АЛЕКСАНДР КЛИМОВ

JavaScript

2-е издание

НА ПРИМЕРАХ

- ОБЩИЕ РЕКОМЕНДАЦИИ ПРИ СОЗДАНИИ СЦЕНАРИЯ
- РАБОТА С ОБЪЕКТАМИ И ЭЛЕМЕНТАМИ ДОКУМЕНТА, ИЗОБРАЖЕНИЯМИ, ДАТОЙ И ВРЕМЕНЕМ, СТРОКАМИ
- СПЕЦЭФФЕКТЫ ДЛЯ ИЗОБРАЖЕНИЙ И ТЕКСТА
- ШУТОЧНЫЕ ПРИМЕРЫ И ИГРЫ НА JavaScript
- СОЗДАНИЕ ИНТЕРАКТИВНЫХ Web-СТРАНИЦ
- РАСШИРЕНИЯ ДЛЯ БРАУЗЕРОВ
- ОБЛАСТИ ПРИМЕНЕНИЯ JavaScript
- СОВЕТЫ И ХИТРОСТИ ПРИ СОЗДАНИИ СЦЕНАРИЕВ

СКАЧАЙ ФАЙЛЫ ПРИМЕРОВ www.bhv.ru

bhv®

Александр Климов

JavaScript

НА ПРИМЕРАХ

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.06
ББК 32.973.26-018.2
К49

Климов А. П.

К49 JavaScript на примерах. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009. — 336 с.: ил.

ISBN 978-5-9775-0361-7

На примерах рассмотрены методы разработки сценариев на языке JavaScript. Представлены нестандартные приемы работы с объектами и изображениями, примеры работы с датами и системными настройками, создание спецэффектов и др. Уделено внимание разработке шуточных программ и игр. Показано создание интерактивных Web-страниц, получение сведений о системе и браузере, создание расширения для браузеров. Приведены практические советы по работе с JavaScript. Все примеры написаны с учетом особенностей двух популярных браузеров: Internet Explorer и Mozilla Firefox. Во втором издании появились новые и переработаны "старые" примеры с учетом появления новых ОС и браузеров.

Для веб-разбработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.09.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 27,09.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

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

ISBN 978-5-9775-0361-7

Оглавление

Введение

Обращение к читателю

Прошло более трех лет с момента, когда я сел за написание первого издания этой книги. За это время произошли большие изменения в IT-сфере. Появились новые версии операционных систем и браузеров. Настало время еще раз пересмотреть написанный код и добавить новые примеры.

Данная книга предназначена для тех, кто хочет расширить свои знания по JavaScript. Здесь собрано большое количество практичных (а может и бесполезных) примеров, написанных на этом языке, которые вы можете использовать на своих Web-страничках для того, чтобы привлечь посетителей.

Для кого эта книга

Книга предназначена для самого широкого круга читателей, желающих глубже изучить JavaScript. В данной книге нет объяснения теоретических основ, но если вы имеете опыт программирования на других языках, то без труда поймете предлагаемые примеры.

О браузерах

По статистике, подавляющее число пользователей Интернета применяют Internet Explorer, поэтому все примеры, приведенные в книге, протестированы в Internet Explorer 7.0, входящем в состав Windows Vista, а также доступном через механизм Windows Update пользователям Windows XP Service Pack 2. Продолжает набирать популярность браузер Mozilla Firefox. Я не мог не отметить этот факт, но, к сожалению, данные браузеры совместимы между собой лишь отчасти, в связи с чем некоторые примеры будут рассмотрены в двух вариантах. Для тестирования я использовал русскую версию Mozilla Firefox 3.0.*x*.

Вынужден огорчить поклонников Opera. Я не стал устанавливать на свой компьютер третий браузер. Если среди читателей найдется много поклонников этой программы, то в следующий раз я постараюсь учесть и их вкусы.

Тем не менее, мои знакомые проверили часть примеров на Opera и сообщили мне дополнительную информацию, которой я с вами поделюсь.

Как пользоваться примерами

В книге даны полные листинги всех примеров; кроме того, на сайте издательства (**www.bhv.ru**) можно найти готовые файлы для загрузки на свой компьютер. Я настоятельно рекомендую писать все примеры самостоятельно. Только в том случае, если что-то не заработало, а вы не можете найти причину ошибки, то воспользуйтесь готовым примером. Для его выполнения достаточно открыть Проводник, найти нужный вам файл и запустить его.

Благодарности

В своей книге я использовал не только примеры из своей коллекции, но и пользовался трудами других людей. Не стоит изобретать велосипед, который давно уже изобрели до вас. Я хотел бы выразить свою признательность всем людям, которые любезно предоставили мне возможность использовать их примеры в моей книге. В частности, это Александр Шуркаев, Ричард Вомерсли (Richard Womersley), llizard aka ejm, Андрей Орехов.

