**Виктор Вахтуров**

# JavaScript

## ОСВОЙ НА ПРИМЕРАХ

Санкт-Петербург
«БХВ-Петербург»
2007

УДК 681.3.06
ББК 32.973.26-018.1
В22

**Вахтуров В. В.**

В22 JavaScript. Освой на примерах. — СПб.: БХВ-Петербург, 2007. — 400 с.: ил. + CD-ROM

ISBN 978-5-94157-877-1

На наглядных примерах даны практические приемы программирования клиентских сценариев для Web-браузеров. Кратко изложены основы создания Web-страниц и скриптов: язык JavaScript, каскадные таблицы стилей (CSS) и объектная модель документа (DOM). Рассмотрено решение типовых задач программирования скриптов: работа с датой и временем, cookies, регулярными выражениями и протоколами. Даны примеры создания динамических эффектов: управление окном браузера, разработка динамических форм, средства и способы работы с изображениями, анимационные эффекты, реализация перетаскивания (Drag and Drop), эмуляция элементов управления пользовательского интерфейса. Рассмотрено написание функционально законченных приложений: реализация визуального редактора HTML и нескольких известных игр на JavaScript. Исходные тексты всех примеров находятся на прилагаемом компакт-диске.

*Для широкого круга Web-программистов*

УДК 681.3.06
ББК 32.973.26-018.1

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Евгений Рыбаков* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Анна Кузьмина* |
| Компьютерная верстка | *Ольги Сергиенко* |
| Корректор | *Зинаида Дмитриева* |
| Дизайн обложки | *Игоря Цырульникова* |
| Оформление обложки | *Елены Беляевой* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.03.07.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 25.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-94157-877-1

# Оглавление

# Введение

Давным-давно, когда я делал свой первый сайт, только начиная постигать азы Web-разработки, передо мной встала проблема изучения JavaScript. Сайт казался мне простым и неинтересным. Я хотел сделать его более динамичным и красочным, добавив на страницы несколько интерактивных элементов. Я знал, что язык JavaScript решит мои проблемы, но ничего не знал о его применении. Когда я прочел соответствующую документацию, всесторонне описывающую структуру и синтаксис языка, я осознал, что так и не понимаю, как конкретно решать поставленные передо мной задачи. Тогда я принялся искать в Интернете готовые примеры скриптов на JavaScript и изучать их.

Уже через пару недель я с улыбкой вспоминал о проблемах, ранее казавшихся мне неразрешимыми. Изучение готовых примеров дало мне самое главное — понимание принципов разработки скриптов и то, в каком направлении стоит двигаться для совершенствования своих навыков.

А сегодня, по прошествии нескольких лет, имея за плечами большой опыт Web-разработки с применением самых различных Web-технологий, я рад представить вам свою книгу, рассказывающую о программировании клиентских сценариев на JavaScript. Помня о трудностях, возникших передо мной в свое время, я решил изложить материал самым доступным способом — на примерах.

## О чем и для кого эта книга

Книга, которую вы сейчас держите в руках, рассказывает о принципах, приемах и тонких аспектах разработки клиентских сценариев (скриптов) для Web-браузеров на самом популярном в настоящий момент языке

создания сценариев JavaScript. Применение клиентских скриптов (фрагментов программного кода, загружаемых вместе с Web-страницей в браузер), позволяет создавать на страницах Web-сайта красочные и интересные эффекты, оригинальные элементы оформления, управления и навигации, контролировать действия пользователя, изменять страницу без ее перезагрузки и многое другое, что недоступно при помощи только HTML и CSS.

Основу материала книги составляют практические примеры, сложность которых варьируется от минимальной до достаточно большой, представляющие собой законченные и готовые к применению на сайте скрипты различного назначения. Однако каждый раздел, посвященный определенной теме, содержит теоретический материал, освещающий соответствующие вопросы.

Книга предназначена для широкого круга читателей — как для тех, кто абсолютно не знаком с какими бы то ни было принципами создания Web-страниц и скриптов, так и тех, кто достаточно хорошо владеет знаниями в этой области. Благодаря наличию вводных глав, посвященных изучению основ — структуры и синтаксиса JavaScript, HTML, CSS, DOM, книгу можно читать независимо от начального уровня квалификации. Иными словами, если вы желаете изучить JavaScript с нуля, совершенствуете и развиваете свои навыки, или просто ищете для себя что-то новое в данной области — эта книга для Вас!

