www.bhv.ru
www.bhv.kiev.ua

Марина Дмитриева

JavaScript

САМОУЧИТЕЛЬ

Основы программирования

Решение вычислительных задач

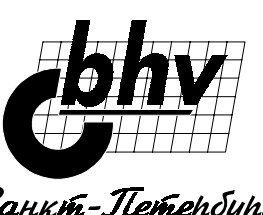
Принципы создания интерактивного Web-сайта

Искусственный интеллект в сценариях

Универсальный язык программирования для создания интерактивных Web-страниц

Марина Дмитриева

САМОУЧИТЕЛЬ
JavaScript



Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Рассматриваются основы программирования на языке JavaScript, базовые объекты и методы, элементы форм, размещаемых на Web-страницах, методы решения задач. Приводятся сценарии и тексты HTML-кода. Изучаются способы представления знаний, понятие логического следствия, получение новых знаний из уже доказанных. Книга содержит примеры решения задач из различных областей: обработка символьной информации, численные расчеты, работа с изображениями, создание меню, обеспечение навигации по Web-документам.

Для программистов и Web-разработчиков

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Наталья Таркова*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Дмитриева М. В.
Самоучитель JavaScript — СПб.: БХВ-Петербург, 2001. — 512 с.: ил.

ISBN 5-94157-122-4

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 13.08.01.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 41,28.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Введение

Предлагаемая книга является самоучителем по одному из разделов в области технологий разработки Web-приложений. При создании интерактивных документов в качестве языка сценариев используется язык JavaScript. В нем удачно сочетаются основные понятия современного программирования, практичность и наличие средств поддержки идей объектно-ориентированного программирования. При написании книги ставилась задача предоставить сведения для получения элементарных практических навыков по составлению сценариев, на простых и содержательных примерах продемонстрировать основные идеи и методы, принятые в современном программировании.

Язык сценариев JavaScript предназначен для создания интерактивных HTML-документов. Обычно ему отводится роль поддержки диалога с пользователем, применения для создания сценариев, обеспечивающих привлекательный вид Web-страниц. Но JavaScript прекрасно подходит и для обучения основам программирования. Центральным аспектом языка является понятие объекта, поэтому уже при начальном изучении основ программирования учащийся вводится в круг представлений о современных технологиях. Для студентов гуманитарной направленности знание языка в предложенном объеме является достаточным для использования его в профессиональной деятельности при подготовке Web-документов.

Язык JavaScript позволяет обрабатывать исходные данные, представленные с помощью различных элементов управления, создавать тестирующие программы, осуществлять контроль вводимых данных и многое другое. Можно создавать как итеративные, так и рекурсивные сценарии, использовать процедурный тип данных для решения многих классов задач, организовывать работу со стандартными объектами, обрабатывать текстовую информацию и решать другие задачи.

Каждый из теоретических разделов книги содержит большое количество примеров решения задач из различных областей: обработка символьной информации, численные расчеты, работа с изображениями, создание меню, обеспечение навигации по Web-документам и др. В частности, рассматриваются алгоритмы, которые обсуждаются в школьном курсе информатики, представлены записи этих алгоритмов на языке JavaScript.

В конце каждой главы приводятся упражнения, которые предлагается выполнить самостоятельно для закрепления рассмотренного материала. Материал книги построен таким образом, что новые понятия объясняются в тот момент, когда все ранее описанные средства уже образуют некоторую сис-

тему. Таким образом, в зависимости от наличия времени допустимо изучение материала в объеме нескольких тем.

Первая часть книги ориентирована на читателей, не знакомых с программированием, но желающих получить простые навыки, позволяющие в одних случаях сделать свой Web-документ более привлекательным. В других случаях требуется написать сценарий для простого анализа данных, например, для обработки анкеты. Для таких читателей важно уметь создавать меню, помещать в документ изображения, извлекать расположенные на Web-странице данные, которые могут представляться с помощью переключателей, флажков, списков, текстовых полей и т. д. Вычисления в аналогичных задачах несложные, поэтому и реализованные сценарии будут простыми. В этой части подробно описываются создаваемые документы и используемые сценарии. Такая степень детализации может быть излишней для следующей предполагаемой категории читателей.

