычислить указанный оператор, если истинно заданное условие.

Вторая форма этой конструкции позволяет вычислить группу операторов, заданных в нескольких строках кода, и имеет следующий синтаксис:

```
if условие then
        операторы
end if
```

Наиболее общий синтаксис конструкции if…then…else следующий:

```
if условие1 then
        [группа-операторов-1]
[elseIf условие2 then
        [группа-операторов-2]…
[else
        [группа-операторов-n]]
end if
```

Сначала проверяется условие1. Если оно ложно, то проверяется условие2. Если и оно ложно, то проверяется следующее условие из группы elseIf до тех пор, пока не будет найдено истинное условие, операторы которого и выполняются. После чего управление передается оператору, непосредственно следующему за оператором end if.

Если не найдено ни одно истинное условие, то выполняется группа операторов из блока else, если он присутствует в конструкции. В противном случае управление передается оператору, следующему за оператором end if.

Блоков elseIf в конструкции if…then…else может быть сколько угодно, тогда как блок else всегда один, если он задан.

Если в предыдущей конструкции принятия решения проверяется равенство одного выражения разным условиям, она становится не достаточно эффективной как с точки зрения ее выполнения, так и сточки зрения легкости восприятия текста. В этом случае следует использовать конструкцию select case:

```
select case тестируемое_выражение
        [case список_значений1
                [группа-операторов-1]]
        [case список_значений2
                [группа-операторов-2]]
                .
                .
                .
        [case else
                [группа-операторов-n]]
end select
```

Вычисляется единственное выражение тестируемое_выражение и последовательно сравнивается со значениями из списка значений блоков case. Если значение выражения совпадает со значением, заданным в списке какого-либо блока case, то выполняется группа операторов данного блока, и после этого управление передается оператору, непосредственно следующему за оператором end select.

Если не найдено ни одного соответствия значения тестируемого выражения со значениями из списков значений, то выполняется группа операторов блока case else (в случае его наличия).

Список значений блока case может состоять из одного или нескольких значений. В случае нескольких значений они разделяются запятыми.

Операторы цикла

Для повторного выполнения несколько раз группы операторов VBScript, как и любой другой язык программирования, предоставляет разнообразные типы операторов цикла:

- do…loop
- for…next
- for each…next

Конструкция do…loop применяется для выполнения группы операторов, пока некоторое выражение ложно или истинно. Она имеет несколько разновидностей, отличающихся моментом проверки условия завершения цикла (до начала выполнения группы операторов или после) и тем, истинно или ложно это условие.

Цикл do while выполняется до тех пор, пока истинно условие окончания цикла:

```
do while условие_окончания
        группа-операторов
loop
```

Перед выполнением операторов цикла проверяется, истинно ли выражение условие_окончания. Если оно истинно, то выполняется группа-операторов (в ней изменяются значения переменных, входящих в выражение условие_окончания). После этого снова проверяется условие окончания цикла и, в случае его истинности, выполняется группа операторов тела цикла. Процедура выполняется до тех пор, пока выражение условие_окончания не станет ложным.

Цикл do while не будет выполнен ни разу, если при первой проверке условие_окончания ложно.

Другая разновидность цикла do while сначала выполняет группу операторов, а потом проверяет условие окончания цикла:

```
do
        группа-операторов
loop while условие_окончания
```

Этот цикл обязательно выполнит один раз группу операторов, определенных в теле цикла.

Цикл do until аналогичен первой разновидности цикла do while, за исключением того, что он выполняется, пока значение выражения условие_окончания ложно:

```
do until условие_окончания
        группа-операторов
loop
```

Этот цикл также может не выполниться ни одного раза, если при первой же проверке условия завершения цикла, оно оказывается истинным.

Во второй разновидности цикла do until условие окончания завершения цикла проверяется после выполнения группы операторов, и, таким образом, он обязательно выполнится хотя бы один раз:

```
do
        группа-операторов
```

```
loop until условие_окончания
```

В циклах do…loop заранее не известно количество итераций повторения группы операторов, но иногда требуется выполнить точно заданное число повторений цикла. Такую возможность предоставляет цикл for…next.

В этом цикле задается переменная, называемая счетчиком цикла, которая увеличивается (или уменьшается) на заданную величину после выполнения группы операторов. Цикл завершает свои итерации, когда значение счетчика превысит (или станет меньше) заданной величины. Синтаксис такой конструкции цикла следующий:

```
for счетчик = нач_значение to кон_значение [step приращение]
        операторы
next
```

В начале выполнения этого цикла переменной счетчик присваивается значение, заданное параметром нач_значение. Выполняются операторы цикла, и значение переменной цикла увеличивается или уменьшается (в зависимости от знака) на величину приращение. Осуществляется проверка, не превысило ли (или не стало меньше) новое значение счетчика значение параметра кон_значение. Если нет, то итерации повторяются, если да, то цикл завершает свое выполнение.

