JavaScript

The Definitive Guide

Fourth Edition

David Flanagan

O'REILLY®

JavaScript

Подробное руководство

Четвертое издание

Дэвид Флэнаган

Санкт-Петербург — Москва
2004

Дэвид Флэнаган

JavaScript. Подробное руководство, 4-е издание

Перевод Л. Фрейдина

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Королев*
Редактор	*В. Овчинников*
Корректура	*С. Беляева, С. Журавина*
Верстка	*В. Овчинников*

Флэнаган Д.
JavaScript. Подробное руководство. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 960 с., ил.
ISBN 5-93286-055-3

Четвертое издание бестселлера по JavaScript содержит полное описание базового языка JavaScript, а также традиционной и стандартизованной объектных моделей документа, реализованных в веб-броузерах. Примеры, включенные в книгу, можно использовать для решения распространенных задач, таких как проверка данных формы, работа с cookies и создание переносимой анимации DHTML. Части с III по V представляют собой подробные справочники по базовому API JavaScript, традиционному клиентскому API и стандартизованному API W3C DOM, в которых описываются все объекты, методы, свойства, конструкторы, константы, функции и обработчики событий этих API. Издание дополнено с учетом возможностей JavaScript 1.5 (ECMAScript v3) и содержит описание стандарта W3C DOM (Level 1 и Level 2), при этом для обратной совместимости сохранен материал по традиционному DOM Level 0.

Эта книга необходима каждому, кто пишет на JavaScript, независимо от его опыта. Она будет особенно полезна тем, кто работает с последними, соответствующими стандартам веб-броузерами, такими как Internet Explorer 6, Netscape 6 и Mozilla. Вебмастеры узнают, как применять JavaScript для построения динамических веб-страниц. Опытные разработчики смогут быстро приступить к написанию сложных программ.

ISBN 5-93286-055-3
ISBN 0-596-00048-0 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 18.02.2004. Формат 70×100 1/16 . Печать офсетная.
Объем 60 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Эта книга посвящается всем,
кто учит жить мирно и противостоит насилию.

Оглавление

Предисловие

Со времени публикации третьего издания этой книги в мире веб-программирования на JavaScript™ произошло много изменений, в том числе:

- Публикация второго и третьего изданий стандарта ECMA-262, обновивших ядро языка JavaScript. Были выпущены соответствующие версии интерпретаторов Netscape JavaScript и Microsoft JScript.
- Исходные тексты интерпретаторов JavaScript от Netscape (один написан на C, другой на Java™) выпущены в виде открытого исходного кода и доступны для всех, кто хочет встроить язык сценариев в свое приложение.
- World Wide Web Consortium (W3C) опубликовал две версии (два уровня) стандарта Document Object Model (DOM). Новейшие броузеры поддерживают этот стандарт (в разной степени) и позволяют клиентскому коду JavaScript взаимодействовать с содержимым документа и создавать сложные эффекты Dynamic HTML (DHTML). Широкую поддержку получили и другие стандарты W3C, такие как HTML 4, CSS1 и CSS2.
- Организация Mozilla, взяв за основу предоставленный Netscape исходный код, создала хороший броузер пятого поколения. Во время написания этой книги броузер Mozilla еще не дошел до уровня версии 1.0, но он уже достаточно зрелый для того, чтобы компания Netscape использовала базу кода Mozilla в качестве базы для своих броузеров версий 6.0 и 6.1.
- Microsoft Internet Explorer стал доминировать на настольных системах. Однако броузер Netscape/Mozilla по-прежнему важен для веб-разработчиков, в особенности по причине отличной поддержки веб-стандартов. Кроме того, как не менее важные надо рассматривать и второстепенные броузеры, такие как Opera (*http://www.opera.com*) и Konquerer (*http://www.konquerer.org*).
- Веб-броузеры (и интерпретаторы JavaScript) больше не ограничены настольными системами и работают даже на PDA и сотовых телефонах.

В итоге ядро языка JavaScript стало более зрелым. Язык стал стандартизованным и используется в более разнообразных средах, чем раньше. Коллапс доли рынка Netscape сделал возможным расширение ниши настольных веб-броузеров, а те из них, в которых реализована поддержка JavaScript, также стали доступными на платформах, отличных от настольных. Произошел явный, если не полный, переход к веб-стандартам. Частичная или полная ре-

ализация стандарта DOM в новейших броузерах дает веб-разработчикам давно ожидаемый независимый интерфейс API.

Что нового в 4-м издании

Это издание «JavaScript: The Definitive Guide» было тщательно переработано в свете только что описанных изменений. Самые важные из них включают полное описание JavaScript 1.5 и третьего издания стандарта ECMA-262, на котором он основан, а также полное описание стандарта DOM Level 2.

Фокус внимания смещен от конкретных реализаций JavaScript и броузеров (JavaScript 1.2, Netscape 4, IE 5 и т. д.) к документированию стандартов, на которых базируются (или должны базироваться) эти реализации. По причине множественности реализаций теперь практически невозможно описать в одной книге все возможности, фирменные расширения, индивидуальные особенности и ошибки всех реализаций, а любому разработчику – понять их. Книга ориентирована на спецификации, а не на реализации, что делает работу с ней проще, и если вы примете тот же подход, то ваш код JavaScript станет более переносимым и легким в сопровождении. Особое внимание в новом материале уделено стандартам по ядру JavaScript и DOM.

Другое существенное изменение в этом издании касается разбиения справочного раздела на три отдельные части. Материал по базовому JavaScript был отделен от материала по клиентскому JavaScript (часть IV) и помещен в самостоятельный раздел (часть III). Это сделано для удобства JavaScript-программистов, работающих с языком в среде, отличной от веб-броузера, и не интересующихся клиентским JavaScript.

Новый материал с описанием W3C DOM помещен в самостоятельный раздел (часть V), отдельно от уже известного материала по клиентскому JavaScript. Стандарт DOM определяет API, значительно отличающийся от «устаревшего» API традиционного клиентского JavaScript. В зависимости от целевого броузера разработчики выбирают тот или иной API и обычно не переключаются с одного на другой. Раздельное описание этих двух API также позволяет сохранить организацию справочного материала по клиентскому JavaScript (часть IV), удобную для читателей, знакомых с третьим изданием.

Чтобы освободить место для нового материала и не увеличивать чрезмерно объем книги, я убрал описания тривиальных свойств объектов, повторяющие материал разделов, посвященных объектам. Всем свойствам, которые требуют основательного рассмотрения, а также всем методам отведены специальные разделы. Кроме того, волшебники из O'Reilly заново оформили книгу; она стала меньше, а читать ее по-прежнему легко и удобно.

Типографские соглашения

В этой книге приняты следующие соглашения:

Шрифт OfficinaSansC

Применяется для выделения клавиш клавиатуры или элементов пользовательского интерфейса, таких как кнопка Back или меню Options.

Курсив

Обозначает первое вхождение термина. *Курсив* также применяется для адресов электронной почты, веб-сайтов, FTP-сайтов, имен файлов и каталогов, а также групп новостей. Кроме того, в этой книге *курсивом* выделяются имена Java-классов, чтобы их нельзя было спутать с именами JavaScript.

Моноширинный шрифт

Применяется для форматирования кода JavaScript, листингов HTML и вообще всего, что непосредственно набирается на клавиатуре при программировании.

Моноширинный курсив

Обозначает аргументы функций и элементы, которые в программе необходимо заменить на реальные значения.

