Введение

Примерно 20 лет назад я написал книгу, о которой давно думал и которая должна была помочь мне освоить новую технологию. Я всегда старался вникнуть в основы новых программных средств и средств разработки, почувствовать растущие достижения и разделить с читателями плоды своего труда. Пятое издание книги *JavaScript. Библия пользователя* призвано донести до читателя те знания и опыт, которые были наработаны в течение пяти лет каждодневной работы в JavaScript и постоянного поиска в группах новостей каверзных вопросов, нерешенных проблем и другого, достойного внимания разработчика сценариев. Моя цель — помочь читателю избежать тех разочарований и мучительной головной боли, которые в свое время испытал не только я сам, но и многие другие, кто работал с различными поколениями браузеров.

В то время как предыдущие издания этой книги были, в основном, ориентированы на доминирующие в то время браузеры Netscape Navigator, современные тенденции развития браузерных технологий все больше указывают на преобладание Microsoft Internet Explorer. В то же время, Netscape удалось завершить превосходную задачу по восстановлению собственного браузера в свете быстрого развития стандартов индустрии. Как результат этих тенденций, существенно пересмотренное и расширенное пятое издание книги в равной степени ориентировано на оба типа браузеров как на эквивалентные технологии, которые вполне соответствуют приоритетам пользователей. Читатель столкнется с моими положительными отзывами или критикой разных версий браузеров и поддерживаемых ими технологий. Но мое самое страстное желание, на самом деле, — это научить читателя создавать документы и разрабатывать производительные сценарии, эффективно выполняемые в любой версии браузера. Поэтому данная книга содержит детальное описание специальных и стандартных методов, позволяющих решать любые поставленные задачи. По моему мнению, это именно то, что больше всего волнует читателей. На случай, если вы усмотрите в этой книге спорные моменты, то не забывайте о том, что создаваемые сценарии должны выполняться в максимально возможном числе браузеров.

Структура и особенности этого издания

Подобно предыдущему изданию, настоящее пятое издание книги *JavaScript. Библия пользователя* содержит намного больше информации, чем можно поместить даже в самую толстую книгу по JavaScript. Полная версия приведена на компакт-диске (в электронном виде), прилагаемом к этой книге. Обновленное издание книги лучше структурировано (это в основном касается информации, представленной в бумажной части книги). В частности, все листинги кодов, которые ранее можно было найти только в виде материалов на компакт-диске, теперь вставлены в текст книги в местах его непосредственного описания. В бумажную часть книги вынесены все главы, в которых рассматриваются основы языка и фундаментальные принципы программирования на JavaScript. В электронных главах книги на компакт-диске описаны узкоспециализированные методы разработки сценариев на JavaScript. Ниже речь пойдет о структурных особенностях книги.

Часть I. Знакомство с JavaScript

Часть I этой книги начинается с главы, в которой продемонстрированы связи JavaScript с Java и определяется их роль в World Wide Web. С того момента, как на рынке Web-технологий впервые появился JavaScript, в браузерах Web и среде создания сценариев произошли существенные изменения. Поэтому глава 2 посвящена тем изменениям, которые наиболее существенным образом влияют на методы создания сценариев и в большей степени определяют быстро изменяющиеся стандарты приложений, поддерживаемые разными браузерами. В главе 3 вы совершите первое знакомство с основами JavaScript; после ее изучения вы сможете самостоятельно создать первый в своей жизни сценарий.

Часть II. Руководство по JavaScript

Часть II полностью посвящена наставлениям для начинающих программистов на JavaScript. В ней приведено девять уроков, позволяющих читателю последовательно изучить внутреннюю структуру браузера, освоить навыки программирования и ознакомиться с основными принципами JavaScript. С помощью всего нескольких явно описанных элементов языка в этих уроках на высоком профессиональном уровне рассматривается весь диапазон принципов создания сценариев, выполняемых во многих браузерах, которые поддерживают технологию JavaScript. В конце каждого урока приведены упражнения, которые позволяют закрепить полученные знания и проверить качество их освоения читателем. Вы также сможете попрактиковаться в использовании изученных средств (ответы приведены в приложении В). Цель этого учебного пособия — научить читателя создавать простые страницы еще до того момента, как дело дойдет до подробного рассмотрения сложных примеров, приведенных в конце книги. В последнем уроке части вы узнаете, как управлять многофреймовыми документами и создавать эффекты ролловеров — изображений, изменяющихся при наведении на них указателя мыши.