# Как организована книга, и как ее читать

Структурно книга состоит из четырех частей, включающих главы, объединяемые общими идеями и концепциями. В *части I* описываются средства создания статических и динамических Web-страниц — язык HTML, каскадные таблицы стилей, объектная модель документа браузера, синтаксис языка JavaScript. К ней относятся *главы 1—4*. В *части II* обсуждаются решения типовых задач, очень часто встречающихся при программировании скриптов для Web-браузеров — получение параметров операционного окружения, работа с cookie, объектом даты/времени, регулярными выражениями и протоколами. В эту часть входят *главы 5—9*. *Часть III*, включающая *главы 10—13*, посвящена рассмотрению приемов программирования и примеров реализации функционально законченных скриптов, готовых для практического применения. В ней рассказывается о работе с окном браузера, формами, изображениями и создании анимации на Web-страницах. *Часть IV* содержит *главы 14* и *15*, в которых обсуждаются более сложные вопросы, чем в предыду-

щих — реализация визуального редактора HTML и приемы эффективного объектно-ориентированного программирования на JavaScript. Далее приводятся более подробные сведения о содержании каждой из глав.

- В *главе 1* приводится описание языка JavaScript — синтаксиса, структур управления, типов данных, методов определения и манипулирования данными, объектно-ориентированного подхода, реализованного в данном языке.
- *Глава 2* содержит сведения о языке гипертекстовой разметки HTML и принципах его применения при создании Web-страниц.
- *Глава 3* описывает синтаксис и применение каскадных таблиц стилей второго уровня, а также модель представления документа в виде блоков, модель визуального форматирования.
- *Глава 4* посвящена изучению объектных моделей браузеров, объектной модели документа (DOM) и способам их использования.
- В *главе 5* рассматриваются способы получения параметров среды, в которой работает сценарий (версия JavaScript, различные параметры браузера, параметры экрана).
- *Глава 6* описывает приемы работы с датой и временем в JavaScript. В ней содержатся примеры оперирования объектом `Date`, вывода времени в различных форматах, вывода текущего времени на Web-странице, разрабатывается объект `CVDate`, имеющий расширенные возможности форматирования даты/времени и скрипт редиректа с заданной задержкой и выводом времени до перехода.
- *Глава 7* подробно освещает аспекты применения и работы с cookie (идентификация пользователя, формирование содержимого страницы в зависимости от установленных cookie и т. д.). В этой главе разрабатывается библиотека, упрощающая работу с cookie в JavaScript.
- *Глава 8* содержит теоретический материал и практические примеры применения регулярных выражений. Здесь разрабатывается библиотека функций для проверки корректности различных данных (идентификаторы, имена пользователей, дата/время, адреса e-mail и т. д.) с помощью регулярных выражений.
- В *главе 9* рассматриваются возможности использования протоколов `mailto` (отправка писем с заданным текстом, например, введенным на Web-странице, через почтовую систему пользователя) и `javascript` (генерация содержимого документов во фреймах, popup-окнах, а также изображений) при программировании сценариев.

