Часть 1

Основы JavaScript

В этой части…

Глава 2

Первые шаги в JavaScript

В этой главе…

- Что нужно для написания сценариев
- Первый сценарий
- Тег <SCRIPT>
- Размещение сценариев
- Если браузер не поддерживает JavaScript
- Можно ли скрыть текст сценария
- Средства отладки сценариев

Теперь, после того как мы прояснили ситуацию в мире Web-технологий и выяснили, что из себя представляет JavaScript, можно приступать к написанию сценариев. В данной главе вы узнаете, какие средства понадобятся для написания документов HTML и сценариев, что вообще из себя представляют сценарии JavaScript, как они подключаются к документам HTML и многое другое.

Что нужно для написания сценариев

Для создания HTML-документов и сценариев JavaScript вам не понадобится никакого специфического программного обеспечения. На самом деле все, что нужно — текстовый редактор и браузер. Большинство операционных систем содержат обе эти программы в стандартном комплекте поставки.

Важным достоинством языка JavaScript, как средства разработки Web-приложений, является то, что для его изучения и написания сценариев не требуется подключения к Интернету. Но если в дальнейшем вы захотите опубликовать созданную страницу, то для этого обязательно понадобится подключение к Интернету для доступа к Web-серверу, на котором она будет размещена.

Системные требования

Выполнение сценариев не требует от компьютера значительных ресурсов. Для этого будет достаточно обыкновенного персонального компьютера с операционной системой

Windows, MacOS или Linux, дискового пространства, которого бы хватило для установки текстового редактора и браузера (или нескольких браузеров), а также объема оперативной памяти, позволяющего одновременно запустить все приложения — текстовый редактор и браузеры.

Выполнение сценариев JavaScript почти не зависит от операционной системы, поэтому большинство примеров из данной книги будут одинаково работать в Windows, MacOS или Linux, главное, чтобы для них существовали необходимые версии браузеров. Тем не менее для определенности будем считать, что на вашем компьютере установлена любая из операционных систем: Windows 98, Windows 2000 или Windows XP (хотя от конкретной версии зависят комбинации клавиш, команды меню и т.п.).

Текстовый редактор

Для создания HTML-документов и сценариев подойдет любой текстовый редактор. Если вы работаете в среде Windows, можно воспользоваться встроенным текстовым редактором Блокнот (Notepad). Преимущество его использования заключается в том, что необходимое программное обеспечение уже установлено на вашем компьютере.

Многие текстовые редакторы, например Microsoft Word, позволяют сохранять документ в различных форматах. В таких редакторах при сохранении нового HTML-документа следует всегда выбирать формат простого текста (Plain Text), т.е. файл будет записан с расширением `.txt` (рис. 2.1). Не следует сразу выбирать сохранение документа в виде Web-страницы или HTML-документа. Расширение `.htm` или `.html` можно ввести в поле для ввода имени файла или изменить вручную уже после того, как документ будет сохранен.

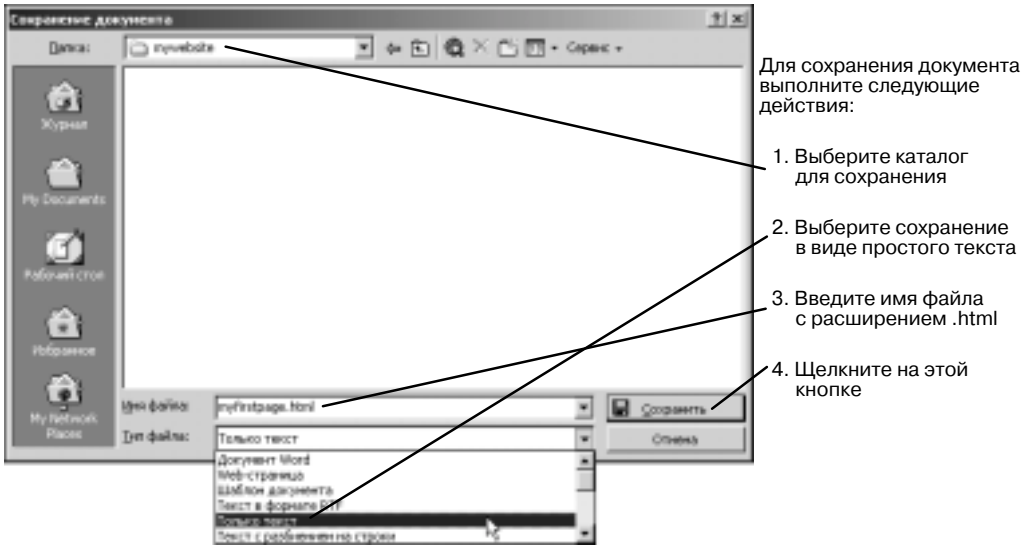

Рис. 2.1. *Сохранение документа в редакторе Microsoft Word*

Кроме того, еще существует множество различных специализированных текстовых редакторов, предназначенных именно для создания документов HTML со сценариями на JavaScript. Умелое использование таких редакторов позволяет значительно сэкономить время, затрачиваемое на разработку Web-страницы.