Ошибки

Для улучшения будущих изданий и тиражей этой книги издательство O'Reilly & Associates просит сообщать о любых ошибках, неточностях, ложных или сбивающих с толку утверждениях и обычных опечатках. O'Reilly поддерживает веб-сайт, где можно найти список всех известных ошибок в этой книге:

http://www.oreilly.com/catalog/jscript4/errata/

На этой странице имеется ссылка на форму, заполнив которую можно отправить сообщение о найденной ошибке. Кроме того, можно сообщить об ошибках или задать вопросы по этой книге, послав письмо по электронной почте на адрес:

bookquestions@oreilly.com

Примеры в Сети

Примеры, приведенные в этой книге, можно загрузить с веб-сайта издательства, пройдя по ссылке *Examples* на странице:

http://www.oreilly.com/catalog/jscript4/

Комментарии и вопросы

Просьба присылать комментарии и вопросы по этой книге издателю:

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (in the United States or Canada)

(707) 829-0515 (international/local)

(707) 829-0104 (fax)

Технические вопросы и комментарии по этой и другим книгам издательства O'Reilly можно также посылать по адресу:

bookquestions@oreilly.com

Дополнительная информация о книгах, конференциях, Resource Center и O'Reilly Network приведена на сайте издательства O'Reilly:

http://www.oreilly.com

Благодарности

Брендан Эйх (Brendan Eich) из Mozilla – один из создателей и главный новатор JavaScript. Я и другие программисты на JavaScript в огромном долгу перед ним за разработку JavaScript и за то, что в его сумасшедшем графике нашлось время для ответов на наши вопросы, причем он даже требовал еще вопросов. Кроме того, что Брендан терпеливо отвечал на мои многочисленные вопросы, он также прочитал первое и третье издания этой книги и дал очень полезные комментарии к ним.

Это руководство получило благословение первоклассных технических рецензентов, комментарии которых весьма способствовали улучшению этой книги. Валдемар Хорват (Waldermar Horwat) из Netscape рецензировал новый материал по JavaScript. Материал по W3C DOM прорецензирован Филиппом Ле Хегаре (Philippe Le Hegaret) из W3C, Питером-Паулем Кохом (Peter-Paul Koch), руководителем отдела программирования клиентских приложений в голландской компании Netlinq Framfab (*http://www.netlinqframfab.nl*), специализирующейся на интернет-консалтинге, разработке и внедрении цифровых технологий в Интернете и маркетинге, Диланом Шиманом (Dylan Schiemann) из SitePen (*http://www.sitepen.com*), а также независимым веб-разработчиком Джеффом Ятсом (Jeff Yates). Питер-Пауль и Джефф поддерживают веб-сайты, посвященные веб-дизайну и DOM – *http://www.xs4all.nl/~ppk/js/* и *http://www.pbwizard.com* соответственно. Джозеф Кесселман (Joseph Kesselman) из IBM Research ничего не рецензировал, но очень помог мне, отвечая на вопросы по W3C DOM.

Третье издание книги рецензировалось Бренданом Эйхом, Валдемаром Хорватом и Вайдуром Аппарао (Vidur Apparao) из Netscape, Германом Вентером

(Herman Venter) из Microsoft, двумя независимыми разработчиками JavaScript – Джеем Ходжесом (Jay Hodges) и Анжело Сиригосом (Angelo Sirigos). Дэн Шейфер (Dan Shafer) из CNET Builder.com выполнил некоторую предварительную работу по третьему изданию. Его материал не нашел применения в этом издании, но принадлежащие ему идеи и общие принципы принесли большую пользу. Норрис Бойд (Norris Boyd) и Скотт Фурман (Scott Furman) из Netscape, а также Скотт Айзекс (Scott Issacs) из Microsoft нашли время, чтобы поговорить со мной о будущем стандарте Document Object Model. И наконец, доктор Танкред Хиршманн (Dr. Tankred Hirschmann) показал глубокое понимание хитросплетений JavaScript 1.2.

Второе издание много выиграло от помощи и комментариев Ника Томпсона (Nick Thompson) и Ричарда Якера (Richard Yaker) из Netscape, доктора Шона Каценбергера (Dr. Shon Katzenberger), Ларри Салливана (Larry Sullivan) и Дэйва С. Митчелла (Dave C. Mitchell) из Microsoft, Линн Роллинс (Lynn Rollins) из R&B Communications. Первое издание рецензировалось Нилом Беркманом (Neil Berkman) из Bay Networks, Эндрю Шульманом (Andrew Schulman) и Терри Алленом (Terry Allen) из O'Reilly & Associates.

Эта книга также стала лучше благодаря многочисленным редакторам, которые над ней работали. Паула Фергюсон (Paula Ferguson) – редактор этого и третьего издания. Она тщательно просмотрела книгу, сделав ее более понятной и легкой для чтения. Френк Уиллисон (Frank Willison) редактировал второе издание, а Эндрю Шульман – первое.

И наконец, мои благодарности Кристи – как всегда и за очень многое.

Дэвид Флэнаган
сентябрь 2001

1

Введение в JavaScript

JavaScript – это облегченный, интерпретируемый язык программирования с объектно-ориентированными возможностями. Универсальное ядро языка было встроено в Netscape, Internet Explorer и другие веб-броузеры, а добавление в него объектов, представляющих окно веб-броузера и содержимое последнего, сделало язык привлекательным для веб-программирования. Эта клиентская версия JavaScript позволяет включать в веб-страницы исполняемое содержимое, то есть веб-страница более не обязана быть статическим HTML-кодом, а может включать в себя программы, взаимодействующие с пользователем, управляющие броузером и динамически создающие HTML.

С точки зрения синтаксиса базовый язык JavaScript напоминает C, C++ и Java такими программными конструкциями, как инструкция `if`, цикл `while` и оператор `&&`. Однако это подобие ограничивается синтаксической схожестью. JavaScript – это нетипизированный язык, то есть в нем не определяется тип переменных. Объекты в JavaScript больше похожи на ассоциативные массивы Perl,[1] чем на структуры C или объекты C++ или Java. Механизм объектно-ориентированного наследования JavaScript похож на механизм в малоизвестных языках Self и NewtonScript и сильно отличается от наследования в C++ и Java. Как и Perl, JavaScript – это интерпретируемый язык, и некоторые его инструменты, например регулярные выражения и средства работы с массивами, вдохновлены языком Perl.

В этой главе приводится краткий обзор JavaScript, объясняется, что JavaScript может делать и что не может, а также разоблачаются некоторые мифы. Здесь базовый язык JavaScript отделяется от встроенных и расширен-

[1] Ассоциативные массивы Perl также известны под названием «хеши». – *Примеч. науч. ред.*

ных версий, таких как клиентский JavaScript, встроенный в веб-броузеры, и серверный JavaScript, встроенный в веб-серверы Netscape (здесь описываются базовый и клиентский JavaScript). Кроме того, в данной главе на нескольких фрагментах клиентского JavaScript демонстрируется практическое веб-программирование.

1.1. Мифы о JavaScript

Вокруг JavaScript довольно много дезинформации и путаницы. Прежде чем двигаться дальше в изучении JavaScript, важно развенчать некоторые распространенные мифы, связанные с этим языком.

1.1.1. JavaScript – это не Java

Одно из наиболее распространенных заблуждений о JavaScript состоит в том, что он представляет собой упрощенную версию Java, языка программирования от компании Sun Microsystems. Кроме некоторой синтаксической схожести и способности предоставлять исполняемое содержимое для веб-броузеров, эти два языка между собой ничто не связывает. Схожесть имен – не более чем уловка маркетологов (первоначальное название языка – LiveScript – было изменено на JavaScript в последнюю минуту).

Однако JavaScript и Java действительно хорошо дополняют друг друга, различаясь по своим возможностям. JavaScript может управлять поведением броузера и содержимым страницы, однако не может рисовать графику и работать с сетью. Java не может управлять броузером в целом, но может создавать графику, работать с сетью и позволяет реализовать многопоточность. Клиентский JavaScript может взаимодействовать и управлять встроенными в веб-страницу Java-апплетами и, в этом смысле, реализовать сценарии Java (подробности см. в главе 22).

1.1.2. JavaScript не простой язык

JavaScript позиционируется как язык сценариев, а не язык программирования, при этом подразумевается, что языки сценариев проще и предназначены для непрограммистов. В самом деле, на первый взгляд JavaScript может показаться довольно простым языком, по сложности сравнимым с BASIC. Некоторые возможности JavaScript призваны сделать язык менее строгим и более легким для новичков и неопытных программистов. Непрограммисты могут применять JavaScript для ограниченного круга задач, выполняемых по точным рецептам.