Вторая часть книги предназначена для читателей, желающих ознакомиться с принципами современного программирования, научиться создавать более сложные сценарии. Спор о том, какой язык программирования выбрать в качестве базового, так окончательно не решен и, видимо, не будет решен никогда. Общепризнанным языком при обучении основам программирования является язык Паскаль. Так как любой язык — это способ записи алгоритмов, то целесообразно использовать в качестве базового языка JavaScript, по крайней мере, для тех категорий пользователей, которым не потребуется работать со сложными структурами данных типа деревьев, графов и т. д. Основное внимание обращается на методы решения различных задач, обсуждаются итерационные алгоритмы, алгоритмы обработки текстовой информации, классические алгоритмы работы с массивами. Рассматриваются задачи, при решении которых осуществляется работа с датами, применяются численные методы и др. Во второй части рассматриваются такие объекты JavaScript, как `String`, `Data`, `Array` и те основные методы, которые предусмотрены в языке сценариев для работы со стандартными объектами. В книге обсуждаются решения задач, которые рассматриваются, как правило, в курсах, посвященных основам программирования.

И, наконец, третья категория читателей — изучившие основы программирования и умеющие писать программы на каком-либо из распространенных процедурных языков, может обратиться к последней части книги. Таких читателей, как правило, интересуют лишь отдельные аспекты использования JavaScript. Например, процедурный тип данных и его реализация в языке JavaScript, методы работы со строками, которых нет в других языках программирования, особенности создания массивов и методы, характерные лишь для языка JavaScript, и т. п. Для них в третьей части рассматриваются рекурсивные методы решения задач, приводятся примеры сценариев, использующих рекурсивные функции, обсуждается метод исчерпывающего перебора и его реализация при решении конкретных задач, а также рассмат-

риваются задачи, при решении которых используется метод рекурсивного спуска.

Особо хочется обратить внимание читателей на последние две главы. Они посвящены решению задач, традиционно относящихся к искусственному интеллекту, — одной из активно развивающихся областей информатики. Способы представления знаний, понятие логического следствия, получение новых знаний из уже доказанных — неполный перечень тем, которые рассматриваются в этих главах. В книге приводится необходимый для понимания проблем теоретический материал. Р. Смаллиан написал ряд занимательных книг по математической логике [15]. Для иллюстрации теоретического материала и в качестве упражнений для самостоятельного решения используются, в основном, задачи из этих увлекательных книг.

Отдельная глава посвящена методам поиска доказательства. Рассмотрены различные стратегии поиска, сформулированы задания на разработку сценариев поиска решений. JavaScript располагает мощными средствами для работы с формулами, воспользоваться которыми рекомендуется при решении аналогичных задач.

При изучении языка сценариев JavaScript предполагается, что читатель знаком с основами языка HTML, хотя не исключается возможность "параллельного" изучения HTML и JavaScript. В книге, как правило, представлены не только функции JavaScript, решающие задачу, но полностью HTML-код документа. Приведенные примеры служат для закрепления навыков создания HTML-страниц, поэтому полезна не только функция, но весь документ. Кроме того, во многих задачах требуется создание документа определенной заданной структуры. Более того, многие сценарии на языке JavaScript призваны "оживить" HTML, т. е. цель создания сценария заключается в том, чтобы продемонстрировать, как будет меняться вид страницы при изменении значений параметров HTML-тегов. Вместе собранные и должным образом оформленные такого рода сценарии являются примером разработанного Web-приложения.

Для некоторых задач приводится несколько вариантов решения, каждый из них отличается набором тех средств, которые к данному моменту рассмотрены, и которые читатель может использовать при построении сценария.

Для закрепления пройденного материала рекомендуется выполнять упражнения, предложенные в конце каждой главы и подобранные таким образом, чтобы не вызывать затруднения у тех, кто внимательно изучил и понял материал очередной главы. Задачи, составляющие упражнения, как правило, имеют одинаковую сложность и могут использоваться в качестве индивидуальных заданий при обучении студентов в компьютерном классе.

