ГЛАВА

3

Первый собственный сценарий

В этой главе...

Программное обеспечение

Настройка среды разработки

Что будет делать первый сценарий

Создание первого сценария

Исследование сценария

Улыбнемся вместе

В этой главе речь пойдет об установке на компьютер программных продуктов, позволяющих с высокой производительностью создавать сценарии, а также выполнять их. Мы также постараемся написать простой сценарий, результат выполнения которого можно просмотреть в любом совместимом с JavaScript браузере.

Благодаря отличиям в работе разных операционных систем, основные настройки среды будут ориентированы на использование двух наиболее популярных систем: Win32 (Windows 95–XP) и MacOS X. В большинстве случаев изменение операционной системы, вплоть до выбора Linux и UNIX, никак не повлияет на решение и результат поставленных задач. Безусловно, для разных браузеров и операционных систем могут наблюдаться некоторые незначительные различия в структуре представления параметров, однако базисные положения останутся неизменными. Большинство рисунков, а также графически иллюстрируемые результаты, полученные в браузере, в этой книге выполнены для Internet Explorer 6 в Windows XP. Поэтому, если вы используете другой браузер и результаты его в окне не соответствуют приведенным в книге, то это не должно вас тревожить.

Программное обеспечение

Лучший способ изучения JavaScript — введение кода сценария непосредственно в текстовом редакторе. Выбор самого редактора остается всецело за вами, хотя в следующем разделе представлены некоторые соображения по этому поводу.

Выбор текстового редактора

С точки зрения материала данной книги, читателю пока следует избегать графических программных средств создания Web-страниц типа WYSIWYG (What You See Is What You Get — Что видишь, то и получаешь) — FrontPage или Dreamweaver.

Эти средства определенно понадобятся вам в будущем, когда у вас будет опыт представления элементов страницы в виде кода. Тогда для формирования основного содержимого документа и макета они вам особо пригодятся. Примеры, приводимые в данном издании, основаны на создании сценариев, которые в любом случае придется вводить вручную, поэтому у знатоков кода HTML в этом смысле проблем возникать не должно. Файлы листингов готовых Web-страниц, описанных в этой книге (кроме учебных глав), вы найдете на прилагаемом компакт-диске.

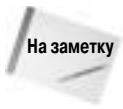
Важным фактором при выборе текстового редактора является удобство, с каким он позволяет вводить и сохранять стандартные текстовые файлы с расширением `.html`. В случае использования Windows любая программа, если только она сохраняет текстовые файлы, по умолчанию позволяет добавлять к ним расширение `.htm` или `.html`, т.е. способна избавить вас от проблемы выбора специального редактора. Если, к примеру, выбрать программу Microsoft Word, то при сохранении текстовых файлов она попытается представить их в двоичном виде — такой файл не в состоянии загрузить ни один браузер Web. Чтобы первоначально сохранить файл как текстовый с расширением `.html`, вам придется провести некоторые настройки в диалоговом окне Save As (Сохранить как). А это раздражает.

Нет ничего страшного в использовании обычного текстового редактора. В той же Windows, предлагающей редактор WordPad, можно установить и более совершенные продукты для введения кодов листингов (например, испытательную версию программы TextPad). Для системы MacOS X вполне подойдет TextEdit, хотя отсутствие функции поиска и замены может привести к ряду трудностей при управлении кодами Web-страниц. Наиболее популярной среди разработчиков HTML и создателей сценариев, использующих Mac, является программа BBEdit (от Bare Bones Software), в состав которой входит большое количество вспомогательных средств для составления сценариев (например, опции нумерации строк, которые помогают при отладке кода JavaScript).

Выбор браузера

Другим компонентом, необходимым при изучении JavaScript, является браузер. Для того чтобы протестировать собственные сценарии с помощью браузера, совсем необязательно быть подключенным к Internet. Все тестирование можно вполне выполнить и в автономном режиме. А это значит, что изучать JavaScript и создавать с использованием сценариев по-настоящему привлекательные Web-страницы можно с помощью небольшого портативного компьютера — прямо хоть в лодке посреди океана.

Тип используемого браузера и его версия целиком определяются вкусами читателя. При разработке страниц советуем устанавливать максимально современный браузер (IE5 и выше для Windows и Macintosh, любой Mozilla-браузер, включая NN7 и выше, и Apple Safari).



В примерах листингов в этой книге используются свойства языка или объектные модели документа (DOM), которые доступны только в определенных браузерах и их версиях. Поэтому проверяйте совместимость введенного кода с используемой версией языка или моделью DOM, чтобы убедиться, что для загрузки страницы был применен подходящий браузер.