Однако за внешней простотой JavaScript скрывается полноценный язык программирования, столь же сложный, как любой другой, и даже более сложный, чем некоторые. Программисты, пытающиеся решать с помощью JavaScript нетривиальные задачи, часто разочаровываются в процессе разработки из-за того, что недостаточно понимают возможности этого языка.

Данная книга содержит всеобъемлющее описание JavaScript, позволяющее вам стать искушенным знатоком.

1.2. Версии JavaScript

JavaScript развивался много лет, и компания Netscape выпустила несколько версий этого языка. Microsoft выпустила похожие версии языка JavaScript под именем «Jscript». Организация ECMA (*http://www.ecma.ch*) опубликовала три версии ECMA-262, стандартизующие язык JavaScript под неуклюжим названием «ECMAScript».

Версии языка, их ключевые возможности и связь друг с другом описаны в табл. 1.1. В этой книге под названием «JavaScript» я часто понимаю любую реализацию языка, в том числе Microsoft JScript. Для того чтобы сослаться на ECMAScript явно, я употребляю термины «ECMA-262» или «ECMA».

Таблица 1.1. Версии JavaScript

Версия	Описание
JavaScript 1.0	Первоначальная версия языка. Содержала много ошибок и сейчас признана совершенно устаревшей. Реализована в Netscape 2.
JavaScript 1.1	Включает настоящий объект Array; большинство серьезных ошибок устранены. Реализована в Netscape 3.
JavaScript 1.2	Введена инструкция `switch`, регулярные выражения и некоторые другие возможности. В основном совместима с ECMA v1, однако содержит некоторые несоответствия. Реализована в Netscape 4.
JavaScript 1.3	Исправление несовместимостей JavaScript 1.2. Версия совместима с ECMA v1. Реализована в Netscape 4.5.
JavaScript 1.4	Реализована только в серверных продуктах компании Netscape.
JavaScript 1.5	Введена обработка исключений. Совместима с ECMA v3. Реализована в Mozilla и Netscape 6.
JScript 1.0	Практически эквивалентна JavaScript 1.0. Реализована в ранних выпусках IE 3.
JScript 2.0	Практически эквивалентна JavaScript 1.1. Реализована в поздних выпусках IE 3.
JScript 3.0	Практически эквивалентна JavaScript 1.3. Совместима с ECMA v1. Реализована в IE 4.
JScript 4.0	Не реализована ни в одном веб-броузере.
JScript 5.0	Поддерживает обработку исключений. Частично совместима с ECMA v3. Реализована в IE 5.
JScript 5.5	В основном эквивалентна JavaScript 1.5. Полностью совместима с ECMA v3. Реализована в IE 5.5 и IE 6. (IE 6 фактически реализует JScript 5.6, однако версия 5.6 ничем не отличается от 5.5 в том, что касается программистов, использующих клиентский JavaScript.)

Версия	Описание
ECMA v1	Первая стандартная версия языка. Стандартизованы базовые возможности JavaScript 1.1 и добавлены несколько новых возможностей. Не стандартизована инструкция `switch` и поддержка регулярных выражений. Соответствующие реализации – JavaScript 1.3 и Jscript 3.0.
ECMA v2	Промежуточная версия стандарта, включавшая некоторые разъяснения, но не определявшая новых возможностей.
ECMA v3	Стандатизована инструкция `switch`, регулярные выражения и обработка исключений. Соответствующие реализации – JavaScript 1.5 и JScript 5.5.

1.3. Клиентский JavaScript

Когда интерпретатор JavaScript встраивается в веб-броузер, результатом является клиентский JavaScript. Это, безусловно, наиболее распространенный вариант JavaScript, и большинство людей, упоминая JavaScript, обычно подразумевают именно клиентский JavaScript. В этой книге описывается клиентский JavaScript вместе с базовым языком JavaScript, который представляет собой подмножество клиентского JavaScript.

Далее в этой главе мы обсудим клиентский JavaScript и его возможности значительно более подробно. Если же говорить кратко, клиентский JavaScript объединяет способность интерпретатора JavaScript исполнять сценарии с объектной моделью документа (DOM), определяемой веб-броузером. Эти две различные технологии объединяются синергетически, поэтому результат превосходит сумму частей: клиентский JavaScript позволяет распространять исполняемое содержимое по сети и является основой нового поколения динамических HTML-документов (DHTML).

Спецификация ECMA-262 определила стандартную версию базового языка JavaScript, а организация World Wide Web Consortium (W3C) опубликовала спецификацию (или рекомендацию) DOM, стандартизующую возможности, которые броузер должен поддерживать в своей объектной модели. В главах с 17 по 19 содержится намного более подробное обсуждение этого стандарта. Хотя стандарт W3C DOM пока не поддерживается в полном объеме, существующей поддержки достаточно, чтобы веб-разработчики уже могли писать код JavaScript на его основе.

В табл. 1.2 представлена базовая версия языка и возможности DOM, поддерживаемые различными версиями броузеров от Netscape и Microsoft. Обратите внимание, что номера версий Internet Explorer, перечисленные в таблице, относятся к Windows-версиям этого броузера. Возможности версий IE для Macintosh часто отличаются (иногда в значительной степени) от версий с теми же номерами для Windows. Также имейте в виду, что IE позволяет обновлять интерпретатор JScript независимо от самого броузера, поэтому можно

встретить конфигурацию IE, поддерживающую более новую версию языка, чем указанная в таблице.

Таблица 1.2. Возможности клиентского JavaScript для разных броузеров

Броузер	Версия языка	Возможности DOM
Netscape 2	JavaScript 1.0	Работа с формами
Netscape 3	JavaScript 1.1	Переключение изображений
Netscape 4	JavaScript 1.2	DHTML с уровнями
Netscape 4.5	JavaScript 1.3	DHTML с уровнями
Netscape 6 / Mozilla	JavaScript 1.5	Поддержка стандарта W3C DOM в основном; прекращение поддержки уровней
IE 3	JScript 1.0/2.0	Работа с формами
IE 4	JScript 3.0	Переключение изображений; DHTML с `document.all[]`
IE 5	JScript 5.0	DHTML с `document.all[]`
IE 5.5	JScript 5.5	Частичная поддержка стандарта W3C DOM
IE 6	JScript 5.5	Частичная поддержка стандарта W3C DOM; отсутствие поддержки событийной модели W3C DOM

Различия и несовместимость между версиями клиентского JavaScript Netscape и Microsoft значительно превосходят различия между соответствующими реализациями базового языка. Однако довольно много средств клиентского JavaScript в обоих броузерах реализовано одинаково. За неимением лучших названий версии клиентского JavaScript часто обозначаются версиями базового языка, используемого в них. Таким образом, в контексте рассмотрения клиентских сценариев термин «JavaScript 1.2» относится к версии клиентского JavaScript, поддерживаемого в Netscape 4 и IE 4. Если номерами версий базового языка обозначаются клиентские версии JavaScript, подразумевается совместимое подмножество средств, поддерживаемых и Netscape и Internet Explorer. Когда я обсуждаю клиентские возможности броузера, я указываю его название и версию.

Заметьте, что Netscape и IE – не единственные броузеры, поддерживающие клиентский JavaScript. Например, Opera (*http://www.opera.com*) также поддерживает клиентский JavaScript. Однако поскольку Netscape и IE владеют львиной долей рынка броузеров, они и обсуждаются в этой книге. Реализации клиентского JavaScript в других броузерах должны достаточно близко соответствовать реализациям в IE и Netscape.

Кроме того, JavaScript – не единственный язык программирования, который может быть встроен в веб-броузер. Например, Internet Explorer поддерживает язык, известный как VBScript – вариант языка Microsoft Visual Basic, предоставляющий во многом те же возможности, что и JavaScript, но используемый только в броузерах компании Microsoft. Кроме того, в специфи-

кации HTML 4.0 при описании HTML-тега <script> в качестве примера встроенного языка сценариев приводится язык программирования Tcl. Среди основных броузеров ни один не поддерживает Tcl, но нет причины, по которой в броузере нельзя было бы реализовать такую поддержку.