Еще раз следует предостеречь начинающих разработчиков от использования WYSIWYG-редакторов. Конечно, они позволяют создавать страницы предельно просто и быстро даже без знаний языка HTML. Но при этом:

- вы получаете весьма ограниченный доступ к возможностям JavaScript и CSS;
- размеры полученных документов могут в несколько раз превышать размеры аналогичных страниц, написанных вручную;
- вопреки названию данного метода, то, что вы увидите на своем экране при создании документа, большое количество посетителей вашего Web-сайта воспримет несколько иначе. Это связано с использованием ими различных типов и версий браузеров.

Браузер

Кроме текстового редактора вам понадобится еще одна программа — браузер для просмотра созданных HTML-документов и проверки работоспособности сценариев. Браузер является вашим окном в Web независимо от того, какими возможностями обладает язык JavaScript. А поэтому выбор браузера гораздо более важен, чем выбор текстового редактора. Ведь от текстового редактора зависит лишь удобство написания документов. В то же время от браузера, для которого эти документы написаны, в значительной мере зависит количество пользователей Web, которые смогут их прочесть, а следовательно — популярность будущего Web-ресурса. Теперь рассмотрим несколько общих рекомендаций при выборе браузера.

- При создании страниц для Web следует учесть, что их будут просматривать обладатели различных браузеров. Поэтому желательно установить на своем компьютере не один браузер, а несколько — из числа наиболее популярных. Еще недавно несомненными лидерами среди браузеров были Microsoft Internet Explorer и Netscape Navigator. Однако сегодня при создании Web-страниц нельзя не учитывать и все возрастающее количество пользователей таких браузеров, как Opera и Mozilla. Если созданные вами Web-страницы будут адаптированы для вышеозначенных браузеров, вы можете быть уверены, что почти все пользователи Web смогут их прочесть.
- При установке браузеров не забывайте, что весьма проблематично инсталлировать на одном компьютере несколько версий браузера одного типа. Поэтому лучше всего установить последние версии браузеров. Это даст возможность использовать на своих страницах последние достижения Web-технологий. С другой стороны, не имея достаточного опыта Web-разработки, вы не будете знать, как работает написанный сценарий в более старых, но, возможно, еще популярных версиях того или иного браузера. Наилучшее решение данной проблемы — проверять страницу на других компьютерах, где установлены другие версии браузеров.

Большая часть примеров, представленных в данной книге, будет одинаково выполняться во всех браузерах, поддерживающих стандарт W3C DOM. Исключение составят лишь некоторые примеры из главы 12, где будут рассмотрены различия между браузерами. В большинстве случаев для работы будет достаточно браузера Internet Explorer 5.x или 6, который, скорее всего, уже установлен на вашем компьютере вместе с операционной системой Windows. Однако весьма желательно загрузить и установить также последние версии браузеров Mozilla и Opera, так как это поможет научиться писать сценарии, наиболее полно использующие возможности различных браузеров.

Мы рекомендуем установить браузер Mozilla, а не Netscape Navigator. Это обусловлено тем, что Mozilla больше ориентирован на разработчиков (хотя оба браузера одинаково отображают Web-страницы). Следует также отметить, что браузер Mozilla содержит несколько полезных утилит и настроек, отсутствующих в браузере Netscape Navigator.

Популярность браузеров в Web

Существует множество Web-сайтов, где можно получить информацию о том, какие браузеры чаще всего используются в Web. Например, можно найти подробную статистику применения браузеров на страницах:

```
http://www.w3schools.com/browsers/browsers_stats.asp
http://www.artlebedev.ru/tools/browsers
```

На этих страницах можно отыскать и много другой полезной статистической информации: используемые операционные системы, разрешения экрана и др. Однако такие статистические данные следует воспринимать с большой осторожностью. Подтверждение тому — данные с различных страниц, которые могут весьма существенно отличаться. Это связано в первую очередь с тем, что каждый Web-сайт привлекает различную аудиторию, в то время как статистика набирается из отчетов посещений небольшого числа Web-сайтов, а то и вовсе одного. Кроме того, программы, анализирующие данные о посещениях, не всегда правильно определяют применяемый браузер.

Поэтому наилучшее решение — после публикации Web-страницы в Интернете следить за тем, какие браузеры используют *именно ее посетители*. Данную информацию (т.е. место для размещения Web-страницы и доступ к серверу) можно получить у компании, предоставляющей услуги хостинга.

В некоторых случаях в книге будут даны рекомендации по вопросам реализации того или иного сценария для более ранних версий браузеров. Конечно, это возможно далеко не для всех сценариев, и если вы посмотрите статистику использования браузеров по указанным выше адресам, то поймете, что большой необходимости включать поддержку таких браузеров в свои сценарии сейчас нет.

Последние версии всех упомянутых браузеров — Internet Explorer, Netscape Navigator, Opera и Mozilla — можно бесплатно загрузить из Интернета с Web-сайтов компаний производителей.

Для того чтобы загрузить браузер Internet Explorer, посетите адрес:

```
http://www.microsoft.com/windows/ie/default.html
```

Для того чтобы загрузить браузер Netscape Navigator, посетите адрес:

```
http://www.netscape.com/download/index.html
```

Для того чтобы загрузить браузер Mozilla, посетите адрес:

```
http://mozilla.org/products/index.html
```

Для того чтобы загрузить браузер Opera, посетите адрес:

```
http://opera.com/downloads/index.html
```

Во время разработки документов HTML и сценариев вам очень часто придется выполнять одну и ту же утомительную последовательность действий: сохранить файл в окне текстового редактора, переключиться в окно браузера, перезагрузить страницу в браузере (при условии, что в текстовом редакторе и браузере открыт один и тот же документ). Для ускорения данной процедуры можно воспользоваться следующими комбинациями клавиш:

<Ctrl+S> — для сохранения документа (используется почти во всех текстовых редакторах);

<Alt+Tab> — для переключения между окнам;

<Ctrl+R> — для перезагрузки страницы в любом браузере.