При работе с книгой предполагается, что читатель знаком именно с основами HTML. Если читатель знает каскадные таблицы стилей, то некоторые

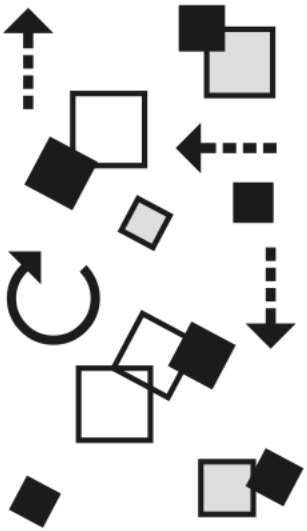
задачи можно решить, используя стили, иным способом, чем тот, что предложен в книге. Такие сценарии полезно написать в качестве упражнения.

Самоучитель разработан на основе опыта преподавания курса информатики для студентов математических и гуманитарных специальностей, для слушателей факультета повышения квалификации.

Я признательна студентам и аспирантам, которые не только прослушали мой курс лекций по технологии создания Web-публикаций, но и выполнили многочисленные индивидуальные задания и курсовые работы.

Я благодарна Сенину Владимиру Даниловичу и Керову Леониду Александровичу за предоставленную мне возможность работать по интересующей тематике. Хочу поблагодарить за сотрудничество Позднякова Сергея Николаевича, любезно согласившегося прочесть рукопись. Выражаю глубокую благодарность Лаврову Святославу Сергеевичу, впервые прочитавшему в университете курс лекций по искусственному интеллекту. Хочу выразить благодарность Рыбакову Евгению Евгеньевичу, после общения с которым возник замысел создания книги. Я признательна Дмитриеву Юрию Игоревичу за неоценимую помощь при работе над книгой и подготовке рукописи к публикации.

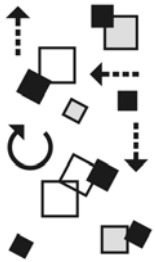
Надеюсь, что книга будет полезна школьникам, студентам младших курсов, слушателям курсов повышения квалификации и всем, кто интересуется вопросами создания и использования Web-приложений.



ЧАСТЬ I

Язык JavaScript для начинающих

Глава 1



Основные положения

Язык сценариев JavaScript

Язык программирования JavaScript разработан фирмой Netscape в сотрудничестве с Sun Microsystems и предназначен для создания интерактивных HTML-документов. Основные области использования JavaScript таковы:

- создание динамических страниц, т. е. страниц, содержимое которых может меняться после загрузки документа;
- проверка правильности заполнения пользователем форм до пересылки их на сервер;
- решение "локальных" задач с помощью сценариев

и некоторые другие сферы.

JavaScript позволяет создавать приложения, выполняемые как на стороне клиента, так и на стороне сервера. При разработке приложений обоих типов используется так называемое *ядро*, в котором содержатся определения стандартных объектов. Клиентские приложения выполняются браузером на машине пользователя.

Программа (сценарий) на языке JavaScript обрабатывается встроенным в браузер интерпретатором. Надо стремиться к тому, чтобы написанные сценарии корректно выполнялись в любом браузере. На первоначальном этапе обучения добиться удовлетворения этого требования сложно. Предлагаемые в книге сценарии отлаживались в Internet Explorer версии 4.01 и выше.

Программа (сценарий) на языке JavaScript представляет собой последовательность операторов. Если несколько операторов располагаются на одной строке, то между ними следует поставить знак "точка с запятой" (`;`). Если каждый оператор размещается на одной строке, то разделитель можно не писать. Один оператор может располагаться на нескольких строках.

Согласно принципам структурного программирования, программу рекомендуется записывать таким образом, чтобы в ней была отражена блочная структура. Это облегчает исследование программы и поиск ошибок.