В предыдущих изданиях этой книги броузеры Netscape рассматривались более подробно, чем броузеры Microsoft. Дело в том, что JavaScript был создан в Netscape, которая (по крайней мере в течение определенного времени) занимала доминирующую позицию на рынке веб-броузеров. Крен в сторону Netscape с каждым переизданием этой книги уменьшался, и здесь обсуждение в значительной степени сосредоточено на стандартах, таких как ECMAScript и W3C DOM, а не на конкретных броузерах. Тем не менее в материале, оставшемся от ранних изданий, все еще можно заметить некоторое преобладание сведений, относящихся к Netscape.

1.4. JavaScript в иных контекстах

JavaScript – это универсальный язык программирования, и его использование не ограничивается веб-броузерами. Он был разработан для встраивания в любое приложение и обеспечения возможности написания в нем сценариев. Фактически с самого начала веб-серверы Netscape включали интерпретатор JavaScript, что позволяло создавать серверные сценарии на этом языке. Аналогично, Microsoft применяет интерпретатор JScript в своем веб-сервере IIS и в продукте Windows Scripting Host, а не только в Internet Explorer.

И Netscape и Microsoft сделали свои интерпретаторы JavaScript доступными для компаний и программистов, которые хотят встраивать их в свои приложения. Интерпретатор Netscape был выпущен в виде открытого исходного кода и доступен теперь через организацию Mozilla (*http://www.mozilla.org/js/*). Фактически Mozilla предоставляет две различные версии интерпретатора JavaScript. Одна написана на C и называется «SpiderMonkey», а другая – на Java и, что лестно автору этой книги, называется «Rhino» (носорог).

Можно ожидать, что количество приложений, использующих JavaScript в качестве встроенного языка сценариев, будет постоянно увеличиваться.[1] Те, кто пишет сценарии для подобного приложения, найдут полезной первую часть этой книги, описывающую базовый язык. Тогда как материал глав, относящихся к веб-броузерам, видимо, им не понадобится.

[1] ActionScript, язык сценариев, доступный в Macromedia Flash 5 и MX, смоделирован в соответствии со стандартом ECMAScript, однако фактически не является JavaScript.

1.5. Клиентский JavaScript: исполняемое содержимое веб-страниц

Веб-броузер, дополненный интерпретатором JavaScript, позволяет распространять по Интернету исполняемое содержимое в виде сценариев JavaScript. В примере 1.1 показана простая JavaScript-программа, или сценарий, встроенный в веб-страницу.

Пример 1.1. Простая программа на JavaScript

```
<html>
<body>
<head><title>Factorials</title></head>
<script language="JavaScript">
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
    document.write(i + "! = " + fact);
    document.write("<br>");
}
</script>
</body>
</html>
```

Будучи загруженным в броузер, поддерживающий JavaScript, этот сценарий выдаст результат, показанный на рис. 1.1.

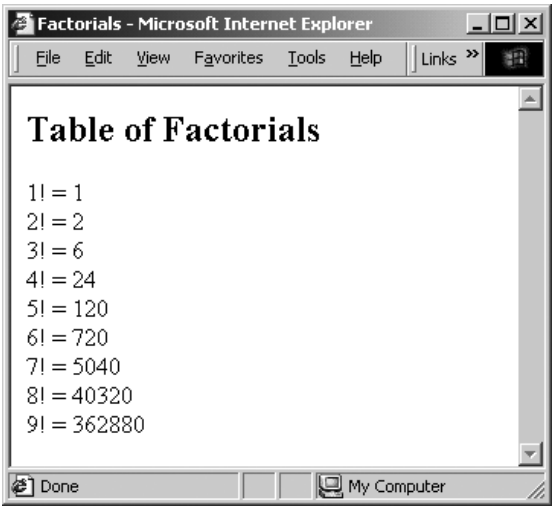

***Рис. 1.1.** Веб-страница, сгенерированная JavaScript*

Как видно из этого примера, для встраивания кода JavaScript в HTML-код применяются теги <script> и </script>. О теге <script> мы узнаем больше в главе 12. Главная возможность JavaScript, демонстрируемая данным при-

мером, – это использование метода `document.write()`.[1] Этот метод осуществляет динамический вывод HTML-текста, который анализируется и отображается веб-броузером, и мы еще не раз встретимся с ним в этой книге.

JavaScript обеспечивает возможность управления не только содержимым веб-страниц, но и броузером и содержимым HTML-форм, присутствующих в броузере. Далее в этой главе мы подробнее узнаем об этих возможностях JavaScript, а еще более подробно – в других главах этой книги.

JavaScript может управлять не только содержимым HTML-документов, но и поведением этих документов. Другими словами, JavaScript-программа может реагировать на ввод пользователем значения в текстовое поле ввода или щелчок по изображению в документе. Это достигается путем определения *обработчиков событий* для документа – фрагментов кода JavaScript, выполняемых при возникновении определенного события, например щелчка по кнопке. В примере 1.2 показана простая HTML-форма, включающая обработчик события, который выполняется в ответ на щелчок по кнопке.

Пример 1.2. HTML-форма, для которой определен обработчик события JavaScript

```
<form>
<input type="button"
       value="Click here"
       onclick="alert('You clicked the button');">
</form>
```

На рис. 1.2 отображен результат щелчка по кнопке.

***Рис. 1.2.** Отклик JavaScript на событие*

[1] Метод – это объектно-ориентированный термин, обозначающий функцию или процедуру.

Атрибут `onclick` из примера 1.2 изначально представлял собой расширение, введенное в HTML компанией Netscape специально для клиентского JavaScript. Однако теперь этот и другие атрибуты обработчиков событий стандартизованы в HTML версии 4.0. Все обработчики событий JavaScript определяются такими же HTML-атрибутами. Значение атрибута `onclick` — это строка кода JavaScript, выполняемая при щелчке по кнопке. В данном случае обработчик события `onclick` вызывает функцию `alert()`. Как видно на рис. 1.2, функция `alert()` выводит диалоговое окно, отображающее указанное сообщение.

Примеры 1.1 и 1.2 демонстрируют лишь простейшие возможности клиентского JavaScript. Реальная его мощь состоит в том, что сценарии имеют доступ к иерархии объектов, базирующихся на содержимом веб-страницы. Так, клиентские программы на JavaScript могут обращаться к любым изображениям, присутствующим в документе, и управлять ими, устанавливать связь и взаимодействовать с апплетами Java и другими объектами, встроенными в документ HTML. После освоения базового языка JavaScript ключом к его эффективному использованию на веб-страницах становится изучение предоставляемых броузером возможностей DOM.

1.6. Возможности клиентского JavaScript

Возможным применением JavaScript является написание программ, выполняющих произвольные вычисления. Например, можно написать простые сценарии для вычисления чисел Фибоначчи или для поиска простых чисел. Однако в контексте Интернета и веб-броузеров более интересным применением языка может быть программа, вычисляющая налог с продаж в онлайн-заказе на основе информации, введенной пользователем в HTML-форму. Как уже было сказано, настоящая мощь JavaScript состоит в поддержке языком объектов броузера и документа. Чтобы дать вам представления о потенциале JavaScript, в следующих разделах приведен список и описание наиболее важных возможностей клиентского JavaScript и поддерживаемых им объектов.

1.6.1. Управление внешним видом и содержимым документа

Объект Document языка JavaScript, как мы уже видели, позволяет в процессе обработки документа броузером выводить посредством метода `write()` произвольный код HTML. Так, можно включить в документ текущую дату и время или отображать различное содержимое на различных платформах.

С помощью объекта Document можно формировать документ целиком. Свойства объекта Document позволяют указывать цвет фона, текста и гиперссылок в документе. Это обеспечивает возможность формирования динамических и условных HTML-документов, что особенно полезно в многофреймовых документах. В самом деле, в некоторых случаях динамическая

генерация содержимого фрейма позволяет программе на JavaScript целиком заменить традиционные серверные сценарии.