Настройка среды разработки

Чтобы облегчить себе задачу по тестированию сценариев, убедитесь, что в компьютере установлено достаточно памяти для одновременного запуска браузера и текстового редактора. При экспериментировании и устранении всевозможных ошибок, которые могут быть допущены в сценарии и, вам нужно будет быстро переключаться между редактором и браузером. В обычном рабочем режиме вам придется выполнить такие действия.

1. Ввести в текстовом редакторе документа исходный HTML-код и код сценария.
2. Сохранить последнюю версию кода на диске.
3. Переключиться на браузер.
4. Дальше можно поступить несколькими способами. Если это новый документ, то откройте его с помощью команды Open (Открыть) браузера. Если же этот файл уже загружен, перезагрузите его в окне браузера.

Последовательность действий, описанную в пп. 2–4, придется повторять довольно часто. Так и хочется назвать эту трехэтапную последовательность сохрани-перключись-перезагрузи. Такую последовательность действий при создании сценариев придется выполнять столь часто, что в конце концов эта физическая процедура будет выполняться вами подсознательно. Как именно организовывать окна приложений на экране и выполнять последовательность действий сохрани-переключись-перезагрузи, зависит от конкретной используемой операционной системы.

Система Windows

В этом случае для "получения полного эффекта" не следует постоянно держать развернутыми на весь экран окна редактора или браузера. Работать будет удобнее, если настроить размеры и расположение окон так, чтобы они имели достаточно большой размер, но при этом оставалась возможность щелкать мышью на заголовках окон. В любом случае можно обратиться за помощью к панели задач и щелкнуть на кнопке соответствующего приложения, чтобы перейти к необходимому окну (рис. 3.1). Если выставить разрешение монитора большим, чем 800×600 пикселей, то это добавит места для окон и панели задач.

Рис. 3.1. Расположение окон редактора и браузера при использовании Windows

На практике, тем не менее, наличие используемой в Windows для переключения между выполняемыми приложениями комбинации клавиш <Alt+Tab> приводит к тому, что реализовать последовательность сохрани-переключись-перезагрузи становится очень просто. Если вами используется система Windows, в которой запущен совместимый с ней текстовый редактор (в котором, скорее всего, для сохранения документа используется комбинация клавиш <Ctrl+S>), то выполнить последовательность сохрани-переключись-перезагрузи будет очень

просто, даже без использования мыши: для сохранения исходного файла нужно нажать <Ctrl+S>, затем переключиться на браузер с помощью комбинации <Alt+Tab> и после этого перезагрузить сохраненный файл, нажав <Ctrl+R>.

Чтобы обратно переключиться из окна браузера к окну текстового редактора, еще раз нажмите <Alt+Tab>.

Система MacOS X

В MacOS X для переключения между программами вам придется использовать средство Dock или комбинацию клавиш <⌘+Tab>. Если одновременно запущено только два приложения, то для перехода от одной из них к другой нажмите <⌘+Tab> (рис. 3.2).

Рис. 3.2. Расположение окон редактора и браузера на экране Macintosh

В таком случае последовательность сохрани-переключись-перезагрузи потребует уже некоторых усилий.

1. Нажмите <⌘+S>, сохранив тем самым исходный файл.
2. Щелкните мышью на окне браузера.
3. Нажмите <⌘+R> и перезагрузите сохраненный исходный файл.

Для того чтобы вернуться к редактированию исходного файла, нажмите <⌘+Tab>.

Особенности перезагрузки

В большинстве случаев простой перезагрузки страницы вполне достаточно для тестирования измененного сценария. Но в некоторых случаях в кэш-памяти браузера (с используемыми по умолчанию настройками) могут при перезагрузке сохраниться некоторые из предыдущих атрибутов страницы, даже если ее исходный код изменен. Для выполнения более основательной перезагрузки при использовании кнопки перезагрузки браузера Reload/Refresh

(Перезагрузить/Обновить) удерживайте нажатой клавишу <Shift>. Альтернативой является отключение кэш-памяти браузера в разделе установок. Однако такие изменения могут негативно сказаться на общем характере функционирования браузера в сеансах работы в Web.

Что будет делать первый сценарий

Сценарий, который будет рассматриваться в следующем разделе, относится к тем, которые запускаются автоматически при загрузке браузером HTML-страницы. Поскольку в данном случае вся работа, связанная со сценарием и загрузкой, выполняется в автономном режиме, то поведение страницы будет таким же, как если бы исходный файл был размещен на сервере и кто-то получал к нему доступ с помощью Web.