В программах на JavaScript можно использовать комментарии. Для того чтобы задать комментарий, располагающийся на одной строке, достаточно перед его

текстом поставить две косые черты (//). Если же поясняющий текст занимает несколько строк, то его следует заключать между символами /* и */.

В JavaScript строчные и прописные буквы алфавита считаются разными символами.

Литералы

Простейшие данные, с которыми может оперировать программа, называются *литералами*. Литералы не могут изменяться. Литералы целого типа могут быть заданы в десятичном (по основанию 10), шестнадцатеричном (по основанию 16) или восьмеричном (по основанию 8) представлении.

Литерал целого типа в десятичном представлении записывается как последовательность десятичных цифр со знаком или без него, например, 15, 123, −156, +3567.

Шестнадцатеричные числа включают цифры 0—9 и буквы *a*, *b*, *c*, *d*, *e*, *f*. Шестнадцатеричные числа записываются с символами 0x перед числом, например, 0x25, 0xa1,0xff.

Восьмеричное число включает только цифры 0—7 и записывается, начиная с нуля, например, 03, 0543, 011.

Запись вещественного литерала отличается от записи вещественного числа в математике тем, что вместо запятой, отделяющей целую часть от дробной, указывается точка, например, 123.34, −22.56. Кроме того, для записи вещественных чисел можно использовать так называемую экспоненциальную форму, например, число 0,000000273, которое можно представить в виде 2.73×10^{-7}, в языке JavaScript записывается так: 2.73e−7. В такой записи знак умножения и число 10 заменяются символом e. В записи вещественного литерала должна присутствовать, по крайней мере, одна цифра и десятичная точка или символ экспоненты (e или E).

Кроме целых и вещественных значений в языке JavaScript могут встречаться так называемые логические значения. Существуют только два логических значения: истина и ложь. Первое представляется литералом true, второе — false. В некоторых реализациях JavaScript может быть использована единица в качестве true, и ноль в качестве false.

Строковый литерал представляется последовательностью символов, заключенной в одинарные или двойные кавычки. Примером строкового литерала может быть строка "результат" или 'результат'. Строковый литерал, представляющий пустую строку, обозначается как "" или ''.

Переменные

Переменные используются для хранения данных. Переменные в сценарии представляются с помощью идентификаторов. Идентификатор должен начинаться с буквы латинского алфавита, либо с символа подчеркивания. Далее может указываться последовательность, содержащая буквы латинского алфавита, цифры или знак подчеркивания, например, `test1`, `_my_test`, `test_1`. Тип переменной зависит от хранимых в ней данных, при изменении типа данных меняется тип переменной. Определить переменную можно с помощью оператора `var`, например:

```
var test1
```

В данном случае тип переменной `test1` не определен и станет известен только после присвоения переменной некоторого значения.

Оператор `var` можно использовать и для инициализации переменной, например, конструкцией

```
var test2=276
```

определяется переменная `test2` и ей присваивается значение `276`.

Значение переменной изменяется в результате выполнения оператора присваивания. Оператор присваивания может быть использован в любом месте программы и способен изменить не только значение, но и тип переменной. Оператор присваивания выглядит так

```
a=b
```

где `a` — переменная, которой мы хотим задать некоторое значение; `b` — выражение, определяющее новое значение переменной. Пусть в сценарии описаны следующие переменные

```
var n=3725
var x=2.75
var p=true
var s="Выполнение завершено"
```

Переменные `n` и `x` имеют тип `number`, тип переменной `p` — логический, переменная `s` имеет тип `string`. В JavaScript определен тип `function` для всех стандартных функций и функций, определяемых пользователем. Объекты JavaScript имеют тип данных `object`. Переменные типа `object` часто называют просто объектами, они могут хранить объекты.

Переменные, описанные в сценарии как в части `<HEAD>`, так и в части `<BODY>`, имеют одну и ту же область действия, доступны любому сценарию текущего документа. Такие переменные называются *глобальными* в отличие от *локальных* переменных, определенных в теле функции.