Internet Explorer 4 и Netscape 4 поддерживают собственные методы создания эффектов Dynamic HTML, позволяющие динамически генерировать, перемещать и изменять содержимое документа. IE 4 также поддерживает полноценную модель DOM, предоставляющую JavaScript доступ к каждому отдельному элементу HTML в документе. IE 5.5 и Netscape 6 поддерживают стандарт W3C DOM (или по крайней мере его ключевые части), в котором задается общий, переносимый способ доступа ко всем элементам и тексту в HTML-документе, их размещению и изменению их внешнего вида путем манипуляции с атрибутами таблиц стилей Cascading Style Sheets (CSS). В этих броузерах клиентский JavaScript имеет полную власть над содержимым документа, открывая неограниченные возможности мира сценариев.

1.6.2. Управление броузером

Управлять поведением броузера можно посредством нескольких объектов JavaScript. Объект Window поддерживает методы отображения диалоговых окон с простыми сообщениями и полями для ввода пользовательских данных. В этом объекте также определен метод для создания и открытия (и закрытия) совершенно новых окон броузера, которые могут иметь любой заданный размер и любую комбинацию пользовательских элементов управления. Это позволяет, например, открывать несколько окон и дать пользователю несколько представлений вашего сайта. Новые окна броузера также полезны для временного отображения сгенерированного HTML и, будучи созданными без строки меню и других пользовательских элементов управления, могут служить в качестве диалоговых окон для более сложных сообщений и полей ввода.

JavaScript не определяет методы, позволяющие непосредственно создавать фреймы и манипулировать ими в окне броузера. Однако возможность динамической генерации HTML позволяет программно выводить HTML-теги, создающие любую схему расположения фреймов.

JavaScript также позволяет определять, какие веб-страницы отображаются в броузере. Объект Location позволяет загружать и отображать содержимое любого URL в любом окне или фрейме броузера. Объект History обеспечивает перемещение вперед и назад по истории посещения страниц, имитируя действия кнопок Forward и Back.

Еще один метод объекта Window позволяет JavaScript выводить произвольные сообщения в строку состояния любого окна броузера.

1.6.3. Взаимодействие с HTML-формами

Еще одним важным аспектом клиентского JavaScript является возможность его взаимодействия с HTML-формами. Эта возможность предоставляется объектом Form и объектами элементов формы, которые в ней содержатся:

Button, CheckBox, Hidden, Password, Radio, Reset, Select, Submit, Text и Textarea. Эти объекты элементов позволяют читать и писать значения в составляющие форму элементы ввода. Например, в онлайн-каталоге может присутствовать HTML-форма заполнения заказа, в которой средствами JavaScript осуществляется чтение входных данных формы при вычислении стоимости заказа, налога с продаж и стоимости доставки. Подобные программы JavaScript действительно очень распространены в Интернете. Скоро мы увидим программу, в которой сочетание HTML-формы и JavaScript обеспечивает пользователю возможность вычисления ежемесячных платежей по закладной дома или другой ссуде. В таких приложениях JavaScript имеет явное преимущество перед серверными сценариями. Код JavaScript выполняется на стороне клиента, поэтому содержимое формы не требуется отправлять на сервер для выполнения относительно простых вычислений.

Другое распространенное применение клиентского JavaScript в сочетании с формами – проверка данных формы перед их передачей на сервер. Если все необходимые проверки правильности пользовательского ввода могут быть выполнены клиентским JavaScript, нет необходимости в передаче данных на сервер для нахождения тривиальных ошибок и сообщения о них пользователю. Клиентский JavaScript может выполнять предварительную обработку входных данных, что сократит объем передаваемых на сервер данных. В некоторых случаях клиентский JavaScript может совсем исключить необходимость выполнения сценариев на сервере! (С другой стороны, JavaScript и серверные сценарии прекрасно работают вместе. Например, серверная программа так же, как она динамически создает HTML-код, может «на лету» создавать код JavaScript.)

1.6.4. Взаимодействие с пользователем

Важной чертой JavaScript является возможность определять обработчики событий – произвольные фрагменты кода, исполняемого при возникновении определенных событий. Обычно эти события вызываются пользователем, когда, например, он проводит мышью над гиперссылкой, вводит значение в форму или щелкает по кнопке Submit в форме. Это очень важный механизм, поскольку программирование графических интерфейсов, таких как HTML-формы, изначально требует событийной модели. JavaScript может выполнять любые виды действий в ответ на пользовательские события. Типичным примером может быть отображение специального сообщения в строке состояния, когда пользователь помещает курсор мыши на гиперссылку, или отображение подтверждающего диалогового окна при сохранении данных формы.

1.6.5. Чтение и запись клиентского состояния с помощью cookies

Файлы cookies – это небольшой объем данных состояния, постоянно или временно сохраняемый на стороне клиента. Файлы cookies могут передаваться вместе с веб-страницей сервером клиенту, который сохраняет их на локаль-

ной машине. Когда клиент затем запрашивает ту же или связанную с ней веб-страницу, он передает соответствующие cookies обратно на сервер, где их значения могут использоваться для изменения содержимого страницы, отсылаемой обратно клиенту. Cookies позволяют веб-странице или веб-сайту что-то вспомнить о клиенте – например, что пользователь ранее посещал сайт, зарегистрировался на нем, получил пароль или выразил свои предпочтения о цвете и схеме расположения веб-страниц. Cookies помогают вам предоставить информацию о состоянии, отсутствующую в протоколе HTTP, используемом во Всемирной паутине.

Изначально файлы cookies предназначались исключительно для серверных сценариев, – хотя они сохранялись на стороне клиента, читать их или записывать в них мог только сервер. JavaScript изменил это правило, так как программы JavaScript могут читать и записывать значения cookies и динамически генерировать содержимое документа, базируясь на этих значениях.

1.6.6. Другие возможности

Кроме средств, которые мы уже обсудили, JavaScript имеет много других возможностей, в том числе следующие:

- JavaScript позволяет управлять отображением графики, выводимой с помощью тега `<img>`, создавать анимационные эффекты и переключать изображения.
- JavaScript может взаимодействовать с Java-апплетами и другими встроенными объектами, присутствующими в броузере. Код JavaScript может читать и писать свойства этих апплетов и объектов, а также вызывать любые методы, определенные в них. Эта возможность действительно позволяет JavaScript придавать Java-апплетам свойства сценариев.
- Мы уже говорили, что JavaScript может выполнять произвольные вычисления. В JavaScript имеется тип данных с плавающей точкой, арифметические операторы для работы с ним и полный набор стандартных математических функций для чисел с плавающей точкой.
- Объект Date в JavaScript упрощает процесс вычислений и работы с датами и временем.
- Объект Document поддерживает свойство, содержащее дату последней модификации текущего документа, и с его помощью можно организовать автоматическое отображение временной метки любого документа.
- В JavaScript имеется метод `window.setTimeout()`, позволяющий любому блоку кода JavaScript исполняться на некоторое количество миллисекунд позднее. Это может быть полезно для создания в JavaScript-программе задержек или выполнения повторяющихся действий. В JavaScript 1.2 метод `setTimeout()` дополнен другим полезным методом – `setInterval()`.
- В объекте Navigator (естественно, названном в честь веб-броузера Netscape) имеются переменные, содержащие название и версию запущенного броузера, а также переменные, обозначающие платформу, на которой он

работает. Эти переменные позволяют сценариям настраивать свое поведение в зависимости от броузера или платформы, чтобы сценарий мог воспользоваться преимуществами дополнительных возможностей, поддерживаемых некоторыми версиями, или чтобы обойти ошибки, существующие на некоторых платформах.

- В клиентском JavaScript 1.2 объект Screen предоставляет информацию о размере и глубине цвета монитора, на котором отображается веб-броузер.
- Что касается JavaScript 1.1, метод `scroll()` объекта Window позволяет программам на JavaScript прокручивать окна в размерностях X и Y. В JavaScript 1.2 этот метод дополняется набором других методов, позволяющих перемещать и изменять окна броузера.

1.6.7. Что не умеет JavaScript

Клиентский JavaScript обладает впечатляющим списком возможностей. Заметьте, однако, что они ограничены задачами, связанными с броузерами и документами. Клиентский JavaScript применяется в ограниченном контексте и потому не имеет средств, обязательных для самостоятельных языков программирования:

- JavaScript не имеет каких-либо графических возможностей, за исключением мощного средства динамической генерации HTML-кода (в том числе изображений, таблиц, фреймов, форм, шрифтов и т. д.), отображаемого броузером.
- По соображениям безопасности клиентский JavaScript не допускает чтения или записи файлов. Очевидно, что мало кто захочет, чтобы программа с любого произвольного веб-сайта запустилась на его компьютере и манипулировала файлами по своему усмотрению!
- JavaScript не поддерживает сетевых функций, он лишь может заставить броузер загружать произвольные URL и посылать содержимое HTML-форм по сети серверным сценариям и по электронной почте.