На рис. 3.3 показано, как будет выглядеть готовая страница. Правда, если используется операционная система, отличная от Windows XP, или в качестве браузера задействован не Internet Explorer, то некоторые несущественные детали могут слегка отличаться. Та часть страницы, что в HTML определяется обычным способом, не содержит ничего особенного: заголовок уровня `<h1>` с горизонтальным разделителем под ним. Если используется браузер, не поддерживающий JavaScript, то все, что можно будет увидеть, так это указанные заголовок и горизонтальный разделитель. Если вами используется совсем уж старомодный браузер, то в этом случае на странице могут появиться еще и некоторые операторы сценария.

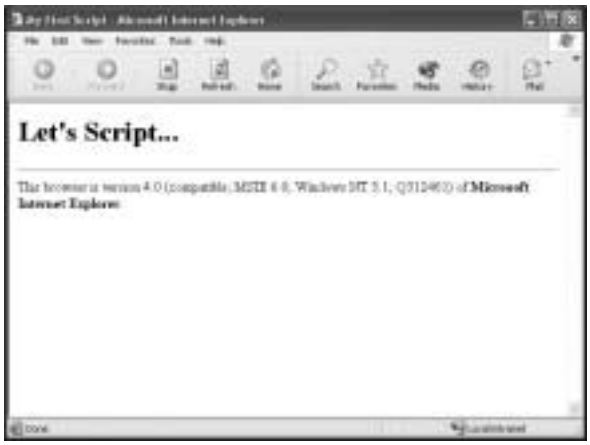


Рис. 3.3. Завершенная страница первого сценария JavaScript

Под разделителем сценарий отобразит текстовую информацию, состоящую из неформатированных данных сведений о браузере, используемом для загрузки документа. Сценарий записывает поток данных HTML в браузер, включая дескрипторы для отображения некоторых фрагментов текста полужирным начертанием. Хотя на странице текстовая информация разбита на две строки, передается она как одна строка — ситуация, подобная той, что имеет место при жестком форматировании текста в HTML.

Создание первого сценария

Итак, пришло время начать работу над первым собственным сценарием JavaScript. Для этого сначала запустите текстовый редактор и браузер. Если браузер предлагает соединиться с провайдером Internet (Internet Service Provider или ISP) или начинает делать это автоматически,

то операцию следует отменить и выйти из программы установки соединения. Если в браузере активна кнопка Stop (Остановить), то для пресечения возможных попыток браузера установить соединение следует незамедлительно ею воспользоваться. При этом может появиться диалоговое окно с сообщением о том, что адрес URL для начальной страницы браузера (обычно такая страница является страницей издателя браузера, если только пользователем не были внесены изменения в настройки) не определен. Это нормально. Ведь вам нужно было всего-навсего запустить браузер, а не соединиться с ISP. Если вы используете автоматическое соединение через местную локальную сеть, то ничего страшного. Более того, пока что сетевое соединение вам не потребуется. Ниже приведена последовательность операций, позволяющих создать и просмотреть первый собственный сценарий JavaScript.

1. Запустите текстовый редактор и создайте новый пустой документ.
2. Введите в окне документа сценарий, как это показано в листинге 3.1.

Листинг 3.1. Исходный код сценария `script1.htm`

```
<html>
<head>
<title>My First Script</title>
<style type="text/css">
.highlight {font-weight: bold}
</style>
</head>

<body>
<h1>Let's Script...</h1>
<hr>
<script type="text/javascript">
<!-- скрытие сценария от старых браузеров
document.write("This browser is version " + navigator.appVersion);
document.write(" of <span class='highlight'>" +
   navigator.appName + "</span>.");
// завершение кода скрытия сценария от старых браузеров -->
</script>
</body>
</html>
```

3. Сохраните документ под именем `script1.htm`.
4. Перейдите в браузер.
5. В меню File (Файл) браузера воспользуйтесь опцией Open (Открыть). В некоторых браузерах этот пункт называется Open File (Открыть файл). Затем необходимо выбрать файл `script1.htm`. В некоторых браузерах для отображения диалогового окна открытия файла нужно сначала щелкнуть на кнопке Browse (Обзор).

Если все строки были введены согласно приведенному выше образцу, то документ в окне браузера должен выглядеть так, как показано на рис. 3.3 (с учетом небольших поправок на разные версии используемой операционной системы и браузера). Если же браузер выдает сообщение о том, что при загрузке документа имеет место ошибка, то на данном этапе ничего предпринимать по этому поводу не нужно. Если появилось диалоговое окно ошибки, просто щелкните на кнопке OK. Сначала лучше разобраться с тем, как устроен весь документ, понять особенности его выполнения и только после этого переходить к отладке.