- В *главе 10* обсуждаются средства и реализуются скрипты для работы с окнами браузера (создание, изменение размеров, положения окна), их частями (заголовок, строка состояния) и прокруткой документа.
- *Глава 11* посвящена работе с формами и элементами управления форм. Рассматриваются различные примеры создания динамических форм. Большое внимание уделяется контролю корректности данных и способам организации дружественного интерфейса.
- В *главе 12* все внимание сосредоточено на работе с изображениями — изображения в объектной модели, организация предварительной загрузки изображений, обработка ошибок загрузки и т. д. Здесь же разрабатываются скрипт слайд-шоу с предварительной загрузкой изображений и скрипт фотогалереи.
- *Глава 13* рассказывает о принципах создания анимации на Web-страницах. В рамках этой главы разрабатываются двадцать скриптов, создающих на странице яркие анимационные эффекты — бегущие строки, вертикальные скроллеры, движение текста по различным траекториям, эффекты летящих звезд, падающего снега, аналоговые часы, а также несколько очень интересных эффектов анимации на основе использования фильтров в Microsoft Internet Explorer.
- *Глава 14* полностью посвящена разработке одного скрипта — визуального редактора HTML, использующего режим редактирования документа Microsoft Internet Explorer. В ней также описываются интереснейшие специфические возможности этого браузера.
- В *главе 15* обсуждаются тонкие аспекты, неочевидные моменты и некоторые трудности ведения объектно-ориентированной разработки на JavaScript, изучаются приемы повышения эффективности объектно-ориентированного программирования на данном языке, а также создается библиотека классов, реализующая расширенный алгоритм наследования, поддерживающая идентификацию типа пользовательских объектов и синхронную обработку событий на основе механизма слотов и сигналов, а также включающая классы для работы с данными геометрических типов и класс таймера.

Порядок чтения книги может быть различным и зависеть от уровня вашей подготовки относительно излагаемых в ней вопросов. Если вы не имеете либо имеете слабые знания в области JavaScript, HTML, CSS и DOM, то можете читать книгу последовательно, с первой до последней главы. Однако для вас может оказаться удобным изучить сначала *главы 2* ("Язык HTML") и *3* ("Каскадные таблицы стилей CSS2"), затем перейти к *главе 1*, а потом читать книгу последовательно, начиная с *гла-*

*вы 4*. Если вы чувствуете себя уверенно в перечисленных вопросах, то можете начать чтение с *главы 5*. Поскольку в *главе 1* излагаются принципы объектно-ориентированного программирования на JavaScript, а *глава 15* посвящена обсуждению продвинутых методов эффективного объектно-ориентированного программирования, *главу 15* можно прочесть сразу после *главы 1*.

# Что содержится на компакт-диске

Компакт-диск, прилагаемый к книге, содержит:

- файлы примеров к *главам 1—15*. Примеры к конкретной главе находятся в подкаталоге с номером главы в каталоге examples. В подкаталоге examples\Lib содержатся файлы библиотек, разработка которых описана в книге: библиотека для работы с cookie, библиотека для работы с регулярными выражениями, библиотека для работы с датой/временем, библиотека для динамической генерации изображений. В каталоге examples\Lib\jsvcl содержится объектно-ориентированная библиотека JSVCL, разработанная в рамках *главы 15*;
- документацию по: языку гипертекстовой разметки HTML версий 4.0 и 4.01, каскадным таблицам стилей первого и второго уровней (CSS1 и CSS2), объектным моделям документа первого, второго и третьего уровней (DOM1, DOM2, DOM3). Документация располагается в каталоге w3c в соответствующих подкаталогах (html, css, dom);
- файлы, содержащие дополнительную справочную информацию к различным главам книги: таблицу предопределенных имен цветов CSS, таблицу соответствия имен свойств CSS свойствам объекта `style` в JavaScript, таблицу идентификаторов форм курсоров в CSS, таблицу соответствия элементов HTML интерфейсам объектов DOM, а также листинг регулярного выражения, описывающего адрес электронной почты в формате, определяемом RFC 822. Эти файлы располагаются в каталоге addons.

Более полное описание содержимого и структуры компакт-диска приведено в *приложении* к книге.

# Благодарности

В заключение я хочу выразить благодарность людям, чье участие значительным образом сказалось на судьбе настоящей книги.

В первую очередь, я благодарю заместителя главного редактора издательства "БХВ-Петербург" Рыбакова Евгения Евгеньевича за предложение написать книгу по данной тематике. Его консультации и замечания относительно структуры, содержания и оформления книги оказались поистине бесценными для меня, начинающего автора, впервые участвующего в подобном проекте. Он точно и ясно отвечал на все вопросы, которые, первое время, возникали у меня очень часто, и прощал многочисленные задержки сдачи рукописи в редакцию.