1.7. Безопасность в JavaScript

Во всех тех случаях, когда программы (такие как сценарии JavaScript, программы Visual Basic или макросы Microsoft Word) включаются в совместно используемые документы, в особенности в документы, передаваемые по Интернету или по электронной почте, имеется потенциальная опасность проникновения вирусов и других злонамеренных программ. Разработчики JavaScript не упустили из виду вопросы безопасности и позаботились о том, чтобы не дать программам JavaScript способность выполнять вредные действия. Например, как было описано выше, программы клиентского JavaScript не могут читать локальные файлы или выполнять сетевые операции.

Однако по причине сложности среды веб-броузеров в их ранних версиях все-таки возникали некоторые проблемы безопасности. Например, в Netscape 2

было возможно написание кода JavaScript, который автоматически похищал адрес электронной почты любого посетителя страницы, содержащей этот код, и затем автоматически посылал email-сообщение от его имени, без уведомления или запроса подтверждения посетителя страницы. Эта и несколько других дыр в системе безопасности были исправлены. Хотя нет гарантии, что не будут найдены другие «дыры», большинство осведомленных пользователей спокойно разрешают современным броузерам запускать код JavaScript, находящийся на веб-страницах. В главе 21 вопросы безопасности в клиентском JavaScript обсуждаются подробно.

1.8. Пример: вычисление платежей по ссуде с помощью JavaScript

В примере 1.3 приведен листинг полноценной нетривиальной программы на языке JavaScript. Программа вычисляет месячный платеж по закладной дома или другой ссуде на основании размера ссуды, процентной ставки и периода выплаты. Как видите, программа состоит из HTML-формы, которая сделана интерактивной с помощью кода JavaScript. На рис. 1.3 показано, как HTML-форма выглядит при отображении в веб-броузере. Однако рисунок может только передать статический снимок программы. Добавление кода JavaScript делает ее динамической: как только пользователь изменяет сумму ссуды, процентную ставку или количество платежей, код JavaScript заново вычисляет месячный платеж, общую сумму платежей и общий процент, уплаченный за все время ссуды.

Первая половина примера – это форма HTML, аккуратно отформатированная с помощью HTML-таблицы. Заметьте, что для нескольких элементов формы определены обработчики событий `onchange` или `onclick`. Веб-броузер запускает эти обработчики в тот момент, когда пользователь изменяет входные данные или щелкает по кнопке `Compute`, отображаемой на форме. Заметьте, что во всех этих случаях значением атрибута обработчика события является строка кода JavaScript: `calculate()`. Вызываемый обработчик события исполняет этот код, приводящий к вызову функции `calculate()`.

Функция `calculate()` определена во второй половине примера, внутри тегов `<script>`. Функция читает введенные пользователем данные из формы, выполняет математические действия, требуемые для вычисления платежей по ссуде, и отображает результаты этих действий, используя три нижних элемента формы.

Пример 1.3 прост, но на его внимательное рассмотрение стоит потратить время. Вы не должны понимать весь код JavaScript сейчас, однако изучение этого примера должно дать вам хорошее представление о том, как выглядят программы на JavaScript, как функционируют обработчики событий и как код JavaScript может интегрироваться в HTML-формы. Заметьте, что комментарии в HTML-коде заключаются между тегами `<!--` и `-->`, а в коде JavaScript – в строки, начинающиеся с символов //.

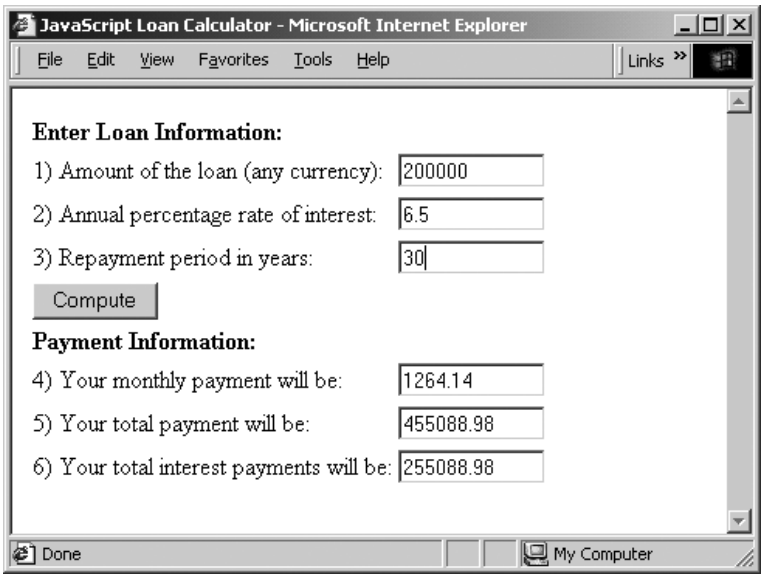

***Рис. 1.3.** Калькулятор платежей по ссуде на JavaScript*

Пример 1.3. Вычисление платежей по ссуде с помощью JavaScript

```
<head><title>JavaScript Loan Calculator</title></head>
<body bgcolor="white">
<!--
  Это HTML-форма, дающая пользователю возможность вводить данные и с
  помощью JavaScript показывать ему результат вычислений.
  Элементы формы для улучшения их внешнего вида помещены в таблицу.
  Сама форма имеет имя "loandata", а поля в форме – такие имена, как
  "interest" и "years". Эти имена полей используются в коде JavaScript,
  следующем за формой. Заметьте, что для некоторых элементов формы
  определены обработчики событий "onchange" или "onclick". В них заданы
  строки кода JavaScript, выполняемого при вводе данных или
  щелчке по кнопке.
-->
<form name="loandata">
  <table>
    <tr><td colspan="3"><b>Enter Loan Information:</b></td></tr>
    <tr>
      <td>1)</td>
      <td>Amount of the loan (any currency):</td>
      <td><input type="text" name="principal" size="12"
                 onchange="calculate();"></td>
    </tr>
    <tr>
      <td>2)</td>
```

```
      <td>Annual percentage rate of interest:</td>
      <td><input type="text" name="interest" size="12"
                 onchange="calculate();"></td>
    </tr>
    <tr>
      <td>3)</td>
      <td>Repayment period in years:</td>
      <td><input type="text" name="years" size="12"
                 onchange="calculate();"></td>
    </tr>
    <tr><td colspan="3">
      <input type="button" value="Compute" onclick="calculate();">
    </td></tr>
    <tr><td colspan="3">
      <b>Payment Information:</b>
    </td></tr>
    <tr>
      <td>4)</td>
      <td>Your monthly payment will be:</td>
      <td><input type="text" name="payment" size="12"></td>
    </tr>
    <tr>
      <td>5)</td>
      <td>Your total payment will be:</td>
      <td><input type="text" name="total" size="12"></td>
    </tr>
    <tr>
      <td>6)</td>
      <td>Your total interest payments will be:</td>
      <td><input type="text" name="totalinterest" size="12"></td>
    </tr>
  </table>
</form>

<!--
  Это программа на JavaScript, которая заставляет пример работать.
  Заметьте, что в этом сценарии определяется функция calculate(),
  вызываемая обработчиками событий в форме. Функция ссылается на значения
  в форме, используя имена, определенные в коде, который приведен выше.
-->
<script language="JavaScript">
function calculate() {
    // Получаем пользовательские данные из формы. Предполагаем, что
    // данные являются корректными. Преобразуем процентную ставку из
    // процентов в десятичное значение. Преобразуем период платежа в
    // годах в количество месячных платежей.
    var principal = document.loandata.principal.value;
    var interest = document.loandata.interest.value / 100 / 12;
    var payments = document.loandata.years.value * 12;

    // Теперь вычисляем сумму месячного платежа.
```

```
    var x = Math.pow(1 + interest, payments);
    var monthly = (principal*x*interest)/(x-1);

    // Проверяем, является ли результат конечным числом. Если да, то
    // отображаем результаты.
    if (!isNaN(monthly) &&
        (monthly != Number.POSITIVE_INFINITY) &&
        (monthly != Number.NEGATIVE_INFINITY)) {

        document.loandata.payment.value = round(monthly);
        document.loandata.total.value = round(monthly * payments);
        document.loandata.totalinterest.value =
            round((monthly * payments) - principal);
    }
    // В противном случае пользовательские данные, видимо, не корректны,
    // поэтому не показываем ничего.
    else {
        document.loandata.payment.value = "";
        document.loandata.total.value = "";
        document.loandata.totalinterest.value = "";
    }
}

// Этот простой метод округляет число до двух десятичных знаков.
function round(x) {
  return Math.round(x*100)/100;
}
</script>
</body>
</html>
```