Исследование сценария

Сразу следует предупредить: не старайтесь запомнить все команды или тот синтаксис, которые будут описываться в этом разделе. Напротив, немного расслабьтесь и посмотрите, как строки кода превращаются в то, что отображается в окне браузера. В листинге 3.1 все строки вплоть до дескриптора `<script>` являются примером кода обычного HTML. К нестандартному дескриптору относится только `<style>`, который входит в спецификацию CSS.

Дескриптор <script>

Каждый раз при включении в документ HTML переменных JavaScript соответствующие строки кода потребуется заключить между парой `<script>..</script>`. Эти дескрипторы сообщают программе браузера, что содержащийся между ними текст нужно интерпретировать как сценарий. Поскольку другие языки составления сценариев (такие, как VBScript компании Microsoft) также могут в полной мере использовать преимущества данных дескрипторов, то вам обязательно необходимо указать точное название того языка, который использован для создания кода. Поэтому, когда браузер получает сообщение о том, что в сценарии используется язык JavaScript, он использует для обработки кода встроенный интерпретатор JavaScript. Проведем такую аналогию с реальной жизнью: если у вас под рукой есть разговорник французского языка, то нужно, чтобы ваш собеседник тоже говорил по-французски. Если попутчик, скажем, из Испании, то французский разговорник вряд ли будет полезен. Точно так же: если используемый браузер имеет только интерпретатор JavaScript, он не сможет понять код, написанный на VBScript.

Далее мы остановимся на еще одном аспекте использования JavaScript, который важен при работе со сценариями. JavaScript чувствителен к регистру (различает большие и маленькие символы). Поэтому необходимо при вводе кода сценария JavaScript следить не только за самими терминами, но и за состоянием регистра, т.е. не путать прописные и строчные литеры. В дескрипторах HTML (включая и `<script>`) могут использоваться оба регистра по усмотрению разработчика, но все, что является кодом JavaScript, чувствительно к регистру. Если строка кода JavaScript не выполняется, сначала следует проверить, правильный ли регистр был использован для ее представления[1]. Всегда следует сравнивать введенный вручную код с теми листингами, которые приведены в книге или других источниках.

Сценарий для всех браузеров

Следующая после дескриптора `<script>` строка в листинге 3.1 напоминает HTML-дескриптор начала комментария. Так оно и есть, только интерпретатор JavaScript трактует дескрипторы комментариев особым образом. Хотя JavaScript послушно игнорирует строки, начинающиеся с дескриптора начала комментария HTML, он воспринимает следующую строку как полноправную строку сценария. Если нужно в код JavaScript вставить комментарий, то этот комментарий должен начинаться с двойной косой черты (`//`). Такой комментарий может располагаться в конце строки (например, после оператора JavaScript, который должен интерпретироваться браузером) или занимать отдельную строку. Как легко заметить, в конце сценария используется последний вариант. Строка комментария начинается с двух косых черт.

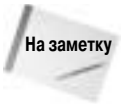
Если теперь внимательно присмотреться, то можно заметить, что внутри дескрипторов комментария стандартного HTML-кода (`<!--комментарий-->`) размещен целый сценарий

[1] *Стили XHTML, если вы планируете использовать их в своих приложениях, требуют введения имен дескрипторов и имен атрибутов в нижнем регистре. Это соглашение вы встретите повсеместно в настоящей книге.*

(с собственными комментариями). Смысл такой организации сценария сразу не совсем понятен. Но это до тех пор, пока вы не столкнетесь с браузером, который не поддерживает язык JavaScript. В этом случае браузер игнорирует дескриптор `<script>` как новый непонятный для него элемент. А вот текст сценария он интерпретирует как обычный текст, который нужно отображать на странице. Если заключить этот текст в дескрипторы комментария HTML, то большинство устаревших браузеров не отобразит строки сценария на странице.

Однако некоторые старые браузеры (далеко не все могут себе позволить обновлять программное обеспечение вовремя) иногда не реагируют на такое развитие событий, но выводят на экран непонятные сведения. Это происходит потому, что они интерпретируют отдельный символ > (а не весь символ `-->` как целое) в качестве символа окончания комментария.

Следует, кроме того, помнить, что некоторые пользователи просто не имеют доступа к современным браузерам (если, например, используют текстовые браузеры Lynx для UNIX или аналогичные Lynx-браузеры в переносных компьютерах). Если выделить строки сценария в качестве комментария, то при использовании относительно современных, но не совместимых с JavaScript браузеров, страница, все же, не будет выглядеть как некорректная.