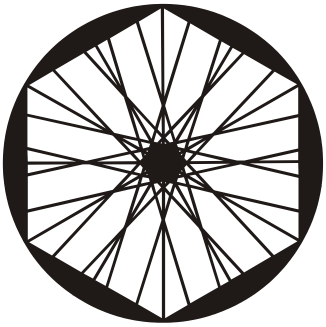
Я также очень благодарен редактору издательства "БХВ-Петербург" Кузьминой Анне Сергеевне за огромную работу, проделанную ей по редактированию данной книги. Она исправила кучу опечаток, ошибок оформления и стилистических неточностей, допущенных мной в процессе создания материала. Именно ее упорный труд придал книге ее настоящий вид.

Особую благодарность я выражаю Александру Пирамидину за предоставленную возможность размещения его переводов спецификаций HTML 4.01 и CSS 2 на прилагаемом к книге компакт-диске. Благодаря ему, читатели смогут иметь в своем распоряжении русские версии этих спецификаций.

**Примечание**

Некоторые термины, употребляемые мной в *главе 2 "Язык HTML"* и *главе 3 "Каскадные таблицы стилей CSS2"* книги, отличаются от соответствующих терминов, используемых в указанных переводах спецификаций. Это, видимо, объясняется моей приверженностью к несколько другому диалекту специальной терминологии, относящейся к области Web-разработки. В переводах Александр Пирамидин использует ряд терминов, близких по звучанию к их английским вариантам. Я же обозначаю соответствующие понятия их русскими аналогами (например, вместо термина "бокс", употребляемого в переводе спецификации CSS 2, я использую слово "блок" в книге). Поскольку смысл терминов, используемых в переводах и в книге, воспринимается идентично и однозначно, я думаю, у читателей не возникнет каких-либо проблем с их сопоставлением.

Наконец, я хочу сказать о человеке, чье участие в судьбе данной книги имеет исключительное значение лично для меня, ее автора — моем маленьком сыне Даре. В череде жизненных неурядиц, так некстати свалившихся на мою голову в период написания материала, иногда только он был причиной, побуждавшей меня с новыми силами браться за работу. Эту книгу я посвящаю ему.



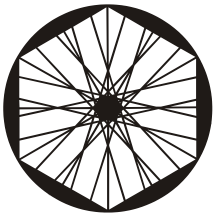
# ЧАСТЬ I

# Основы создания Web-страниц и скриптов

Мы начинаем путешествие в увлекательный и красочный мир динамического HTML — мир ярких эффектов, создаваемых на Web-страницах клиентскими сценариями (скриптами) — фрагментами внедряемого или подключаемого извне программного кода. Наши скрипты будут написаны на JavaScript — самом популярном на сегодняшний день языке программирования сценариев. При помощи этого языка на Web-страницах можно создавать множество интереснейших эффектов. Мы будем уделять основное внимание разработке скриптов, имеющих практическое применение: работа с формами, изображениями, регулярными выражениями, временем, cookie, реализация редактора HTML. Разработанные скрипты вполне готовы к применению на страницах реально действующих Web-сайтов.

На данный момент практически каждая программа-браузер имеет поддержку JavaScript. Однако, к сожалению, для конкретных моделей браузеров существует ряд индивидуальных особенностей, которые необходимо учитывать при разработке скриптов. Данная книга ориентируется на кроссбраузерные решения, работающие одинаково под управлением различных обозревателей. Замечания, касающиеся аспектов реализации для конкретных платформ, будут приводиться по мере необходимости.

Прежде чем перейти к углубленному изучению методов и приемов разработки сложных интерактивных элементов, необходимо разобраться с базовыми принципами создания динамических Web-страниц, играющими важную роль в понимании материала следующих разделов книги. В данной части приводятся основные сведения о языке JavaScript (его структуре и синтаксисе), языке гипертекстовой разметки (HTML), каскадных таблицах стилей (CSS) и объектной модели документа браузеров (DOM).



# ГЛАВА 1

# Структура и синтаксис языка JavaScript

Существует множество языков программирования — низкоуровневые (Assembler), процедурно-ориентированные (такие как C, Pascal), объектно-ориентированные (C++, Python), языки, реализующие логическую парадигму программирования (Prolog), и т. д. JavaScript является высокоуровневым объектно-ориентированным языком. Пожалуй, в качестве главной его особенности можно назвать межплатформенность, предполагающую возможность внедрения поддержки языка в различные программные продукты. JavaScript не предназначен для создания самостоятельных приложений, однако это не означает скудность его языковых средств. Напротив, в нем есть все, что необходимо для эффективного решения широкого круга алгоритмических задач.