Отдельную благодарность хочу выразить своему брату Геннадию Климову за постоянную помощь и поддержку, а также сменному инженеру гостиницы "Вега" Петру Подугольникову за содействие при сборке нового мощного компьютера, на котором я и написал эту книгу.

Александр Климов

Москва, 2008 год

ГЛАВА 1

Первое знакомство с JavaScript

1.1. Первые приготовления

Данная глава предназначена для читателей с начальным уровнем подготовки. В ней я разберу пример в деталях. Это будет первый и последний пример, в котором будет "разжевана" каждая строчка кода. Далее мы помчимся галопом по Европам. Более опытные читатели могут просто мельком взглянуть на эту главу и сразу перейти к следующей.

Для начала надо определиться, в чем мы будем писать свои примеры. Сценарии JavaScript можно писать в любом текстовом редакторе. Существуют текстовые редакторы от сторонних разработчиков с более продвинутыми функциями, например, подсветкой синтаксиса языка. Я же все свои примеры писал в программе Блокнот, входящем в состав любой Windows. Это самый простой путь, не требующий установки новых программ. Тем более, у вас не будет другого выбора, если у вас нет в данный момент подключения к Интернету или диска с нужными программами, а также прав администратора для установки сторонних программ.

Так как сценарии используются в основном в Web-страницах, то для начала надо эту самую Web-страницу создать. Алгоритм действий следующий. Открываем папку Мои документы в Проводнике и создаем новую папку — *JavaScript в примерах*. Далее, открываем созданную папку, щелкаем правой кнопкой мыши на свободном месте и выбираем последовательно пункты контекстного меню **Создать | Текстовый документ**. В результате наших манипуляций будет создан пустой файл под именем Текстовый документ.txt. Так как Web-страницы имеют расширение htm или html, то щелкаем правой кнопкой мыши на созданном файле, выбираем команду **Переименовать** и присваиваем файлу новое имя, например, 1.htm. Windows спросит вас, действительно ли вы хотите присвоить файлу новое расширение. На данный во-

прос следует ответить утвердительно. После этого еще раз щелкаем правой кнопкой мыши на измененном файле и выбираем уже команду **Изменить**. У вас запустится стандартный Блокнот (по умолчанию). Пока он пустой. Вы можете теперь вводить код, приведенный в листинге чуть ниже. Будьте внимательны, постарайтесь избежать ошибок.

1.2. Первый сценарий

По устоявшейся традиции среди программистов первой программой при изучении любого языка программирования должно быть приложение, выводящее на экран сообщение "Здравствуй, мир!" (или "Hello, world!"). Не будем пренебрегать традициями и напишем наш первый сценарий в этом духе. В уже открытом Блокноте пишем следующее (листинг 1.1).

Листинг 1.1. Пишем первый сценарий

```
<html>
  <head>
    <title>Наш первый сценарий</title>
    <script type="text/javascript">
      alert("Здравствуй, мир!");
    </script>
  </head>
  <body>
    <h1>Первый пример</h1>
    <p>Добро пожаловать на наш первый урок!</p>
    <script language = "JavaScript">
      <!--
        document.write("Привет, мир!");
      // -->
    </script>
    <p>
    <input type="button" value="Поздороваться"
           onClick="alert('Еще раз здравствуй')"></p>
  </body>
</html>
```

Сохраните документ под именем helloworld.htm. Мы пока не будем смотреть написанный сценарий в действии, а получше изучим написанное.

1.3. Разбор полетов

Для того чтобы добавить сценарий JavaScript на Web-страничку, используется пара тегов `<script> </script>`. Данная пара дает указание браузеру рассматривать текст программы как сценарий. Исключение составляют только обработчики событий, где использование команды `script` не требуется. Здесь стоит немного задержаться. На сегодняшний день существует рекомендация использовать строчку

```
<script type = "text/javascript">
```

Совсем недавно более распространенным вариантом был код

```
<script language = "JavaScript">
```

Обратите внимание, что тег `script` содержит параметр `JavaScript`. Этот параметр определяет используемый язык сценария. Кроме того, в нем можно указать и номер версии языка.

Сейчас подобная конструкция считается устаревшей и разработчикам настоятельно рекомендовано отказываться от такого выражения. Существует вероятность, что новые версии браузеров будут более строго подходить к рекомендованным стандартам и не станут понимать устаревший синтаксис. В то же время очень старые версии браузеров не понимают нового формата. Поэтому некоторые программисты используют такую универсальную строчку:

```
<script type = "text/javascript" language = "JavaScript">
```

Существует еще один вариант для ленивых, которые не любят набирать лишние символы на клавиатуре:

```
<script>
...
</script>
```

Так как по умолчанию считается, что в браузерах используется именно сценарий JavaScript, то этот код не вызовет ошибки. Но подобный способ вызывает дополнительную нагрузку на браузер, и от такой экономии следует всячески воздерживаться.