Выражения

Выражения строятся из литералов, переменных, знаков операций, скобок. В результате вычисления выражения получается единственное значение, которое может быть либо числом (целым или вещественным), либо строкой, либо логическим значением. Используемые в выражении переменные должны быть инициализированы. Если при вычислении выражения встречается неопределенная или неинициализированная переменная, то фиксируется ошибка. В JavaScript существует литерал `null` для обозначения неопределенного значения. Если переменной присвоено значение `null`, то она считается инициализированной.

Выражения формируются из операндов и обозначений операций. Например, в формуле `a*b` операндами являются `a` и `b`, обозначением операции — знак `*`.

Операции делятся на *унарные (одноместные)* или *бинарные (двуместные)*. Выражение записывается либо в виде ⊕A, если ⊕ — обозначение унарной операции, либо A⊕B, если ⊕ — обозначение бинарной операции. Вычисление выражения ⊕A сводится к вычислению операнда A и применению операции ⊕ к значению операнда. Вычисление выражения вида A⊕B состоит из следующих шагов:

1. Вычисляются A и B.
2. Операция ⊕ применяется к значению операндов, полученных на шаге 1.

В зависимости от типа вычисленного значения выражения можно разделить на арифметические, логические и строковые. Арифметические выражения получаются при выполнении операций, перечисленных в табл. 1.1.

Таблица 1.1. *Арифметические операции*

Операция	Название
+	Сложение
−	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения операнда на единицу
−−	Уменьшение значения операнда на единицу

Операторы в выражении вычисляются слева направо в соответствии с приоритетами арифметических операций. При необходимости с помощью скобок можно изменить порядок выполнения операций. В языке JavaScript определены операторы, в которых производятся арифметические действия над

левым и правым операндом и результат присваивается переменной, заданной левым операндом. Операции так называемой сокращенной формы присваивания представлены в табл. 1.2.

Таблица 1.2. *Сокращенные формы оператора присваивания*

Оператор	Эквивалентный оператор присваивания
`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`

Операции отношения применимы к операндам любого типа. Результат операции — логическое значение `true`, если сравнение верно, и `false` — в противном случае. Перечислим операции сравнения:

- `<` (меньше);
- `<=` (меньше или равно);
- `==` (равно);
- `!=` (не равно);
- `>=` (больше или равно);
- `>` (больше).

Операция `!` (логическое НЕ) применяется к операндам логического типа, если значение операнда `a` равно `true`, то значение выражения `!a` — `false`, если значение операнда `a` равно `false`, то значение выражения `!a` — `true`. Результат применения логических операций `&&` (логическое И) и `||` (логическое ИЛИ) приведен в табл. 1.3.

Таблица 1.3. *Логические операции*

A	B	A&&B	A||B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	False

Значение выражения `A&&B` истинно, если истинны оба операнда, и ложно в противном случае. Значение выражения `A||B` истинно, если значение хотя бы одного из операндов истинно, и ложно в противном случае.

Над строковыми значениями определена операция *конкатенация (соединение) строк*. Обозначается операция знаком плюс. Результатом выполнения этой операции является строка, состоящая из строковых значений операндов, например, в результате выполнения оператора присваивания

```
st = "текущее "+"состояние"
```

переменная `st` получит значение `"текущее состояние"`.

Рассмотрим еще один пример. Пусть выполнено

```
st1 = "текущий "
st2 = "момент"
```

В результате выполнения

```
st1 += st2
```

переменная `st1` получит значение `"текущий момент"`.

Приоритет операций определяет порядок, в котором выполняются операции в выражении. В табл. 1.4 перечислены рассмотренные операции в порядке убывания приоритетов.