Часть III. Объекты документа

Часть III — это наибольшая часть книги, которая посвящена всестороннему освещению объектных моделей документа, реализуемых в браузерах и обеспечивающих до настоящего времени средства создания динамических документов. Во всех справочных разделах совместимость описываемых средств JavaScript с разными версиями браузера задается в специальной таблице. В главе 15 приведен справочный материал, который используется практически во всех остальных главах части III.

Часть IV. Объекты языка JavaScript

В части IV приведена справочная информация по базовому языку JavaScript. Как и в части III, будут представлены таблицы совместимости браузеров с описываемыми средствами языка JavaScript. Ключевые слова в нижней части страниц помогут читателю быстро отыскать нужный термин, описываемый на странице.

Часть V. Приложения

Несколько приложений, приведенных в конце книги, предоставят довольно важный справочный материал. В них включена справка по JavaScript и объектам браузера (приложение А), список зарезервированных в JavaScript слов (приложение Б) и ответы на упражнения части II (приложение В). В приложении Г представлен список Internet-ресурсов, посвященных JavaScript, а в приложении Д описано содержимое компакт-диска и электронных глав книги.

Прилагаемый компакт-диск

Прилагаемый к книге компакт-диск — это кладезь полезной информации. На диске вы найдете все главы дополнительной части VI, представленной только в электронном виде. В этих главах рассмотрены следующие вопросы.

- Специальные объекты DOM, XML и JavaScript.
- Возможности DHTML, методы проверки данных и защита сценариев.
- Методы разработки и отладки профессиональных Web-приложений.
- Девять полноценных приложений JavaScript, используемых в реальной жизни.

Более того, на компакт-диске, прилагаемом к книге, содержится электронная версия настоящей книги (на английском языке, в формате PDF). Вы также найдете на нем огромное количество готовых к использованию HTML-документов, которые послужат примером создания Web-страниц в большинстве объектных моделей документа. Все они созданы с использованием зарезервированных слов JavaScript и методов, описанных в частях III и IV настоящей книги. Эти примеры можно запускать непосредственно в браузере, по умолчанию поддерживающем JavaScript. Однако обязательно убедитесь, что в папке листингов (как шлюз для их запуска) используется страница `index.html`. Эта страница определяет те браузеры, которые совместимы с приведенными в папке листингами. Она содержит немного информации, но ознакомиться с ней перед запуском остальных сценариев советуем всем пользователям. Просмотр полноценных HTML-документов (даже достаточно простых) просто необходим для полного усвоения всех описанных в книге принципов. Листинги из учебного раздела этой книги (часть II) умышленно опущены, чтобы привлечь читателя к самостоятельному написанию сценариев. Вы многому научитесь, даже просто вводя листинги, приведенные в книге, — стоит только начать. На компакт-диске представлены листинги из частей I и V.

Не забудьте проверить папку с листингами для главы 13. Эта папка содержит весьма интересный файл `evaluator.html`, созданный специально для этой книги. Читатель может познакомиться с результатом использования этого приложения на практике, чтобы получить детальные сведения о его эффективности.

Справочное руководство из приложения A также представлено на диске в формате `.pdf`. Это позволяет распечатать его и использовать в качестве настольного пособия. На диске также содержится программа Adobe Acrobat Reader, позволяющая просматривать содержимое файлов в формате `.pdf`. Наконец, для упрощенного поиска на диске в том же формате представлен текст всей книги.

Что потребуется для изучения JavaScript

Для чтения этой книги совсем не обязательно иметь опыт программирования, однако чем больше читатель создал с помощью HTML Web-страниц, тем легче ему будет понять, как JavaScript взаимодействует с разными программными элементами, размещенными на странице. Иногда, чтобы ощутить все прелести написания сценариев, придется несколько видоизменять HTML-дескрипторы. Если вам ранее приходилось работать с этими дескрипторами, то изучение новых особенностей, добавляемых JavaScript, не представит для вас особых сложностей.

В большинстве стандартных разработок JavaScript формы и их элементы (текстовые поля, кнопки и списки выбора) играют особенно важную роль. Читателю стоит внимательно ознакомиться с этими элементами и их атрибутами в HTML. К счастью, вам не потребуется разбираться в тонкостях составления серверных сценариев и способах отправки информации с формы на сервер. В данном случае основной акцент сделан на написании

сценариев, выполняемых на стороне клиента (после полной загрузки браузером, поддерживающим JavaScript, в составе страницы они управляются независимо от сервера).