В этой главе приводится достаточно полное описание языка — его структуры, синтаксиса, основных языковых конструкций.

## Определения и термины

Изучение любой области знаний обычно начинается с определения терминов и понятий, используемых для точного описания объектов этой области. Языки программирования не являются исключением, поэтому, прежде чем начать знакомство с основами JavaScript, следует ввести несколько специальных терминов, которые достаточно часто будут использоваться в нескольких следующих разделах книги. Приведенная далее терминология применима ко всем языкам программирования, однако, поясняющие примеры корректны именно в отношении JavaScript.

*Символ языка.* Под символами языка будем понимать множество возможных неделимых лексических единиц (лексем), корректных с точки зрения грамматики конкретного языка и составляющих его лексику. Символами языка являются: допустимые имена идентификаторов, ключевые слова, операторы, разделительные знаки, т. е. все объекты исходного текста программы, имеющие самостоятельное значение.

**Примечание**

Если выражаться формально, то под множеством символов языка здесь подразумевается множество терминальных символов грамматики языка.

Примеры:

- `5`, `10.843`, `2e+6`, `"Строка символов"` — литералы;
- `function`, `var` — ключевые слова;
- `oDiv`, `strTemp` — идентификаторы.

*Ключевое слово.* Это символ языка, являющийся частью его синтаксиса. Некоторые ключевые слова могут использоваться в качестве *литералов*: `true`, `false`, `null`. Ключевые слова не могут быть использованы в качестве *идентификаторов*.

*Зарезервированное слово.* Под зарезервированными словами обычно понимают лексемы, корректные с точки зрения грамматики языка, но не являющиеся частью его синтаксиса. Зарезервированные слова не могут употребляться в качестве идентификаторов, поскольку планируется их введение в состав языка в будущем.

*Литерал.* Литерал — это элемент данных, содержащийся непосредственно в программном коде. Литерал представляется некоторой последовательностью символов (литер), имеющей самостоятельное значение. Можно сказать, что литерал — это фиксированное значение (определенное литерально), содержащееся в коде программы. Литералами могут являться: числовые, строковые константы, инициализаторы массивов, объектов. Литералами не являются: имена переменных, констант и т. д.

Понятие литерала тесно связано с понятием типа данных. Применительно к JavaScript можно выделить следующие типы литералов:

- булевы (например, `true`, `false`);
- целочисленные (например, `10`, `109`);
- действительные, или с плавающей точкой (`1.15`, `2.64E10`);

- строковые (например: `"ABCD"`, `'aBcD'`);
- литералы массивов (`[10, 20, 40]` или `['one', 1, "two", 2]`);
- литералы объектов (`{ a: 10, b: 20 }`).

Некоторые литералы являются ключевыми словами: `true`, `false`, `null`, `undefined`.

Подробнее методы определения литералов различных типов будут рассмотрены в соответствующих разделах.

*Идентификатор.* Идентификатор представляет собой строку символов, обозначающую в программном коде некоторый объект. Идентификаторами являются имена переменных, объектов, функций, констант и т. д.

Пример:

```
a = 10; // здесь a - идентификатор.
```

*Оператор.* Применительно к языкам программирования, термин "оператор" в русском языке имеет двойственное значение (в английском языке существуют два разных термина, которые обычно переводятся на русский одинаково).

- Оператор *(statement)* — предложение языка, определяющее некое законченное действие в виде последовательности операций, задающей алгоритм решения задачи (исполняемые операторы), либо задающее набор описаний, необходимых для трансляции программного кода (невыполняемые операторы). Исполняемые операторы вызывают изменения состояния среды исполнения, служат для управления потоком вычислений, как, например, операторы цикла, условия, присваивания, перехода. С этой точки зрения программа представляется просто списком операторов. Следует заметить, что под данное определение гораздо больше подходят такие термины, как "утверждение", "инструкция". Но существует много устоявшихся выражений, включающих понятие "оператор" в приведенном контексте (например, "операторный блок"), поэтому в данной книге термин "оператор" часто используется как синоним слова "инструкция".
- Оператор *(operator)* — это символ языка, определяющий операцию обработки данных (например, сложение, умножение, сдвиг).