Таблица 1.4. *Таблица приоритетов операций*

Название	Обозначение
Инкремент	`++`
Декремент	`--`
Отрицание	`!`
Унарный минус	`-`
Умножение	`*`
Деление, остаток от деления	`/, %`
Сложение	`+`
Вычитание	`-`
Сравнение	`<, >, <=, >=`
Равенство	`==`
Неравенство	`!=`
Логическое И	`&&`
Логическое ИЛИ	`||`
Присваивание	`=, +=, -=, *=, /=, %=, !=`

Упражнения

1. Напишите выражение, истинное тогда и только тогда, когда:
   - значение целой переменной `M` делится нацело на значение целой переменной `K`;
   - значения вещественных переменных `A`, `B` и `C` образуют неубывающую последовательность;
   - значение переменной `X` является наибольшим из трех попарно различных значений `X`, `Y`, `Z`;
   - ни одна из логических переменных `A`, `B`, `C` не истинна;
   - по крайней мере одна из логических переменных `A`, `B`, `C` истинна.
2. Начертите и заштрихуйте область, такую, что заданное выражение, в котором значения `X` и `Y` трактуются как координаты точки на плоскости, принимает значение `true`:
   - `(X*Y>0)`;
   - `Y+X<5 && X*X+Y*Y>1`;
   - `Y<X*X+2 || Y>6`.
3. Напишите формулу, истинную тогда и только тогда, когда точка на плоскости с координатами `X` и `Y` попадает в заштрихованную область (рис. 1.1—1.4).

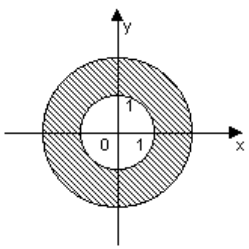


Рис. 1.1. Точка попадает в область, образованную исключением двух кругов

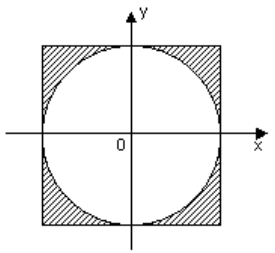


Рис. 1.2. Точка попадает в область, образованную исключением квадрата и круга

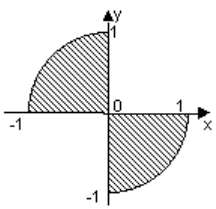


Рис. 1.3. Точка попадает в область, образованную двумя секторами

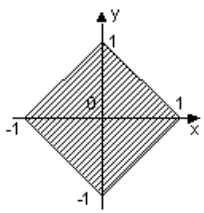


Рис. 1.4. Точка попадает в область, образованную ромбом

Сценарии в HTML-документе

Сценарии, написанные на языке JavaScript, могут располагаться непосредственно в HTML-документе между тегами `<script>` и `</script>`.

Одним из параметров тега `<script>` является `language`, который определяет используемый язык сценариев. Для языка JavaScript значение параметра равно `"JavaScript"`. Если применяется язык сценариев VBScript, то значение параметра должно быть равным `"VBScript"`. В случае использования языка JavaScript параметр `language` можно опускать, т. к. этот язык выбирается браузером по умолчанию.

Обычно браузеры, не поддерживающие какие-либо теги HTML, эти теги просто игнорируют. Попытка браузера проанализировать содержимое не поддерживаемых тегов может привести к неверному отображению страницы. Чтобы избежать такой ситуации, рекомендуется помещать операторы языка JavaScript в теги комментария `<!-- ... -->`. Для правильной работы интерпретатора перед закрывающим тегом комментария `-->` следует поставить символы `//`.

Итак, для размещения сценария в HTML-документе следует написать следующее:

```
<script language="JavaScript">
<!--
  Операторы языка JavaScript
//-->
</script>
```

Документ может содержать несколько тегов `<script>`. Все они последовательно обрабатываются интерпретатором JavaScript. В следующем примере в раздел `<BODY>` (в тело) HTML-документа вставлены операторы языка JavaScript.

Вычисление площади треугольника

Необходимо написать сценарий, определяющий площадь прямоугольного треугольника по заданным катетам. Сценарий разместим в разделе `<BODY>` HTML-документа (листинг 1.1).