Вам также потребуются знания основных стандартов HTML. Например, при описании основ управления фреймами, главный упор делается на написании сценариев для соответствующих структурных элементов, а не на методах оформления с их помощью готовых страниц. Более детально данная тема описана в документации от Microsoft, Netscape и т.п.

Для тех, кому не приходилось программировать

Если вы изучали азы работы в HTML несколько лет назад по тоненькому справочнику, то размеры этой книги могут вас просто испугать. Может быть, JavaScript далеко не самый простой для изучения язык в мире, но, вне всяких сомнений, он ни в какое сравнение не идет с такими фундаментальными средами программирования, как Java или C. В отличие от разработки завершенных приложений (которые практически каждому из вас приходилось приобретать на компакт-дисках), с помощью JavaScript можно экспериментально создавать небольшие фрагменты программного кода, используемые в качестве компонентов более сложных программ. Интерпретатор JavaScript, который встроен во все браузеры, поддерживающие сценарии, выполнит за вас бо́льшую часть черновой работы.

Программирование заключается в создании серии инструкций для выполнения их компьютером. Если проводить аналогию с повседневной жизнью, то люди постоянно следуют инструкциям, даже если сами этого не подозревают. Путешествие к дому друга является примером следования последовательности несложных инструкций: пройти три квартала в одном направлении; возле аптеки повернуть налево; после прохождения ее повернуть направо. Выполняя эти инструкции, вам приходится принимать определенные решения: если на светофоре горит красный свет, то нужно остановиться; если — зеленый, то можно идти; если — желтый, то следует приготовиться к движению или остановке. Время от времени некоторые действия приходится повторять по несколько раз (как, например, хождение по улицам в поисках заветной аптеки). Программа для компьютера содержит не только последовательность основных действий, но и предопределяет, какие решения принимать или какова повторяемость действий (как, например, реагировать на сигналы светофора или как вести себя в случае, если рядом расположено две аптеки) для достижения целей, стоящих перед программой.

Первым препятствием на пути к освоению азов программирования является та точность, с какой в языках программирования представлены в инструкциях слова и числа. Соответствующие правила (как и в обычных языках) определяют синтаксис языка программирования. Хотя современные компьютеры являются весьма производительными, по своей натуре они до безобразия точны. Поэтому они не очень-то прощают, когда с ними разговаривают на непонятном для них языке. Если вы, разговаривая с другим человеком, допускаете в предложении стилистические ошибки, то вы все же, рассчитываете, что собеседник вас полностью поймет. В большинстве случаев так и происходит. Совсем иначе дело обстоит с компьютерами. Если синтаксис не является идеальным (или, по крайней мере, не отклоняется от стандарта в пределах очевидной корректировки), компьютер самым нахальным образом заявит, что вами допущена синтаксическая ошибка.

Оптимальным вариантом в такой ситуации является изучение допущенных синтаксических ошибок. Ведь даже самые опытные программисты делают ошибки. При таком подходе каждая допущенная синтаксическая ошибка добавляет знаний и опыта в программировании на используемом языке.

Тем, кто уже немного программировал

Наличие опыта работы с такими процедурными языками программирования, как BASIC или Pascal, может, скорее, вызвать головную боль, чем помочь в изучении JavaScript. Вы можете иметь достаточно полное представление о синтаксисе языка, однако важно и то, что

общая концепция выполнения программ в этих языках радикально отличается от принятой в JavaScript. Частично это определяется обычными задачами, решаемыми с помощью сценариев (программирование действий в ответ на выполняемые пользователем манипуляции на странице Web). Однако большой объем работы реализуется посредством объектно-ориентированного программирования.

В типичной программе с использованием процедур программист непосредственно несет ответственность как за то, что появляется на экране, так и за те действия, которые непосредственно выполняются программой. При первом запуске программы огромная часть кода предназначена для того, чтобы определить графическую среду. На экране при этом может располагаться несколько текстовых полей и кнопок. Чтобы определить, на какой кнопке щелкнул пользователь, программа должна знать координаты точки, на которой был выполнен щелчок, а затем сравнить эти координаты на совпадение с координатами одной из кнопок на экране. В зависимости от того, какая кнопка используется, выполняются заранее определенные для текущего случая инструкции.