Параметр приращение цикла for…next является необязательным. Если он не задан, то по умолчанию переменная цикла увеличивается на 1.

Конструкция for each…next позволяет организовать цикл по элементам массива или по объектам некоторого набора (семейства) объектов, не зная заранее число элементов в массиве или число объектов в наборе. Синтаксис этой конструкции следующий:

```
for each элемент in группа
        операторы
next
```

Параметр группа задает имя массива или имя набора объектов. Переменная элемент на каждом шаге цикла будет содержать ссылку на элемент массива или объект набора. Цикл завершает свое выполнение, как только завершится последовательный перебор всех элементов массива или объектов набора.

Процедуры

JavaScript

Процедура, или *функция*, – это именованная последовательность операторов, которая инициализируется и выполняется простой ссылкой на имя функции.

Процедура задается оператором function, имеющим следующий синтаксис:

```
function имя_функции ([параметры] {
        [операторы]
}
```

где имя_функции – любое правильное имя языка JavaScript, параметры – список передаваемых в процедуру параметров, элементы которого отделяются запятыми.

Оператор function только определяет процедуру, но не выполняет ее. Для вызова процедуры достаточно указать ее имя с заданными в скобках параметрами.

Если в процедуре параметры отсутствуют, наличие скобок без параметров в операторе вызова процедуры обязательно.

Процедура может возвращать некоторое вычисляемое в ней значение. В этом случае обычно она называется функцией, и в операторах, определяющих последовательность выполняемых ею действий, обязательно должен присутствовать оператор return, задающий возвращаемое функцией значение. Вызов функции осуществляется аналогично вызову процедуры, но ее можно использовать в выражениях JavaScript.

Обычно все определения процедур и функции задаются в разделе <HEAD> документа. Это обеспечивает интерпретацию и сохранение в памяти всех процедур при загрузке документа в браузер.

VBScript

VBScript предусматривает создание двух типов процедур:

- Процедура sub
- Процедура function (или функция)

Процедура sub выполняет последовательность действий, но не возвращает никакого, значения, ассоциированного с ее именем. Она имеет следующий синтаксис:

```
sub имя_процедуры ([список-параметров])
        операторы
end sub
```

Через необязательный параметр можно передать в процедуру внешние данные или, наоборот, получить некоторые вычисленные ею значения.

Вызов процедуры sub осуществляется оператором call, после которого указывается имя процедуры и в круглых скобках параметры. Процедуру можно вызвать и простым указанием ее имени, но в этом случае передаваемые ей параметры задаются без скобок. Следующие два способа вызова процедуры эквивалентны:

```
call MyProc(firstarg, secondarg)
MyProc firstarg, secondarg
```

Функция также выполняет определенную последовательность операторов и ей можно передать внешние данные через параметры процедуры, но, в отличие от процедуры sub, она возвращает значение, присваиваемое ее имени, и может быть использована в выражениях VBScript. Она имеет следующий синтаксис:

```
function имя_процедуры ([список-параметров])
        операторы
        имя_процедуры = значение
end function
```

В операторах процедуры обязательно должен присутствовать оператор присвоения имени процедуры некоторого значения.

VBScript предоставляет два способа передачи параметров в процедуру:

- По ссылке
- По значению

Способ передачи параметров по ссылке, применяемый по умолчанию, передает фактический адрес переменной, используемой в качестве параметра. Это позволяет изменить содержимое соответствующего адреса памяти, а тем самым, и значение переменной.

Передача параметров по значению предполагает передачу в процедуру копии переменной, а не адреса самой переменной. Поэтому любые изменения параметра внутри процедуры воздействуют на копию, а не на саму переменную, и, следовательно, значение переменной, переданной в качестве параметра, изменяться не будет. Для указания интерпретатору, что параметр передается в процедуру по значению, используется ключевое слово ByVal, задаваемое перед параметром в описании процедуры.

Объекты

JavaScript

На самом верхнем уровне иерархии находится объект window, представляющий окно браузера и являющийся “родителем” всех остальных объектов. Расположенные ниже в иерархии объекты могут иметь свои подчиненные объекты. На рис. 1 показана структура объектов клиента (браузера).