Листинг 1.1. Первый сценарий в документе

```
<HTML>
  <HEAD>
    <TITLE>Первый сценарий в документе</TITLE>
  </HEAD>
```

```
  <BODY>
    <P>Страница, содержащая сценарий.</P>
    <script>
    <!--
      var a=8; h=10
      document.write("Площадь прямоугольного треугольника равна ",
a*h/2,".")
    //-->
    </script>
    <P>Конец формирования страницы, содержащей сценарий</P>
  </BODY>
</HTML>
```

В сценарии описываются и инициализируются две переменные, затем значение выражения записывается в документ. Для формирования вывода в HTML-страницу используется метод `write` объекта `document`. Строки, записываемые в документ, могут включать в себя теги HTML и выражения JavaScript.

Тег `<noscript>` определяет HTML-код, отображаемый на экране в случае, если JavaScript не поддерживается браузером или поддержка отключена. Этот тег следует после кода, заключенного в теги `<script>` и `</script>`. Если поддержка включена, то тег `<noscript>` игнорируется.

В дальнейших примерах будем считать, что поддержка JavaScript включена.

Функции: описание и использование

При создании программы разумно выделить в ней логически независимые части (так называемые *подпрограммы*). Каждую часть при необходимости можно разбить на отдельные подпрограммы и т. д. Разбиение программы на подпрограммы облегчает процесс отладки, т. к. позволяет отлаживать каждую подпрограмму отдельно. Имеет смысл распределить работу по созданию сложной программы между отдельными программистами. Некоторые подпрограммы можно использовать для решения разных задач.

Один раз созданную и отлаженную программу можно использовать произвольное число раз.

Во многих языках программирования понятие подпрограммы реализуется с помощью конструкций процедур, функций, модулей и т. п.

Основным элементом языка JavaScript является *функция*. Описание функции имеет вид

```
function F (V) {S}
```

где F — идентификатор функции, задающий имя, по которому можно обращаться к функции; V — список параметров функции, разделяемых запятыми; S — тело функции, в нем задаются действия, которые нужно выполнить, чтобы получить результат. Необязательный оператор return определяет возвращаемое функцией значение.

Описание функции не может быть вложено в описание другой функции. Параметры функции внутри ее тела играют ту же роль, что и обычные переменные, но начальные значения этим параметрам присваиваются при обращении к функции. Если описание функции имеет вид

```
function F (v1, v2, ..., vn) {S}
```

то вызов функции должен иметь вид

```
F (e1, e2, ..., en)
```

где e1, e2, ..., en — выражения, задающие *фактические* значения параметров. Параметры v1, v2, ..., vn, указанные в описании функции, называются *формальными* параметрами, чтобы подчеркнуть тот факт, что они получают смысл только после задания в вызове функции фактических параметров e1, e2, ..., en, с которыми функция затем и работает. Если в функции параметры отсутствуют, то описание функции имеет вид

```
function F (){S}
```

Наличие скобок в операторе вызова функции обязательно, т. е. вызов функции в этом случае должен иметь вид:

```
F()
```

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

В следующем примере в разделе заголовка описана функция care, вычисляющая площадь прямоугольного треугольника по заданным катетам.

Сценарий с функцией

Необходимо написать сценарий, определяющий площадь треугольника по заданному основанию и высоте. Для вычисления площади воспользуемся функцией, описанной в разделе <HEAD> HTML-документа (листинг 1.2).

Листинг 1.2. Использование сценария с функцией

```
<HTML>
  <HEAD>
    <TITLE>Использование сценария с функцией</TITLE>
    <script language="JavaScript">
```

```
    <!-- //
      function care (a, h)
      { return a*h/2 }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Начало отображения страницы со сценарием и функцией.</P>
    <script>
    <!--
      var a1=4; h1=16
      var s=care (a1,h1)
      document.write("При вызове функции получено значение ", s,".");
    //-->
    </script>
    <P>Конец формирования страницы.
  </BODY>
</HTML>
```

Тело функции состоит лишь из оператора `return`, который определяет возвращаемое функцией значение. Вызов функции осуществляется в теле документа при выполнении оператора присваивания: `s=care(a1,h1)`. Формальным параметрам `a` и `h` присваивается значение фактических параметров `a1` и `h1`, и выполняется тело функции. Полученное значение помещается в документ с помощью метода `write`.