*Выражение.* Это синтаксическая конструкция, представляющая собой последовательность идентификаторов и/или литералов, комбинируемых при помощи операторов (имеет обычно вид формулы), которая может быть вычислена в определенное значение.

Ключевые и зарезервированные слова, переменные, типы данных, операторы, структуры управления, функции, типы данных, их применение и использование будут подробно рассмотрены далее в этой главе.

# Структура языка

Теперь, когда мы определились с терминологией, можно приступить непосредственно к изучению JavaScript. Изучение любого языка программирования логично начинать с рассмотрения его структуры и основных характеристик, таких как: правила записи кода (различных предложений языка, комментариев, операторных блоков и т. д.), порядка интерпретации/компиляции кода, перечня и правил использования разделительных и пробельных символов. Мы начнем именно с этого, учитывая специфику использования JavaScript в исполняющей среде браузера.

## JavaScript и HTML

Как уже говорилось, в этой книге мы будем рассматривать исключительно программирование клиентских сценариев для Web-браузеров, т. е. наши скрипты будут оперировать различными объектами Web-страниц, загружаемых в программу-обозреватель. Для того чтобы браузер был в состоянии исполнить код скрипта, этот код необходимо внедрить в Web-страницу. Для более полного восприятия информации данного раздела необходимо понимание основ языка гипертекстовой разметки HTML *(описание HTML см. в главе 2)*.

### Способы внедрения JavaScript-кода в документы HTML

Программный код на языке JavaScript может быть внедрен в HTML-документ несколькими способами:

- непосредственно включаться в HTML-документ как содержимое элемента `SCRIPT`;
- подключаться из внешнего источника, указанного в атрибуте `SRC` элемента `SCRIPT`;
- содержаться в виде кода обработчика события элемента HTML, задаваемого такими атрибутами, как, например, `onclick`, `onmousemove`;
- указываться в виде псевдо-URL с протоколом доступа `javascript:` как значение атрибутов `href`, `action`, `src` и т. д.

Как видно, в первых двух случаях должен использоваться элемент SCRIPT, поэтому рассмотрим его подробнее. Для этого элемента обязательны начальный и конечный теги (<SCRIPT> и </SCRIPT>). Также для него допустимы следующие атрибуты:

- src — этот атрибут задает URI внешнего скрипта. Если атрибут не определен, то все содержимое элемента SCRIPT интерпретируется как программный код. Если атрибут определен, то содержимое элемента полностью игнорируется, а пользовательский агент пытается загрузить и исполнить содержимое, URI которого задается этим параметром;
- type — этот атрибут задает язык скрипта элемента SCRIPT. Учитывая, что браузеры могут поддерживать различные языки клиентских сценариев (например, JavaScript, VBScript, TCL), необходимо либо указывать язык сценариев по умолчанию для документа, либо явно определять его для каждого элемента SCRIPT путем задания значения данному атрибуту. Язык скрипта указывается аналогично определению MIME-типа содержимого (например, "text/javascript", "text/vbscript", "text/tcl"). Значение данного атрибута имеет приоритет над языком скрипта, заданным по умолчанию. Во избежание возможных ошибок неверной интерпретации скриптов пользовательскими агентами, этот атрибут желательно указывать для каждого элемента SCRIPT, однако практика показывает, что все популярные программы-обозреватели на данный момент при отсутствии информации о языке скрипта пытаются интерпретировать код, исходя из предположения, что он написан на JavaScript;
- language — данный атрибут, также как и type, определяет язык скрипта элемента. Значением атрибута является идентификатор языка (например, "javascript" или "VBS"). Поскольку идентификаторы языков не стандартизированы, этот атрибут помечен в спецификации HTML 4.0 как нежелательный, поэтому вместо него следует использовать атрибут type;
- defer — это логический атрибут, его определение в теге <SCRIPT> обеспечивает возможность отложенного исполнения скрипта, сообщая браузеру о том, что данный сценарий не производит генерирования содержимого Web-страницы. Если этот атрибут не установлен, то отображение страницы пользовательским агентом прерывается до момента завершения исполнения всех инструкций, определяемых в элементе SCRIPT.