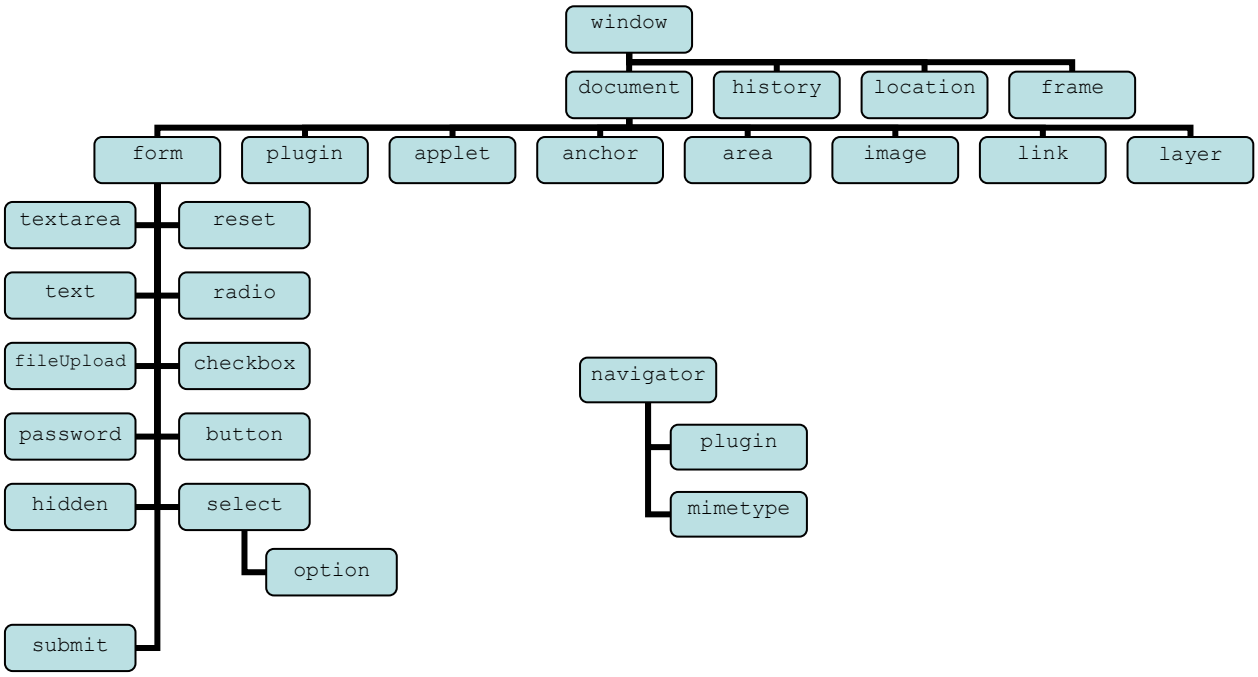

Рис. 1. Иерархия объектов JavaScript на стороне клиента

Особняком стоит объект navigator с двумя дочерними (подчиненными) объектами. Он относится к самому браузеру, и его свойства позволяют определить характеристики программы просмотра.

Каждая страница в добавление к объекту navigator обязательно имеет еще четыре объекта:

- window – объект верхнего уровня, свойства которого применяются ко всему окну, в котором отображается документ.
- document – свойства которого определяются содержимым самого документа: связи, цвет фона, формы и т.д.
- location – свойства которого связаны с URL-адресом отображаемого документа.
- history – представляет адреса ранее загружавшихся HTML-страниц.

Кроме указанных объектов страница может иметь дополнительные объекты, зависящие от ее содержимого, которые являются дочерними объектами объекта document. Если на страницы расположена форма, то все ее элементы являются дочерними объектами этой формы. Для задания точного имени объекта используется точечная нотация с полным указанием всей цепочки наследования объекта. Это возможно, так как объект верхнего уровня имеет свойство,

значением которого является объект нижнего уровня. Ссылка на объект осуществляется по имени, заданному параметром NAME тэга HTML.

```
<FORM NAME=”form1”>
Фамилия: <INPUT TYPE = “text” name = “studentName” size = 20>
Курс: <INPUT TYPE = “text” name = “course” size = 2>
</FORM>
```

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку document.form1.studentName.value, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку document.form1.course.value.

VBScript

Во главе иерархии, как и в случае с Netscape Navigator, стоит объект window, представляющий окно браузера и порождающий все остальные объекты модели. При ссылке в программе на любой объект из иерархии можно не указывать “родительский” объект window.

Модель охватывает практически все элементы HTML-страницы. На рис. 2 показана иерархическая структура объектной модели, которая отражает подчиненность элементов страницы.