Первый сценарий

Итак, будем надеяться, что все подготовительные вопросы исчерпаны, а значит, пришло время переходить к действию — написанию нашего первого сценария JavaScript. Причем для того, чтобы не загружать читателей лишней на данном этапе информацией,

следует отметить, что это будет единственный сценарий в настоящей главе. В последующих разделах книги мы будем постоянно на него ссылаться и вносить в него изменения.

Для того чтобы написать первый сценарий JavaScript и увидеть результаты его работы, выполните такую последовательность действий:

1. Запустите текстовый редактор и создайте в нем новый пустой документ.
2. Введите текст HTML-документа, содержащий ваш первый сценарий JavaScript (листинг 2.1). При вводе операторов сценария будьте очень внимательны и помните, что JavaScript чувствителен к регистру символов. Кроме того, во избежание путаницы на данном этапе пока не будет описываться смысл каждого из вводимых операторов JavaScript.
3. Сохраните созданный документ с именем, например, `script1.html` (не забудьте о том, что говорилось о сохранении файлов в начале данной главы).
4. Запустите браузер.
5. Откройте в окне браузера созданный файл `script1.html`. Для этого в любом браузере следует выбрать команду меню Файл⇨Открыть (File⇨Open).

Листинг 2.1. Первый сценарий

```
<HTML>
<HEAD>
 <TITLE>Самый первый сценарий</TITLE>
</HEAD>
<BODY>
Это обычный текст HTML, а ниже вы увидите
текст, созданный с помощью сценария.
<BR>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
 var d = new Date()
 var year = d.getFullYear()
 document.write('Сейчас на дворе <STRONG>', year, '</STRONG> год.')
</SCRIPT>
<BR>
А здесь еще немного обычного текста.
</BODY>
</HTML>
```

Данный пример содержит стандартные элементы, присущие любому HTML-документу (теги — <HTML>, <HEAD>, <TITLE>, <BODY>), а также немного обычного текста и один сценарий JavaScript.

Этот сценарий (листинг 2.1) будет одинаково работать в любом браузере, поддерживающем JavaScript. На рис. 2.2 представлен результат его выполнения в окне браузера Internet Explorer 6. Как вы уже догадались, ваш первый сценарий определяет текущий год и отображает полученное значение в теле документа.

При использовании Netscape Navigator все настройки выполняются точно так же, как это описано для браузера Mozilla. Однако если вы применяете одну из других реализаций браузера Mozilla, например Mozilla Firefox, то структура меню и настроек будет отличаться от описанных.

Рис. 2.2. *Первый сценарий, выполненный в окне браузера Internet Explorer 6*

Включение поддержки JavaScript в браузерах

Если в окне браузера не появился результат работы данного сценария, это может означать, что выполнение сценариев в вашем браузере отключено.

Для того чтобы включить выполнение сценариев в браузере Mozilla, следует выбрать команду меню Edit⇨Preferences (Правка⇨Параметры). В открывшемся окне в списке Category (Категория) выберите категорию Scripts&Plugins (Сценарии и надстройки) из вложенного списка Advanced (Дополнительно). Далее справа в этом же окне установите флажок Navigator под заголовком Enable JavaScript for.

В браузерах Internet Explorer выполнение сценариев может быть отключено лишь для документов в Интернете или локальной сети, но не для документов, расположенных на вашем компьютере. Для того чтобы включить поддержку JavaScript для внешних документов, выберите команду меню Сервис⇨Свойства Обозревателя (Tools⇨Internet Options). В открывшемся окне выберите вкладку Безопасность (Security) и на ней пиктограмму Интернета (или Local Intranet для локальной сети). Далее проще всего щелкнуть на кнопке По умолчанию (Default Level), и все станет на свои места. Но можно щелкнуть и на кнопке Другой (Custom Level) и в раскрывшемся списке установить все три переключателя в группе Сценарии (Scripting) в положение Разрешить (Enable).

Если в окне браузера вы получили то же, что изображено на рис. 2.2 (с поправкой на текущий год), можно переходить к анализу написанного сценария. Этому будут посвящены несколько следующих разделов, но кое-что следует отметить сразу.

- Для включения сценариев JavaScript в HTML-документ служит тег `<SCRIPT>`. При его обнаружении браузер с поддержкой JavaScript понимает, что весь текст, заключенный между тегами `<SCRIPT>` и `</SCRIPT>`, не должен отображаться на экране, а должен интерпретироваться как текст сценария. Иначе говоря, сценарии JavaScript интегрируются в HTML-документы посредством тега `<SCRIPT>`. HTML-документы могут содержать много тегов `<SCRIPT>`, и каждая пара может включать в себя несколько наборов операторов JavaScript.
- Сценарий может генерировать содержимое HTML-страницы. Самый простой способ сделать это — воспользоваться функцией `document.write()`. В окне браузера текст, созданный таким способом, ничем не отличается от остального содержимого документа. Также следует заметить, что данный текст на странице будет расположен именно в том месте документа, где вставлен соответствующий контейнер `<SCRIPT>`.