Обратите внимание на то, что строки комментария, содержащие операторы, которые предотвращают выполнение сценариев в старых браузерах, расположены внутри дескрипторов `<script>..</script>`. Нельзя строки комментария выносить за пределы этих дескрипторов. Иначе этот метод не сработает.

И еще несколько замечаний о способах скрытия комментариев, описанных в этой книге. Чтобы сэкономить место, в большинстве примеров не были приведены комментарии. Но, как в этом можно убедиться на примере реальных приложений (главы 48–57, приведенные на прилагаемом к книге компакт-диске), комментарии в них присутствуют. Если же сценарий создается для общих нужд, следует всегда добавлять комментарии в код сценария.

Отображение текста

В сценарии листинга 3.1 есть две строки одного типа, которые используются для отображения текста в данном документе. Это строки `document.write()`. О самом же объекте `document` речь пойдет в главе 18.

Когда бы ни запрашивался объект (в данном случае это `document`) с целью выполнения определенного задания, после названия задания обязательно поставьте круглые скобки. В качестве примера можно привести оператор `write()` — JavaScript должен знать, с какими данным его нужно применять. После имени задания указываются эти данные, называемые также *параметрами*. Поэтому если в документе нужно отобразить имя первого президента Соединенных Штатов, то сделать это можно так.

```
document.write("George Washington")
```

Строка текста, которую сценарий выводит на экран, начинается с некоторого предопределенного фрагмента `"This browser is version"` *(Это браузер версии)*, в который впоследствии добавляется версия браузера. Далее на экран снова выводится статический текст, содержащий дескриптор HTML (`"of <span class='highlight'>"`), затем более значимый текст (название приложения), после чего — закрывающий дескриптор HTML и точка в конце предложения (`"</span>."`). В JavaScript символ “плюс” (+) используется для объединения (*конкатенации*) фрагментов текста в одну большую строку текстовых символов, которые отображаются в документе. Ни JavaScript, ни тем более символ + ничего не знают об используемых словах и пробелах, поэтому сценарий полностью отвечает за выделение нужного места, что и отражается в параметрах. Обратите внимание, что после слова `"version"` в параметре первого оператора `document.write()` оставлено свободное место. Также свободное место вы увидите с обеих сторон от `"of"` во втором операторе `document.write()`.

Для получения информации о версии браузера и названии, которые используются в параметрах, укажите JavaScript извлечь соответствующие свойства объекта `navigator`. Извлечь свойство можно, если добавить его через точку после имени объекта (в данном случае это `navigator`). Извлекаемые имена в приведенном примере разделены пробелом. Если вам тяжело сориентироваться в записи, то попробуйте прочитать ее справа налево. В этом случае то, что находится справа, является свойством расположенного слева. В нашем случае `appVersion` является свойством объекта `navigator`. Этот "точечный" синтаксис во многом напоминает вызов операторов действий (к примеру, `document.write()`), но только после имени свойства не нужно ставить круглые скобки. В любом случае, ссылка на свойство в сценарии дает указание JavaScript вставить значение этого свойства туда, откуда сделан запрос. При первом обращении JavaScript-сценарий запросит информацию о браузере, представит ее в виде текстовой строки и отобразит в документе.

Наконец, обратите внимание на точку с запятой в конце каждого оператора JavaScript. На самом деле вводить ее совсем не обязательно. Ничего плохого в последнем случае не произойдет. Тем не менее, если вы планируете изучать более серьезные языки программирования, например, C++, то использование точек с запятой в конце операторов будет обязательным требованием.

Улыбнемся вместе

Если при первой попытке загрузить в браузер данный документ произошла ошибка, следует вернуться в текстовый редактор и проверить строчка за строчкой сценарий, сравнив его с листингом 3.1. При этом помните о том, что было рассказано выше. Может оказаться, что один из символов расположен не на своем месте или вместо символа верхнего регистра введен символ нижнего регистра, или наоборот. В листинге могут отсутствовать кавычки или скобки.

Чтобы увидеть, насколько динамичен сценарий `script1.htm`, вернитесь в текстовый редактор и замените слово `"browser"` на `"client software"`. Теперь для отображения изменения в тексте документа нужно все сохранить, перейти в браузер и перезагрузить страницу. Вы можете смело заменять любой текст, приведенный в кавычках в виде параметра в операторе `document.write()`. Параметры оператора `document.write()` являются текстом HTML, поэтому в нем можно использовать дескриптор `"<br>"` для разрыва строк. При этом помните, что для отображения на экране любых внесенных изменений нужно сохранить документ, перейти в браузер, а затем перезагрузить сохраненную страницу.