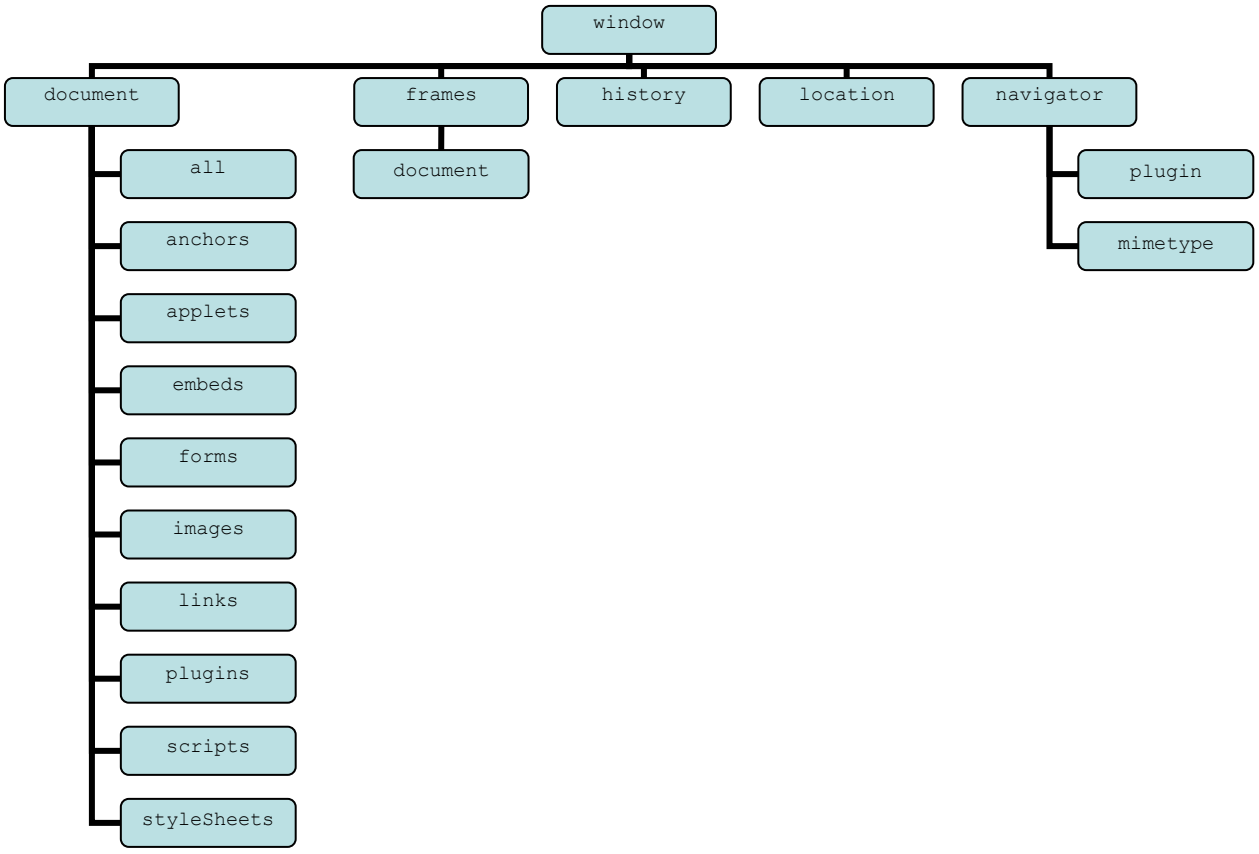

Рис. 2. Объектная модель MS Internet Explorer

Для каждого типа элементов в модели предусмотрены соответствующие наборы. Например, для объектов image, определяемых тэгами <IMG>, существует набор images. Ссылку на соответствующий объект можно определить с использованием имени объекта, задаваемого значением параметра NAME, или с помощью набора объектов, в данном случае images. В наборе объекты расположены в последовательности, в которой они задаются на HTML-странице.

Список литературы

1. Матросов А.В., Сергеев А.О, Чаунин М.П. HTML 4.0. – СПб.: БХВ-Петербург, 2000.
2. Кенин А.М., Печенкина Н.С. Новый уровень создания HTML-документов. – Екатеринбург: Деловая книга, 1996.
3. Дэвис С. Языки JavaScript и VBScript. – К.: Диалектика, 1996.
4. Микляев А. Основы HTML. – М.: Солон, 1998.
5. Зубкова С.В. Интерактивные Web-документы. – М.: ДМК Пресс, 2000.
6. Ратбон Э. JavaScript для чайников. – К.: Диалектика, 1995.
7. Ставровский А.Б. Учебник по VBScript. – К.: BHV, 2000.