Теперь рассмотрим указанные выше способы включения кода скрипта в HTML-документ более подробно.

### Включение кода в элемент *SCRIPT*

Должно быть, это наиболее часто встречающийся способ добавления скриптов в Web-страницы. Последовательность инструкций просто записывается между парой тегов <SCRIPT> и </SCRIPT>, корректно размещенных в HTML-документе:

```
<HTML><BODY>
...
<SCRIPT type="text/javascript">
<!--
    var x = 10;
    alert('JavaScript на примерах');
// -->
</SCRIPT>
...
</BODY></HTML>
```

Браузер интерпретирует полностью содержимое такого блока, как программный код, написанный на языке, указанном в атрибуте type. Как можно заметить, в приведенном примере код заключен в HTML-комментарий (<!-- и -->). Это стандартный прием для обеспечения совместимости со старыми версиями браузеров. Такой подход позволяет избежать вывода на страницу кода скриптов, если обозреватель не поддерживает их исполнение. В действительности, браузеров, не "знающих" о существовании тега <SCRIPT>, сейчас просто не найти. Тем не менее этот прием широко используется и в настоящее время.

### Включение кода из внешнего источника

Этот способ подразумевает указание атрибута src в теге <SCRIPT>, значением которого является URI внешнего скрипта:

```
<HTML><BODY>
...
<SCRIPT type="text/javascript" src="external.js"></SCRIPT>
...
</BODY></HTML>
```

Как уже упоминалось, встречая тег <SCRIPT> с атрибутом src, браузер пытается получить данные из источника, указанного значением этого атрибута, и интерпретировать их как программный код на языке, обо-

значенном в атрибуте `type`. При этом все содержимое элемента `SCRIPT` полностью игнорируется. Фактически, это выглядит так, как если бы содержимое элемента `SCRIPT` заменялось на содержимое внешнего скрипта.

Данный метод обладает несколькими существенными преимуществами по сравнению с непосредственным включением кода в Web-страницу:

- отделение логики от представления. Это подразумевает раздельное существование структуры, предоставляющей пользовательский интерфейс (в нашем случае — HTML-документ) и алгоритмов работы с данными и управления этой структурой (в нашем случае — внешний скрипт). Такой подход, в частности, позволяет вести раздельную разработку интерфейса и алгоритмов обработки данных (часто — разными разработчиками);
- возможность использования одного внешнего скрипта многими Web-страницами;
- возможность динамической генерации кода скрипта на сервере;
- интенсивное кэширование подключаемых данным способом скриптов современными браузерами. Это приводит к уменьшению трафика для конечных пользователей и увеличению скорости загрузки страниц сайтов.

### Включение кода в обработчики событий элементов Web-страниц

Для многих элементов Web-страниц существует возможность назначения обработчиков событий пользовательского ввода (движение указателя мыши, потеря и получение фокуса, нажатия клавиш). Обработчики событий задаются путем указания значений специальных атрибутов тегов этих элементов. Например, для назначения обработчика события движения мыши для некоторого элемента необходимо определить для него атрибут `onmousemove`. Значениями таких атрибутов являются строки программного кода на одном из языков сценариев, поддерживаемых браузером. Например, следующий код демонстрирует назначение обработчика события щелчка мышью элементу `DIV`:

```
<HTML><BODY>
...
<DIV onclick="alert('Щелчок мышью');">
  Щелкните мышью здесь</DIV>
...
</BODY></HTML>
```

### Включение кода в виде псевдо-URL

Этот способ похож на предыдущий. Он подразумевает формирование URL специального вида путем указания протокола доступа к данным `javascript:`, после чего записывается строка программного кода. Подобный URL может быть задан, например, как значение свойства `href` тега `<A>`, свойства `action` тега `<FORM>`, свойства `src` тега `<FORM>` или `<IMG>`. Например:

```
<HTML><BODY>
...
<a href="javascript: alert('Щелчок');">Ссылка JavaScript</a>
...
</BODY></HTML>
```

Данный способ часто используют с целью создания на странице ссылок для добавления этой страницы в список закладок браузера и других похожих задач. *Подробнее об использовании протокола javascript: будет рассказано в главе 8.*

**Примечание**

Следует заметить, что при использовании двух последних методов включения JavaScript-кода в HTML-документ обязательно указание для документа языка сценариев по умолчанию. Это может быть сделано путем включения соответствующего объявления `META` в тег `HEAD`: `<META http-equiv="Content-Script-Type" content="text/javascript">`, либо HTTP-заголовком `Content-Script-Type: text/javascript`.