1.9. Структура этой книги

Остальной материал данной книги представлен пятью частями. Часть I, следующая непосредственно за этой главой, содержит описание базового языка JavaScript. Главы 2–6 начинают этот раздел простым, но необходимым материалом – в них приведена базовая информация, необходимая при изучении нового языка программирования:

- Глава 2 «Лексическая структура» описывает базовую структуру языка.
- Глава 3 «Типы данных и значения» описывает типы данных, поддерживаемые JavaScript.
- Глава 4 «Переменные» содержит сведения о переменных, диапазоне их действия и охватывает другие связанные с этим темы.
- Глава 5 «Выражения и операторы» содержит описание выражений JavaScript и всех операторов, поддерживаемых JavaScript. За основу синтаксиса JavaScript взят язык Java, который, в свою очередь, создавался

на основе C и C++, поэтому программисты, имеющие опыт работы с C, C++ и Java могут лишь бегло просмотреть большую часть этой главы.

- Глава 6 «Инструкции» описывает синтаксис и применение каждой из инструкций JavaScript. Опять же, программисты на C, C++ и Java могут бегло просмотреть некоторые разделы, но не всю эту главу.

Следующие пять глав первой части более интересны. В них продолжено рассмотрение ядра языка JavaScript, но описываются те аспекты языка, с которыми не знакомы даже те, кто знает C или Java. Тому, кто действительно хочет понимать JavaScript, эти главы следует изучить тщательно.

- Глава 7 «Функции» описывает порядок определения и вызовов функций в JavaScript и других манипуляций с ними.
- Глава 8 «Объекты» содержит документацию по объектам, наиболее важному типу данных JavaScript. В этой главе обсуждается объектно-ориентированное программирование в JavaScript и объясняется, как в JavaScript можно определять собственные классы объектов.
- Глава 9 «Массивы» описывает создание массивов и работу с ними в JavaScript.
- Глава 10 «Регулярные выражения» описывает применение регулярных выражений в JavaScript для поиска по маске и операций поиска/замены.
- Глава 11 «Прочие вопросы программирования на JavaScript» рассматривает более сложные темы, не затронутые в других местах. При первом чтении эту главу можно пропустить, однако материал, содержащийся в ней, важен для понимания, если вы хотите стать экспертом по JavaScript.

В части II описан клиентский JavaScript. Ее главы документируют объекты веб-броузера, составляющие основу клиентского JavaScript, и содержат подробные примеры работы с ними. Любая представляющая интерес программа JavaScript, работающая в веб-броузере, в значительной степени полагается на возможности, специфичные для клиентской среды.

Вот что вы найдете в части II:

- Глава 12 «JavaScript в веб-броузерах» посвящена интеграции JavaScript в веб-броузеры. В ней обсуждается веб-броузер как программная среда и приведены различные способы интеграции JavaScript в веб-страницы для исполнения на стороне клиента.
- Глава 13 «Окна и фреймы» документирует наиболее существенный и важный объект клиентского JavaScript – объект Window, а также несколько важных объектов, связанных с окном.
- Глава 14 «Объект Document» описывает объект Document и связанные с ним объекты, предоставляющие содержимое HTML-документа для кода JavaScript.
- Глава 15 «Формы и элементы форм» описывает объект Form, представляющий HTML-формы. В ней документируются различные объекты эле-

ментов, присутствующие в HTML-формах, и приведены примеры программирования форм в JavaScript.

- Глава 16 «Сценарии и cookies» иллюстрирует использование файлов cookie в веб-программировании для сохранения состояния.
- Глава 17 «Объектная модель документа» содержит описание базовых частей стандарта W3C DOM и показывает, как программа на JavaScript может обращаться к любым элементам документа HTML.
- Глава 18 «Каскадные таблицы стилей и Dynamic HTML» содержит описание частей стандарта W3C DOM, позволяющих программе на JavaScript управлять стилем, внешним видом и местоположением элементов внутри HTML-документа. В этой главе показано, как создавать многочисленные эффекты DHTML с помощью свойств CSS.
- Глава 19 «События и обработчики событий» рассматривает события JavaScript и обработчики событий, очень важные для всех JavaScript-программ, взаимодействующих с пользователем. В этой главе описываются традиционная событийная модель, событийная модель стандарта W3C DOM и собственная событийная модель Internet Explorer.
- Глава 20 «Приемы обеспечения совместимости» исследует важный вопрос совместимости в программировании на JavaScript и содержит обсуждение приемов написания программ JavaScript, корректно работающих (или изящно сбоящих) на широком разнообразии веб-броузеров.
- Глава 21 «Безопасность в JavaScript» посвящена рассмотрению ограничений, встроенных в клиентский JavaScript из соображений безопасности, и обоснованию их необходимости.
- Глава 22 «Совместное применение Java и JavaScript» содержит обсуждение средств JavaScript, позволяющих организовать управление Java-апплетами. В ней также рассматривается обратная задача – вызов кода JavaScript из Java-апплетов.

Части III, IV и V представляют собой справочники, документирующие объекты, определяемые, соответственно, в базовом языке JavaScript, в традиционном клиентском JavaScript и в новом стандарте W3C DOM.

1.10. Изучение JavaScript

Реальное изучение нового языка программирования невозможно без написания программ. Рекомендую вам при чтении этой книги опробовать возможности JavaScript в процессе их изучения. Вот несколько приемов, призванных облегчить эти эксперименты.

Наиболее очевидный способ изучения JavaScript – это написание простых сценариев. Одно из достоинств клиентского JavaScript состоит в том, что любой, кто имеет веб-броузер и простейший текстовый редактор, уже имеет полноценную среду разработки. Для того чтобы начать писать программы

на JavaScript, нет необходимости в покупке или загрузке специального программного обеспечения. В начале этой главе вы видели пример с вычислением факториалов. Предположим, вы хотите изменить его так, чтобы он выводил вместо факториалов числа Фибоначчи:

```
<script>
document.write("<h2>Таблица чисел Фибоначчи </h2>");
for (i=0, j=1, k=0, fib =0; i<50; i++, fib=j+k, j=k, k=fib){
    document.write("Фибоначчи ("  + i +  ") = " + fib);
    document.write("<br>");
}
</script>
```

Этот код может показаться запутанным (и не волнуйтесь, если вы пока не понимаете его), но для того чтобы поэкспериментировать с подобными короткими программами, достаточно набрать код и запустить в веб-броузере с помощью локального URL `file:`. Заметьте, что для вывода результатов вычислений применяется метод `document.write()`. Это полезный прием при экспериментах с JavaScript. В качестве альтернативы для отображения текстового результата в диалоговом окне можно использовать метод `alert()`:

```
alert("Fibonacci ("  + i +  ") = " + fib);
```

Обратите внимание, что в подобных простых экспериментах с JavaScript можно опускать теги `<html>`, `<head>` и `<body>` в HTML-файле.