- Для работы с объектами в JavaScript используется так называемый *точечный* синтаксис. Это означает, что можно обратиться к методу `write()` объекта `document` с помощью конструкции `document.write()`. Более подробно об объектах и о точечном синтаксисе речь пойдет в главе 4, "Объекты языка JavaScript".
- Здесь используется один из объектов DOM — `document`. Мы будем и далее применять в сценариях объекты DOM, хотя к их детальному описанию приступим лишь во второй части книги. Это связано с тем, что понимание их работы требует предварительного знакомства с базовым языком JavaScript, но без их использования невозможен ни один действительно полезный сценарий.
- Язык JavaScript, в отличие от HTML (как отмечалось выше), чувствителен к регистру символов. Это означает, что в сценариях при вводе всех переменных, операторов, ключевых слов, ссылок на объекты и т.д. нужно следить за тем, какие буквы должны быть прописными, а какие — строчными. Например, если вместо `document.write()`ввести `document.Write()`, сценарий сразу же перестанет выполняться.

Тег <SCRIPT>

То, как сценарии включаются в HTML-документ с помощью тега `<SCRIPT>`, необходимо рассмотреть подробнее. Этот тег содержит ряд важных атрибутов, предназначенных для указания языка сценария и подключения внешних файлов сценариев.

Указание языка сценария

Поскольку для написания сценариев предназначены различные языки, то в открывающем теге `<SCRIPT>` должен быть обязательно указан используемый язык. И хотя большинство современных браузеров автоматически определяют тип применяемого языка, не стоит этим злоупотреблять. В листинге 2.1 тег `<SCRIPT>`имеет следующий вид:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
```

Оба указанных атрибута `LANGUAGE` и `TYPE` служат для определения языка, на котором написан сценарий. Спрашивается, зачем использовать целых два разных атрибута для указания на одно и то же?

Все браузеры, поддерживающие выполнение сценариев, начиная с Netscape Navigator 2 и Internet Explorer 3, признают атрибут указания языка `LANGUAGE`. Кроме значения `JavaScript`, в данном атрибуте возможны также значения, явно указывающие используемую версию языка: `JavaScript1.1`, `JavaScript1.2` и т.д. Конечно, такое явное определение версии не добавит вашему сценарию дополнительных возможностей. Единственная причина, по которой разработчики применяют данную возможность — предотвратить запуск сценариев, содержащих новейшие средства, в старых браузерах. То есть если сценарий начинается с тега `<SCRIPT LANGUAGE="JavaScript1.2">`, то браузеры, которые не поддерживают язык JavaScript 1.2, даже не будут пытаться интерпретировать данный сценарий (о различных версиях языка JavaScript речь шла в предыдущей главе). Такой способ выделения нужных браузеров обладает рядом недостатков.

- Как уже упоминалось, основные отличия при выполнении сценариев различными браузерами заключаются не в языке JavaScript, а в моделях DOM. Яркий пример — браузеры Internet Explorer 4 и Netscape Navigator 4. Они поддерживают язык JavaScript 1.1, но при этом невозможно написать серьезный сценарий, который выполнялся бы в обоих браузерах.

- Только браузеры Netscape Navigator и Mozilla поддерживают те или иные версии JavaScript "в чистом виде". Что касается остальных браузеров (в первую очередь Internet Explorer), то далеко не всегда можно сказать, какая версия JavaScript полностью поддерживается в той или иной версии этого браузера. Например, браузеры Internet Explorer 5.x поддерживают в полной мере лишь JavaScript 1.3, а версии 1.4 и 1.5 лишь частично. При этом данные браузеры всегда будут пытаться интерпретировать сценарии, в которых указан атрибут `LANGUAGE="JavaScript1.5"`.

Суммируя все вышесказанное, можно сделать следующий вывод — если вы используете для определения языка атрибут `LANGUAGE`, лучше всего указывать в нем значение `JavaScript`, а разделение сценария между браузерами проводить программными методами, о которых будет рассказано далее.

Для браузеров Internet Explorer в атрибуте `LANGUAGE` также можно указывать и другие поддерживаемые языки: `JScript` и `VBScript`.

В стандарте HTML 4.0 атрибут `LANGUAGE` считается устаревшим и использовать его не рекомендуется. Вместо него введен новый атрибут `TYPE`, в котором язык указывается в виде стандартного типа MIME. Для определения сценариев JavaScript служит тип `text/javascript`. Таким образом, атрибут `TYPE` обрабатывают только браузеры с поддержкой HTML 4.0, но атрибут `LANGUAGE` могут перестать поддерживать новые браузеры. Поэтому может быть полезным использование обоих атрибутов для обеспечения более полной совместимости.

Тот факт, что в атрибуте `TYPE` нельзя указать версию языка, еще раз свидетельствует о том, что данная возможность считается излишней, и производители браузеров собираются от нее отказаться.

Типы MIME

Типы MIME (Multipurpose Internet Mail Extension — многоцелевые расширения электронной почты) служат для идентификации любых типов данных, передаваемых по Интернету. Это нужно для того, чтобы программа-клиент, например, браузер, всегда заранее "знала", какие данные сейчас поступят к ней с сервера и интерпретировала их соответствующим образом. Типы MIME существуют для всех разновидностей файлов, распространяемых через Интернет. Они всегда имеют структуру "тип/подтип". Тип указывает общий тип данных, а подтип — их конкретный формат. Например, тип `image/jpeg` соответствует изображению в формате JPEG, а `text/html` — HTML-тексту.

Подключение сценария из внешнего файла

Кроме размещения текста сценария в теле HTML-документа его можно поместить в отдельный файл. Это должен быть обычный текстовый файл, но обязательно с расширением `.js`. Для его создания можно воспользоваться тем же текстовым редактором, который используется для написания HTML-документов. В этом файле не должно быть ничего кроме операторов JavaScript. Например, для того чтобы вынести сценарий из листинга 2.1 в отдельный файл, нужно будет выполнить следующие действия.

1. Запустите текстовый редактор и создайте новый текстовый файл.
2. Введите только текст сценария (не нужно помещать туда тег `<SCRIPT>` или другие теги):