Объектно-ориентированное программирование представляет собой нечто совершенно противоположное. Кнопка интерпретируется как объект — вещь вполне реальная. Объект имеет свойства (надпись, размер, событие и т.п.). Объект также может содержать сценарий. В то же время, системные программы и браузеры могут посылать объекту сообщения — в зависимости от того, что делает пользователь, — для запуска сценария. Например, если пользователь щелкает мышкой на записи текстового поля, то система или браузер сообщает полю, что кто-то щелкает там (это значит, что данное поле активизируется), передавая полю полномочия по поводу принятия решения, что ему следует делать в ответ на такое действие. Именно здесь в силу вступают сценарии. Сценарий связан с полем и содержит инструкции, которые выполняются при активизации поля. Другой набор инструкций может использоваться для контроля данных, вводимых пользователем в поле, или на вкладке, или в результате щелчка на опции.

Некоторые из создаваемых сценариев по своей структуре могут напоминать процедуры: они содержат простые инструкции, которые поочередно выполняются. Однако когда вы перейдете к управлению данными формы, то поймете, что инструкции JavaScript выполняются исключительно как объектно-ориентированные. Каждая форма является объектом. Это же относится ко всякому элементу управления (например, переключателю или текстовому полю). В этом случае для выполнения запланированной работы сценарий вызывает свойства соответствующих объектов.

Переход от процедурного к объектно-ориентированному программированию может быть самым трудным моментом в обучении пользователя. Когда несколько лет назад мне самому пришлось столкнуться с объектно-ориентированным программированием, не могу сказать, что у меня все получилось. Но когда я во всем разобрался (а в этом мне помог напряженный и ответственный труд), в моей голове появилось столько “грандиозных” идей, что, я уверен, буду реализовывать их еще много лет. С тех пор я полностью уверен, что объектно-ориентированное программирование является единственным целесообразным способом написания программ.

Тем, кто программировал на C

По причине занудности синтаксиса Java (который, кстати сказать, походит от C и C++), в JavaScript можно найти много синтаксических характеристик C. Поэтому знакомые с языком C программисты будут чувствовать себя, как рыба в воде. Символы операторов, условные конструкции и циклы выдержаны исключительно в традициях C. При этом в JavaScript, по сравнению с C, можно меньше беспокоиться о типах данных. В JavaScript задаваемые переменные не ограничены конкретным типом данных.

Поскольку очень многое в синтаксисе JavaScript будет вам знакомо, то сконцентрируйтесь на концепциях объектной модели документа, что является абсолютно новым понятием для C-программиста. Кроме того, для реализации своих навыков в JavaScript на практике советуем основательно изучить HTML (особенно это касается дескрипторов элементов форм).

Для тех, кто работал в Java

Несмотря на очень похожие названия, данные языки имеют только поверхностное сходство. Это касается организации циклов и условных конструкций, обращения к объектам с использованием точки (как это делается в C), фигурных скобок для выделения групп, некоторых ключевых слов и ряда других атрибутов. Что же касается объявления переменных, то они совершенно различны. Такая ситуация имеет место по той причине, что JavaScript в этом смысле является достаточно прогрессивным языком. Переменная может содержать целочисленное значение с одной стороны оператора сравнения и строчные данные — с другой (хотя это и не всегда хорошо). В тех случаях, когда Java обращается к чему-то как к методу, JavaScript вызывает методы (когда они связаны с предопределенным объектом) или функции (для определенных в сценарии действий). Методы и функции JavaScript могут возвращать значения любых типов без предварительного их объявления.

Возможно, при работе с JavaScript наиболее важным наследием Java является объектно-ориентированное представление о классах, наследовании, экземплярах и передаче сообщений. Эти понятия при написании сценариев просто неактуальны. Тем не менее, JavaScript-дизайнеры хорошо знают, что всегда нужно подстраховываться. Например, хотя в JavaScript и не требуется в конце каждой строки ставить точку с запятой, однако, если в исходном коде JavaScript парочка-другая их все же "проскакивает", то интерпретатор JavaScript от этого хуже работать не станет.