Обработчики событий

В предыдущих примерах пользователю не предоставлялась возможность вводить значения, и в зависимости от них получать результат, например, при выполнении функции.

Интерактивные документы можно создавать, используя формы. Предположим, что мы хотим создать форму, в которой поля **Основание** и **Высота** служат для ввода соответствующих значений. Кроме того, в форме создадим кнопку **Вычислить**. При щелчке мышью по этой кнопке мы хотим получить значение площади треугольника.

Действие пользователя (например, щелчок кнопкой мыши) вызывает событие. События в основном связаны с действиями, производимыми пользователем с элементами форм HTML. Обычно перехват и обработка события задается в параметрах элементов форм. Имя параметра обработки события

начинается с приставки on, за которой следует имя самого события. Например, параметр обработки события Click будет выглядеть как onClick.

Значением параметра обработки события могут быть операторы языка JavaScript. В качестве значения параметра обработки события можно задать вызов функции, которая должна выполняться при возникновении события, определяемого параметром обработки события. В этом случае может быть использована следующая форма:

```
<FORM name="form1">
  Основание: <input type="text" size=5 name="st1"><hr>
  Высота:    <input type="text" size=5 name="st2"><hr>
  <input type="button" value=Вычислить
  onClick="care(document.form1.st1.value,document.form1.st2.value)">
</FORM>
```

Обработка значений из формы

Напишем функцию, вычисляющую площадь треугольника по заданному основанию и высоте. Создадим форму для ввода исходных данных.

HTML-код представлен в листинге 1.3.

Листинг 1.3. Обработка значений из формы

```
<HTML>
  <HEAD>
    <TITLE>Обработка значений из формы</TITLE>
    <script language="JavaScript">
    <!-- //
      function care (a, h)
      { var s= (a* h) / 2;
      document.write ("Площадь прямоугольного треугольника равна ",s);
      return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Пример сценария со значениями из формы</P>
    <FORM name="form1">
      Основание: <input type="text" size=5 name="st1"><hr>
      Высота:    <input type="text" size=5 name="st2"><hr>
```

```
    <input type="button" value=Вычислить
    onClick="care(document.form1.st1.value,document.form1.st2.value)">
   </FORM>
  </BODY>
</HTML>
```

Рассмотрим подробнее значение параметра обработки события, представляющего собой вызов функции `care`.

При интерпретации HTML-страницы браузером создаются объекты JavaScript. Взаимосвязь объектов между собой представляет иерархическую структуру. На самом верхнем уровне иерархии находится объект `windows`, представляющий окно браузера. Объект `windows` является *предком* или *родителем* всех остальных объектов. Каждая страница кроме объекта `windows` имеет объект `document`. Свойства объекта `document` определяются содержимым самого документа: цвет фона, цвет шрифта и т. д. В последнем примере на странице расположена форма. Форма (`form`) является *потомком* объекта `document`, а все элементы формы выступают потомками объекта `form`. Ссылка на объект может быть осуществлена по имени, заданному параметром `name` тега `<HTML>`. Для получения значения основания треугольника, введенного в первом поле формы, должна быть выполнена конструкция

```
document.form1.st1.value
```

При ссылке на формы и их элементы можно не указывать объект `document`. В рассмотренном примере получить значение первого поля ввода можно и следующим образом

```
form1.st1.value
```

Итак, когда в функцию передаются данные простых типов, например, чисел, как в рассмотренном случае, передача параметров осуществляется *по значению*. Формальному параметру `a` присваивается значение фактического параметра `form1.st1.value`, а формальному параметру `b` значение `form1.st2.value`. После этого выполняется тело функции.

Ситуация изменится, когда фактическим параметром функции станет объект. В этом случае говорят, что передача параметра осуществляется *по ссылке* или *по наименованию*. Пусть тело документа описано следующим образом:

```
<BODY>
  <P>Вычисление площади прямоугольного треугольника</P>
  <FORM name="form1">
    Основание: <input type="text" size=7 name="st1"><hr>
    Высота:    <input type="text" size=7 name="st2"><hr>
    <input type="button" value=Вычислить
```