```
var d=new Date()
var year=d.getFullYear()
document.write('Сейчас на дворе <STRONG>',year,'</STRONG> год.')
```

3. Сохраните сценарий в виде текстового файла `myscript.js` в том же каталоге, где находится HTML-документ, содержащий листинг 2.1 (такое имя файла и его расположение не являются обязательными, а выбраны исключительно для определенности и удобства).
4. В тексте HTML-документа содержимое контейнера `<SCRIPT>` следует привести к следующему виду:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript"
 SRC="myscript.js">
</SCRIPT>
```

5. Сохраните полученный файл с другим именем, например `script2.html`,так как файл `script1.html` нам еще понадобится.

Все, что мы сделали с тегом `<SCRIPT>` — это вытерли все строки сценария и добавили новый атрибут `SRC`. Именно он и предназначен для подключения внешних файлов сценариев. Вообще говоря, значением данного атрибута должен быть URL-адрес файла сценария. При ссылках на файлы в пределах одного Web-сайта, а также если все файлы находятся на диске вашего компьютера, вместо адреса URL можно указывать относительный путь к файлу сценария. А это означает:

- если файл сценария находится в том же каталоге, что и HTML-документ, в атрибуте `SRC` можно просто указать имя файла, как это сделано выше;
- если файл расположен во вложенном каталоге, например `/scripts`, то значение атрибута `SRC` должно представлять собой относительный путь к требуемому файлу сценария — в данном случае это будет `scripts/myscript.js`;
- если файл сценария помещен в родительский каталог, то значение атрибута `SRC` должно представлять собой относительный путь к файлу сценария — в нашем случае это будет `../myscript.js`.

После внесения указанных изменений документ будет работать так же, как раньше. Это означает, что если сценарий подключается из внешнего файла, для браузера это равносильно размещению файла внутри контейнера `<SCRIPT>`.

Может возникнуть вопрос: а если в теге `<SCRIPT>` использован атрибут `SRC`, а между тегами `<SCRIPT>` и `</SCRIPT>` также содержатся строки сценария? В таком случае будет выполнен лишь внешний сценарий, а строки сценария, заключенные между `<SCRIPT>` и `</SCRIPT>`, будут полностью проигнорированы.

Вынесение кода сценариев в отдельный файл обладает во многих случаях рядом преимуществ по сравнению с его непосредственным включением в HTML-документ.

- Если один и тот же сценарий используется в нескольких документах, то его вынесение в отдельный файл позволит уменьшить размер этих документов, а значит — ускорить процесс их загрузки с сервера для пользователей вашего Web-сайта. Кроме того, если вам понадобится внести в сценарий те или иные изменения, это придется сделать всего лишь один раз — в коде сценария отдельного файла.
- Если в текст большого и сложного HTML-документа включен большой и сложный сценарий, то процессы отладки и внесения изменений могут превратиться в сущий ад.

В этом случае вынесение сценария в отдельный файл (или даже несколько файлов) может значительно упростить и ускорить поиск нужных фрагментов кода.

- Подключая внешний файл сценария, вы можете быть уверены, что он не будет неправильно интерпретирован браузерами, не поддерживающими JavaScript. Подробнее об этом речь пойдет далее в настоящей главе.

Конечно, при работе с небольшим сценарием, используемым всего в одном документе, выносить его в отдельный файл не имеет смысла. Это только создаст лишние неудобства при редактировании.

Размещение сценариев

Является ли сценарий частью HTML-документа или импортируется из внешнего файла, под его *размещением* понимается *расположение контейнера* <SCRIPT> в документе. Этот контейнер может быть расположен как в заголовке документа (между тегами <HEAD> и </HEAD>), так и в его теле (между тегами <BODY> и </BODY>).

Один документ может содержать произвольное количество сценариев. Причем они не будут изолированы друг от друга. Все переменные, функции и объекты, созданные в одном из контейнеров <SCRIPT>, будут доступны и во всех остальных сценариях, расположенных далее в документе.

Следует заметить, что исходя из способа запуска сценарии можно разделить на два вида: сценарии, которые выполняются во время загрузки браузером содержимого HTML-документа (к ним относится и сценарий из листинга 2.1), и *отсроченные сценарии*, которые выполняются уже после загрузки документа в браузер при возникновении того или иного события или через определенный промежуток времени. Общая идеология размещения сценариев таких видов различается кардинальным образом.

Для сценариев, выполняющихся непосредственно во время обработки содержимого документа, можно сформулировать следующие правила размещения:

- если сценарий должен выводить в теле документа текст, то его следует размещать именно в том месте, где должен находиться данный текст;
- если сценарий использует те или иные объекты документа, то он должен располагаться в документе ниже, чем теги, создающие эти объекты.