Тем, кто писал сценарии и макросы

Опыт создания сценариев с помощью других средств программирования или макросов в прикладных программах является хорошей основой при освоении всевозможных концепций JavaScript. Наиболее важной из них, пожалуй, является концепция комбинирования действий или обработки данных определенного типа с помощью нескольких операторов. Например, в Microsoft Excel можно написать макрос, который будет выполнять преобразование данных финансовых отчетов, поступающих из другого компьютера, для отображения в виде диаграмм. Макрос встраивается в меню Макрос, и запускать его нужно, выбрав соответствующую опцию меню, независимо от того, когда поступают новые показатели.

Более сложные сценарии, который можно создать с помощью Toolbook или HyperCard, максимально соответствуют объектно-ориентированной концепции JavaScript. В этих средах объекты экрана содержат сценарии, которые выполняются при взаимодействии пользователя с этими объектами. Большое число создаваемых в JavaScript сценариев будут выдержаны в рамках описанной ниже концепции. Фактически, эти среды напоминают оболочку выполняющего сценарии браузера, обеспечивая использование набора предопределенных объектов, которые имеют фиксированные свойства и атрибуты. Эта предопределенность применяется для строгого определения характеристик всей среды, а потому способствует более эффективному и производительному выполнению приложений.

Соглашение о представлении данных и имен

Листинги сценариев и их фрагменты представлены в этой книге `специальным шрифтом`. Это сделано для того, чтобы облегчить их поиск на фоне остального текста. По причине ограниченности ширины страниц строки в листингах иногда будут неестественным образом прерываться. В этих случаях разорванная строка продолжается на следующей строке, как это делается в текстовом редакторе при переносе слов. Если при введении сценария из-за этого у вас возникают проблемы, можно воспользоваться этим же сценарием, размещенным на компакт-диске, прилагаемом к книге.

Когда читатель доберется до части III книги, может оказаться, что дальше, чем на одну страницу после описания объектных моделей или средств языка, требующих определенной

версии браузеров, ему продвинуться не удается. Чтобы указать, какой же в данном случае необходим браузер и его версия, используется двухсимвольная система аббревиатур с указанием числовой версии. Например, IE5 означает Internet Explorer 5 для всех операционных систем, а NN6 означает Netscape Navigator 6, опять же, для всех операционных систем. Если сценарий или отдельное средство поддерживается определенной версией браузера и всеми последующими версиями, то после номера версии стоит знак (+). Например, если средство помечено как IE4+, то это означает, что для его поддержки вам потребуется как минимум Internet Explorer 4, но средство совместимо и с IE5, и с IE5.5 и т.д. Случается, что средство или некоторый атрибут применим только в определенной операционной системе. К примеру, если для средства использована пометка WinIE4+, то это значит, что оно выполняется в Internet Explorer версии 4 и выше, но только в Windows. Как пример ссылок, первыми работающими со сценариями браузерами были NN2, WinIE3 и MacIE3.01. Эти условные обозначения в первую очередь можно увидеть на сравнительных диаграммах в справочных разделах.

Пиктограммы “На заметку”, “Совет” и “Внимание” время от времени появляются в книге для отображения наиболее важных моментов.

Благодарности

Конечно, такую толстенную книгу, которую вы держите в своих руках, мне никогда не удались бы написать самостоятельно за те короткие сроки, которые устанавливаются конъюнктурой рынка печатных изданий. В процессе изучения терминологии и подготовки обучающих примеров к этой книге мне приходилось неоднократно консультироваться с Майклом Моррисоном — таким же автором, как и я сам. С его помощью проект приобрел тот вид, который имеет сейчас. Технический редактор Дэвид Уолл на протяжении многих лет занимается подготовкой книги *JavaScript. Библия пользователя* к изданию, на что у него уходит очень много сил и почти все рабочее время. Хотелось бы также отметить и многих других сотрудников Wiley Publishing, принимавших участие в написании и подготовке к печати этой книги. Среди них Дебра Вильямс Коли, Мэри Бет Уэйкфилд и Анжела Смит. И наконец, я искренне признателен всем тем читателям предыдущих изданий книги, которые принимали активное участие в ее обсуждении: присылали электронные письма, в которых выражали свои замечания и пожелания. Ваш опыт работы с JavaScript оказался просто неоценим!

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится вам эта книга или нет, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: info@dialektika.com

WWW: http://www.dialektika.com

Адреса для писем:

из России: 115419, Москва, а/я 783

из Украины: 03150, Киев, а/я 152