Для еще большего упрощения экспериментов с JavaScript можно использовать URL с псевдопротоколом `javascript:` для вычисления значения выражения JavaScript и получения результата. URL JavaScript состоит из признака протокола `javascript:`, за которым следует произвольный код JavaScript (инструкции отделены одна от другой точками с запятой). Загружая такой URL, броузер исполняет код JavaScript. Значение последнего выражения в таком URL преобразуется в строку, и эта строка выводится веб-броузером в качестве нового документа. Например, для того чтобы проверить свое понимание некоторых операторов и инструкций JavaScript, можно набрать следующие URL JavaScript в поле Location веб-броузера:

```
javascript:5%2
javascript:x = 3; (x < 5)? "x is less": "x is greater"
javascript:d = new Date(); typeof d;
javascript:for(i=0,j=1,k=0,fib=1; i<10; i++,fib=j+k,k=j,j=fib) alert(fib);
javascript:s=""; for(i in document) s+=i+":"+document[i]+"\n"; alert(s);
```

Не любой ваш код, написанный при изучении JavaScript, будет работать так, как вы ожидаете, и вам захочется его отладить. Базовая методика отладки для JavaScript совпадает с методикой для многих других языков: вставка в код инструкций, которые будут выводить значения нужных переменных так, чтобы можно было понять, что же на самом деле происходит.

Как мы уже видели, иногда для этого можно применить метод `document.write()`. Однако этот метод не годится для обработчиков событий и имеет другие недостатки, так что часто для вывода отладочных сообщений в отдельном диалоговом окне бывает проще использовать функцию `alert()`.

Цикл `for/in` (описанный в главе 6) также полезен в отладке. Например, его можно применять вместе с методом `alert()` для написания функции, отображающей имена и значения всех свойств объекта. Такая функция может быть удобна при изучении языка или при отладке кода.

Удачи вам с JavaScript и счастливой работы!

Часть I. Базовый JavaScript

Данная часть книги включает главы со 2 по 11 и документирует базовый язык JavaScript в том виде, в каком он используется в веб-броузерах, веб-серверах и других встроенных реализациях JavaScript. Этот материал задуман как справочный, и, прочитав главы этой части один раз, вы, возможно, будете неоднократно возвращаться к ним, чтобы освежить в памяти некоторые особенности языка.

- Глава 2 «Лексическая структура»
- Глава 3 «Типы данных и значения»
- Глава 4 «Переменные»
- Глава 5 «Выражения и операторы»
- Глава 6 «Инструкции»
- Глава 7 «Функции»
- Глава 8 «Объекты»
- Глава 9 «Массивы»
- Глава 10 «Регулярные выражения»
- Глава 11 «Прочие вопросы программирования на JavaScript»

2

Лексическая структура

Лексическая структура языка программирования – это набор элементарных правил, определяющих, как пишутся программы на этом языке. Это низкоуровневый синтаксис языка; он задает вид имен переменных, символы, используемые для комментариев, и как одна инструкция отделяется от другой. Эта короткая глава документирует лексическую структуру JavaScript.

2.1. Набор символов

При написании программ на JavaScript используется набор символов Unicode. В отличие от 7-битовой кодировки ASCII, подходящей только для английского языка, и 8-битовой кодировки ISO Latin-1, подходящей только для английского и основных западноевропейских языков, 16-битовая кодировка Unicode обеспечивает представление практически любого письменного языка. Эта возможность важна для интернационализации и особенно важна для программистов, не говорящих на английском языке.

Американские и другие англоговорящие программисты обычно пишут программы с помощью текстового редактора, поддерживающего только кодировки ASCII или Latin-1, и потому не имеют легкого доступа к полноценному набору символов Unicode. Однако никаких трудностей это не порождает, поскольку кодировки ASCII и Latin-1 представляют собой подмножества Unicode, и любая программа JavaScript, написанная с помощью этих наборов символов, абсолютно корректна. Программисты, привыкшие рассматривать символы как 8-разрядные значения, могут быть сбиты с толку, узнав, что JavaScript представляет каждый символ с помощью двух байтов, однако на самом деле для программиста это обстоятельство остается незаметным и может просто игнорироваться.

Стандарт ECMAScript v3 допускает использование символов Unicode в любом месте программы на JavaScript. Однако версии 1 и 2 стандарта допускают применение символов Unicode только в комментариях и строковых литералах, заключенных в кавычки, а все остальные составляющие программы ECMAScript v1 ограничены набором символов ASCII. Версии JavaScript, предшествующие стандарту ECMAScript, обычно вообще не поддерживают Unicode.

2.2. Чувствительность к регистру

JavaScript – это язык, чувствительный к регистру. Это значит, что ключевые слова, переменные, имена функций и любые другие идентификаторы языка должны всегда содержать одинаковые наборы заглавных и строчных букв. Например, ключевое слово `while` должно набираться как while, а не While или WHILE. Аналогично, `online`, `Online`, `OnLine` и `ONLINE` – это имена четырех различных переменных.

Заметим, однако, что HTML не чувствителен к регистру. По причине близкой связи HTML и клиентского JavaScript это различие может привести к путанице. Многие объекты JavaScript и их свойства имеют те же имена, что и теги и атрибуты HTML, которые они обозначают. Если в HTML эти теги и атрибуты могут набираться в любом регистре, то в JavaScript они обычно должны набираться строчными буквами. Например, атрибут обработчика события `onclick` чаще всего задается в HTML как `onClick`, однако в коде JavaScript он должен быть обозначен как `onclick`.

Если базовый JavaScript полностью чувствителен к регистру, то в клиентском JavaScript допускаются исключения из этого правила. Например, в Internet Explorer 3 все клиентские объекты и свойства были не чувствительны к регистру. Однако это приводило к нарушениям совместимости с Netscape, поэтому в Internet Explorer 4 и более поздних версиях клиентские объекты и их свойства стали чувствительными к регистру.

2.3. Символы-разделители и переводы строк

JavaScript игнорирует пробелы, табуляции и переводы строк, присутствующие между лексемами в программе, кроме тех, которые являются частью строковых литералов и регулярных выражений. *Лексема* – это ключевое слово, имя переменной, число, имя функции или какая-либо другая сущность, внутри которых не допускается присутствие пробела или перевода строки. Если в лексему поместить пробел, табуляцию или перевод строки, то получатся две лексемы. Например, `123` – это одна числовая лексема, а `12 3` – это две отдельных лексемы (и, между прочим, такая запись синтаксически ошибочна).

Поскольку допускается наличие в программе пробелов, знаков табуляции и переводов строк (кроме строковых литералов, регулярных выражений и лексем), программист может отформатировать текст четким и непротиворе-

чивым образом, облегчающим чтение и понимание кода. Заметьте, однако, что есть одно небольшое ограничение в расположении переводов строк; оно описано в следующем разделе.

2.4. Необязательные точки с запятой

Простые инструкции JavaScript обычно завершаются точками с запятой (;), как в C, C++ и Java. Точка с запятой служит для отделения инструкций друг от друга. Однако в JavaScript точку с запятой можно не ставить, если каждая инструкция помещается на отдельной строке. Например, следующий код может быть записан без точек с запятой:

```
a = 3;
b = 4;
```

Однако если обе инструкции расположены в одной строке, то первая точка с запятой должна присутствовать обязательно:

```
a = 3; b = 4;
```

Пропуск точек с запятой нельзя признать правильной практикой программирования, и поэтому надо выработать привычку их использовать.

Теоретически JavaScript допускает переводы строк между любыми двумя лексемами, но привычка синтаксического анализатора JavaScript автоматически вставлять точки с запятой за программиста приводит к некоторым исключениям из этого правила. Если в результате разделения строки кода та ее часть, которая предшествует символу перевода, оказывается законченной инструкцией, JavaScript может подумать, что точка с запятой пропущена случайно, и вставить ее, изменив смысл программы. К подобным ситуациям, требующим внимания,– относятся, среди прочих, инструкции `return`, `break` и `continue` (описанные в главе 6). Рассмотрим, например, следующий код:

```
return
true;
```

JavaScript предполагает, что программист имеет в виду следующее:

```
return;
true;
```

Однако он, видимо, хотел написать

```
return true;
```

Вот случай, когда следует быть внимательным, – данный код не вызовет синтаксической ошибки, но приведет к неочевидному сбою. Похожая неприятность возникает, если написать:

```
break
outerloop;
```