Сценарии второго вида — отсроченные, служащие для обработки событий, обычно имеют вид функций (о функциях подробно будет рассказано в следующей главе). В программировании считается "хорошим тоном" размещать такие сценарии в заголовке документа (в контейнере <HEAD>). Такое местонахождение прежде всего удобно, так как при меньшем объеме тела документа его легче редактировать, но в случае очень больших HTML-документов оно может быть действительно полезным. Например, пользователь может щелкнуть на кнопке еще до того, как весь документ будет загружен.

Однако в большинстве случаев данные функции могут быть определены в любой части документа, так как весь небольшой документ будет проанализирован до того, как возникнет необходимость в обработке событий и вызове соответствующих функций.

Если сценарий написан без ошибок, но почему-то не работает, это может быть связано с его неправильным размещением в документе. Попробуйте перенести весь сценарий в самый конец тела документа (непосредственно перед тегом </BODY>). Если и после этого он не заработает, то следует внимательнее поискать ошибку. Если же он заработает, но не так, как хотелось бы, придется искать другое решение поставленной задачи.

Если браузер не поддерживает JavaScript

На сегодняшний день все еще существуют браузеры, вообще не поддерживающие выполнения сценариев на языке JavaScript. Кроме того, у некоторых пользователей современных браузеров поддержка JavaScript может быть просто отключена. При создании Web-страниц не принимать во внимание подобные факторы просто нельзя.

Для того чтобы увидеть, как выглядит страница в браузере, не поддерживающем сценарии, вовсе не обязательно искать и устанавливать на своем компьютере такой браузер. Для этого достаточно просто удалить теги `<SCRIPT>` и `</SCRIPT>`. Еще быстрее будет изменить слово `SCRIPT` таким образом, чтобы браузер его не распознал. Например, можно стереть последнюю букву и получить несуществующий тег `<SCRIP>`. Подобным образом можно поступать с любым тегом, если необходимо иметь представление, как выглядит страница в браузерах, которые этот тег не поддерживают.

Что же случится при попытке запустить пример из листинга 2.1 в браузере, не поддерживающем выполнение сценариев? Если браузер не знает тега `<SCRIPT>`, он его просто проигнорирует, и в окне браузера вы получите весь текст сценария. Конечно, это вовсе не то, чего бы нам хотелось. А хотелось бы получить удовлетворительный вид страницы в любых браузерах. Для этого придется решить две задачи.

- Необходимо, чтобы текст сценария был полностью проигнорирован браузером, который не может его выполнить. Для решения данной задачи предлагается два способа.
  - Первый способ (он уже упоминался ранее) — это вынесение сценария в отдельный файл. В результате контейнер `<SCRIPT>` станет пустым, и, следовательно, на экран ничего лишнего выведено не будет.
  - Другой, наиболее распространенный способ скрытия сценария от старых браузеров, — заключение всего текста сценария в комментарий HTML-документа. Любой текст, находящийся в HTML-документе между парой тегов "`<!--`" и "`-->`", называется *комментарием* и не отображается браузерами. После внесения описанных изменений сценарий из листинга 2.1 будет выглядеть следующим образом:

    ```
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
    var d=new Date()
     var year=d.getFullYear()
     document.write('Сейчас на дворе <STRONG>',year,'</STRONG> год.')
    // -->
    </SCRIPT>
    ```

    Интерпретаторы JavaScript обычно игнорируют символы HTML-комментария и без проблем выполняют такой сценарий. Однако перед символом окончания HTML-комментария также следует ставить символ комментария, но уже языка JavaScript (`//`) — поскольку некоторые интерпретаторы все же могут его неправильно трактовать.
- Также хотелось бы, чтобы вместо результата работы сценария в браузере, не поддерживающем их выполнение, отображался определенный текст. Причем этот текст не должны воспроизводить браузеры, поддерживающие выполнение сценариев. Для решения данной задачи тоже предусмотрено два различных способа.
  - Можно ввести нужный текст в последней строке сценария после символов комментария, например:

    ```
    // --> Ваш браузер не поддерживает выполнение сценариев.
    ```

В этом случае интерпретатор JavaScript его не воспримет, поскольку для него этот текст находится в строке комментария. В то же время для браузеров, не поддерживающих выполнение сценариев, данный текст будет расположен *снаружи* от комментария HTML-документа и поэтому должен быть отображен на экране. Недостатком такого способа является то, что введенный текст не будет воспроизведен браузерами, в которых отключено выполнение сценариев.

- В браузерах, поддерживающих выполнение сценариев, предусмотрен специальный тег `<NOSCRIPT>`. Данный контейнер обычно помещается сразу после закрывающего тега `</SCRIPT>`. Если браузер не поддерживает выполнение сценариев, то он не распознает и тег `<NOSCRIPT>`, а следовательно, в любом случае выведет в окне его содержимое. Если же браузер распознает тег `<SCRIPT>`, то текст контейнера `<NOSCRIPT>` будет отображен только в том случае, если в браузере выполнение сценариев отключено.

В стандарте HTML 4.0 браузеры обязаны отображать содержимое тега `<NOSCRIPT>` даже тогда, когда указанный язык сценария ими не поддерживается. Однако из рассматриваемых нами браузеров это требование реализовано лишь в Opera 7.x.