## Порядок исполнения скриптов в документе

Блоки кода JavaScript, внедренные в HTML-документ с помощью тега `<SCRIPT>` (как непосредственно, так и из внешнего источника), исполняются последовательно в том порядке, в котором они найдены браузером в документе. При этом блоки с установленным атрибутом `defer` могут быть проигнорированы и исполнены позже (установка атрибута `defer` не обеспечивает безусловного отложенного исполнения скрипта, а лишь декларирует такую возможность).

Код, добавленный в документ в виде обработчиков событий элементов либо в виде псевдо-URL, исполняется при возникновении соответствующих событий.

# Основные характеристики языка

Теперь рассмотрим характеристики JavaScript, обуславливающие правила написания программного кода.

## Регистрозависимость

JavaScript-код интерпретируется с учетом регистра символов. Это означает, что исполняющая система различает строчные и прописные буквы в написании идентификаторов, ключевых слов, операторов и других символов языка. Таким образом, в следующем фрагменте кода производится объявление, и присваиваются значения трем различным переменным:

```
var nvalue = 10;
var nValue = 20;
var nVALUE = 30;
```

## Интерпретация пробельных символов

Пробельные символы (символы перевода строки, символы табуляции, пробелы) в JavaScript служат преимущественно для форматирования кода. Они не определяют логическую структуру программы (как, например, в языке Python). Количество следующих подряд пробельных символов не имеет значения. В случаях простых и однозначных конструкций пробелы могут опускаться. Таким образом, следующие строки кода семантически идентичны:

```
x=Math.sqrt(10);
x = Math.sqrt(10);
x           =    Math.sqrt   (   10   )    ;
```

Тем не менее во многих случаях удаление пробелов (например, между некоторыми операторами и идентификаторами) приводит к образованию новых конструкций, не идентичных исходным. Рассмотрим следующий код:

```
var oMyObject = new MyObject;
```

Удалив здесь один пробел, получим совершенно иную по смыслу инструкцию:

```
var oMyObject = newMyObject;
```

Количество пробельных символов и их тип также имеют значение при определении литералов строк.

## Комментарии

В JavaScript существуют два вида комментариев: однострочные и многострочные. Однострочный комментарий начинается с двух знаков "прямой слэш" (//) и заканчивается символом перевода строки (т. е. в конце строки вы нажимаете клавишу <Enter>). Пример однострочного комментария:

```
var x = 10; // однострочный комментарий
```

Многострочные комментарии начинаются последовательностью символов /* и заканчиваются */. Например:

```
/*
  Многострочный комментарий
  Многострочный комментарий
*/
```

Следует учесть, что не стоит пытаться записывать вложенные многострочные комментарии, т. к. первая же последовательность символов */ вложенного комментария будет воспринята интерпретатором как признак завершения всего комментируемого блока, а оставшаяся часть закомментированного текста будет рассматриваться как программный код, что, скорее всего, вызовет возникновение синтаксических ошибок.

## Запись инструкций, символы-разделители

Инструкции JavaScript могут быть записаны в одну или несколько строк и могут разделяться символом ;. При записи нескольких инструкций в одну строку, их разделение точкой с запятой обязательно. Могут существовать *пустые инструкции*, образующиеся при записи подряд нескольких символов ;. Разделение инструкций не обязательно, если каждая из них начинается с новой строки (правда, это не лучший стиль оформления кода, к тому же при таком подходе повышается вероятность допустить синтаксические ошибки). Например, следующий код корректен:

```
var x = 10;
x =
  x +
  4; ;; x = x - 2;
y = x + 5
x = y - 10
```