Можно ли скрыть текст сценария

В любом браузере при необходимости можно просмотреть HTML-текст открытой страницы. В браузере Internet Explorer для просмотра исходного кода страницы нужно воспользоваться командой меню Вид⇨Исходный текст (View⇨Source) или же командой из контекстного меню (контекстное меню можно вызвать щелчком правой кнопки мыши в любом месте страницы). В других браузерах это делается похожим образом.

Сценарии JavaScript можно считать программными продуктами, а написав действительно хороший сценарий, у вас может возникнуть желание спрятать его исходный код от желающих воспользоваться им на своей странице. Позвольте вас разочаровать — это сделать невозможно. Можно лишь усложнить получение исходного кода сценария или сделать его неудобочитаемым.

Если вы, например, вынесете текст сценария в отдельный файл, то его, конечно, нельзя будет прочитать описанным выше способом. Но зато можно прочитать его URL-адрес, а затем ввести этот адрес в адресной строке браузера и сохранить загруженный файл с помощью команды меню Файл⇨Сохранить как (File⇨Save As). Можно также с помощью сценария заблокировать вызов контекстного меню по щелчку правой кнопкой мыши, но в таком случае все равно можно воспользоваться командой меню Вид для просмотра исходного кода страницы. Вообще говоря, любую уловку можно обойти, поэтому лучше к ним даже не прибегать. Ведь, в конце концов, вы тоже можете прочитать любой сценарий в Web и использовать его на своей странице либо почерпнуть из него новые идеи для собственных изысканий.

Средства отладки сценариев

При создании большого и сложного сценария неизбежно возникают ошибки, которые непросто найти и исправить. Для многих языков программирования существуют специальные средства разработки, содержащие многочисленные функции отладки, — т.е. отслеживания и выявления ошибок. Поскольку для написания сценариев используется текстовый редактор, то здесь ни о каких средствах отладки не может быть и речи.

Однако все рассматриваемые браузеры, а в особенности Mozilla, способны предоставить некоторые средства отладки сценариев.

Для того чтобы включить средства отладки сценариев в браузере Internet Explorer, нужно выбрать команду меню Сервис⇨Свойства Обозревателя (Tools⇨Internet Options). Далее следует перейти на вкладку Дополнительно (Advanced) и установить флажок Выводить сообщение о каждой ошибке сценария (Display a notification about every script error). После этого, если сценарий содержит ошибку, при попытке его запуска (будь то при загрузке страницы или при обработке события) будет выведено соответствующее окно сообщения. На рис. 2.3 приведено окно сообщения, которое выдаст браузер, если в сценарии из листинга 2.1 вместо `document.write` ввести `document.Write`.

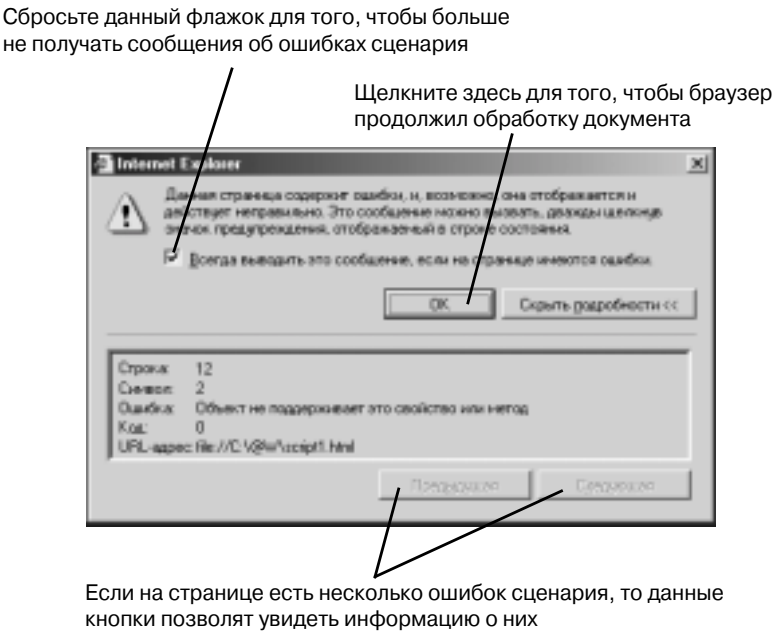

Рис. 2.3. *Сообщение об ошибке сценария в браузере Internet Explor 6*

Как видите, в нижней части данного окна отображена весьма полезная информация об обнаруженной ошибке. Здесь можно увидеть номер строки, в которой обнаружена ошибка — поле Строка (Line), номер символа в строке, с которого начинается ошибочный оператор — поле Символ (Char), и описание ошибки — поле Ошибка (Error). Такое окно сообщения будет выведено для каждой ошибки сценария на странице. Хотя данный способ и поможет определить положение всех ошибок сценария, тем не менее всегда будьте внимательны. Ведь реальная ошибка могла быть допущена до того, как это привело к нарушению работы сценария.

Если в окне сообщения об ошибке не отображена подробная информация о ней, то в нем следует щелкнуть на кнопке Показать детали (Show Details).

В браузерах Netscape Navigator, Opera и Mozilla реализовано другое средство отслеживания ошибок сценариев. Оно называется JavaScript Console (Консоль JavaScript). Для его вызова в браузерах Mozilla и Navigator достаточно набрать в строке адреса — `javascript:`

(не забудьте поставить в конце двоеточие). На рис. 2.4 изображено окно консоли JavaScript при обнаружении той же ошибки, что была описана выше. Здесь также можно увидеть номер сроки, в которой произошла ошибка (поле Line), и ее описание (поле Error). Консоль JavaScript сохраняет информацию обо всех ошибках, обнаруженных в текущем сеансе работы с браузером. Для того чтобы очистить окно консоли, следует щелкнуть на кнопке Clear (Очистить).

Рис. 2.4. *Окно консоли JavaScript в браузере Mozilla*

Последние версии браузера Mozilla предоставляют кроме описанной консоли JavaScript еще два более мощных средства для отладки сценариев — DOM Inspector и JavaScript Debugger. Для запуска этих средств отладки, а также консоли JavaScript можно воспользоваться подменю Tools⇨Web Development.

Утилита DOM Inspector (Инспектор DOM) предназначена для просмотра объектной структуры HTML-документа. Кроме того, с ее помощью при необходимости можно мгновенно узнать все свойства любого объекта DOM в загруженном документе. Описание DOM Inspector можно найти по адресу:

```
http://www.mozilla.org/projects/inspector/
```

Утилита JavaScript Debugger (отладчик JavaScript), также называемая Venkman — это многофункциональный отладчик сценариев, сравнимый по функциональности с отладчиками таких серьезных средств разработки, как Microsoft Visual Studio. Описание отладчика Venkman ищите по адресу:

```
http://www.mozilla.org/projects/venkman/
```

Обе утилиты вполне заслуживают детального рассмотрения, но ограниченный объем данной книги не позволяет это сделать — в главе 11 приводится лишь краткое описание типовых способов их использования. Основную информацию по работе с данными утилитами вы можете найти по указанным адресам, а детальной документации для них на момент написания настоящей книги не существовало вовсе.

Резюме

В данной главе были рассмотрены основы самого процесса написания сценариев. В первую очередь было указано, какое для этого понадобится программное обеспечение — текстовый редактор и браузер. Также были описаны основные принципы выбора этих программ.

Далее был написан наш первый сценарий JavaScript, анализируя который мы узнали о существовании тега <SCRIPT>, используемого для включения сценариев в HTML-документ.

Этот тег может располагаться как в заголовке, так и в теле документа, причем его местонахождение важно только для тех сценариев, которые выполняются во время обработки документа. Также были рассмотрены основные атрибуты данного тега и их назначение:

- `LANGUAGE` — язык сценария (может быть также указана версия языка);
- `TYPE` — также язык сценария, но в формате типов `MIME`;
- `SRC` — адрес внешнего файла сценария. Такой файл должен обязательно иметь расширение `.js`.

Текст сценария может быть скрыт от браузеров, которые не поддерживают их выполнение, с помощью символов комментария HTML-документа или путем вынесения сценария в отдельный файл. Но при этом невозможно скрыть текст сценариев от просмотра посетителями вашей Web-страницы. С помощью тега `<NOSCRIPT>` можно также заставить браузеры, не выполняющие сценарий, выводить на его месте заданный текст.

В последнем разделе было описано, как воспользоваться средствами отладки сценариев в различных браузерах. Следует заметить, что браузер Internet Explorer хотя и является наиболее популярным среди пользователей Web, содержит весьма слабые средства отладки. А наиболее мощными средствами отладки обладает браузер Mozilla, который и ориентирован, в основном, на разработчиков.

Тесты

Эти тесты помогут закрепить материал данной главы. Ответы ищите в приложении А.

Найдите правильный ответ

Некоторые из предложенных вопросов могут иметь несколько правильных ответов.

1. Какие программы понадобятся для написания сценариев JavaScript:
   - **а)** браузер;
   - **б)** JavaScript;
   - **в)** текстовый редактор;
   - **г)** WYSIWYG-редактор.
2. Какие атрибуты определяют язык сценария:
   - **а)** `LANGUAGE`;
   - **б)** `SCRIPT`;
   - **в)** `SRC`;
   - **г)** `TYPE`;
   - **д)** `NOSCRIPT`.
3. В каком случае браузер не выводит содержимое контейнера `<NOSCRIPT>`:
   - **а)** если браузер не поддерживает выполнение сценариев;
   - **б)** если в браузере отключено выполнение сценариев;
   - **в)** если в браузере включено выполнение сценариев;
   - **г)** если сценарий загружается из внешнего файла.

4. Каким образом можно полностью защитить исходный код сценария от просмотра:
   а) с помощью комментария HTML;
   б) с помощью комментария JavaScript;
   в) нужно вынести текст сценария в отдельный файл;
   г) это невозможно.

Правда или ложь?

Каждое утверждение либо верно, либо нет.

5. Можно подключить к документу внешний файл сценария.
6. Все сценарии выполняются браузером во время обработки документа.
7. Браузеры содержат средства отладки сценариев.
8. Консоль JavaScript — это средство для разработки сценариев JavaScript.

Практикум

Чтобы лучше усвоить навыки, полученные в данной главе, выполните предложенные практические задания.

9. Перенесите сценарий в листинге 2.1 в другие части документа. Посмотрите, изменится ли при этом выполнение сценария и как преобразится вид документа.
10. Измените сценарий так, чтобы от содержал различные ошибки (неправильный регистр, незакрытые скобки или кавычки и т.д.), и посмотрите, как работают средства отладки в различных браузерах. Особое внимание обратите на описание ошибок (значения поля error).