Как вы уже заметили, сценарии могут размещаться в различных частях документа. Всего способов их размещения четыре:

- сценарий находится между тегами `<head> </head>`;
- сценарий находится в теле программы после тега `<body>`;
- сценарий находится в обработчиках событий. Данный способ позволяет выполнять сценарий JavaScript при наступлении нужного события;
- сценарий находится в отдельном файле. JavaScript позволяет создавать собственные файлы с расширением js. В этом случае на странице указывается местоположение такого файла. В нашем примере этот способ не использовался.

Таким образом, на одной странице могут располагаться сразу несколько сценариев. В какой последовательности браузер будет выполнять эти сценарии? Ничего сложного тут нет. Просто надо запомнить несколько правил. Сначала идет анализ сценария в заголовке документа HTML[1] (между тегами `<head> </head>`). После идет выполнение сценариев, расположенных в теле документа (между тегами `<body> </body>`). А обработчики событий запускаются при выполнении соответствующих событий. Например, обработчик кнопки `onClick` выполняется после нажатия кнопки. Взглянем на пример еще раз. Первый сценарий находится в заголовке документа:

```
<script type = "text/javascript">
  // выводим окно с сообщением
   alert("Здравствуй, мир!");
</script>
```

Между тегами сценария находится единственная команда `alert`, которая выводит специальное окно с выбранным сообщением. Внешний вид данного окна может отличаться в разных браузерах. Следующий сценарий уже находится в теле документа после тега `<body>` и тоже содержит единственную команду:

```
<script type = "text/javascript">
   document.write("Привет, мир!");
</script>
```

Команда `document.write` позволяет вводить новые строчки в документе. Текст "Привет, мир!" будет записан сразу после строчки "Добро пожаловать

[1] HTML (HyperText Markup Language) — язык гипертекстовой разметки.

на наш первый урок!" И, наконец, третий сценарий находится в обработчике события `onClick`. Как вы уже догадались, данное событие наступает при нажатии кнопки. В этом сценарии используется уже знакомая нам команда `alert`.

1.4. Скрытие сценария

У первых браузеров, предназначенных для просмотра Web-страничек, не было возможности использования сценариев. Тег `<script>` был добавлен в спецификацию HTML чуть позднее. Поэтому старые браузеры не выполняют сценарии JavaScript и просто выводят текст вашего сценария на странице. Внешний вид такой странички будет выглядеть не совсем так, как вы задумали.

Чтобы избежать подобных проблем, рекомендуется использовать скрытие сценария, для чего следует обрамлять код сценария тегами комментариев. Они дают указание старым браузерам игнорировать программу сценария. Комментарии в HTML начинаются с дескриптора `<!--` и заканчиваются дескриптором `-->`:

```
<script type="text/javascript">
  <!--
    document.write("Ваш браузер поддерживает JavaScript");
  // -->
</ script >
```

Впрочем, браузерами без поддержки JavaScript на практике никто не пользуется, поэтому можете не обращать на эту рекомендацию особого внимания. Я больше не буду в дальнейших примерах использовать этот прием.

1.5. Комментарии в JavaScript

Как и во многих языках программирования, существует возможность создания комментариев и для JavaScript. Они не выводятся на экран пользователя, но позволяют включить в код сценария необходимую документацию. Если вы напишете очень сложный код, то по прошествии длительного времени можете просто забыть, что он делает. Кроме того, комментарии помогут дру-

гому разработчику разобраться в вашем сценарии (если вы в этом заинтересованы).

Чтобы включить комментарий в сценарий JavaScript, начните строку с двух символов косой черты (слэш):

```
// это комментарий
```

Также можно включить комментарий после строчки кода:

```
a = a + 1; // к переменной a прибавляем единицу
```

Если вам понадобится закомментировать сразу несколько строк, то оставлять в начале каждой строки две наклонные косые черты может оказаться утомительным занятием. В этом случае используют комментарии, начинающиеся с `/*` и заканчивающиеся `*/`, что гораздо удобнее. Данный способ демонстрируется в приведенном ниже примере:

```
/* Этот сценарий содержит самые различные команды
   и операторы,
   а также комментарии */
```

Все, что находится между знаками комментариев, игнорируется браузером и не выполняется. Не забывайте, что комментарии JavaScript используются только в самом сценарии между тегами `<script>`. Их нельзя вводить в программном коде HTML.

1.6. Проблемы

Итак, мы написали свой первый сценарий, изучили его содержимое. Теперь самое время увидеть написанный сценарий в действии. Найдите созданный файл helloworld.htm в Проводнике и запустите его двойным щелчком мыши. У вас запустится Internet Explorer. Так как мы основную часть примеров будем тестировать на компьютерах с Windows Vista/XP Service Pack 2, то нас поджидают некоторые трудности. Компания Microsoft в последнее время уделяет много внимания безопасности компьютера. Сценарии JavaScript, написанные нехорошими людьми, могут представлять потенциальную опасность для вашего компьютера. Поэтому по умолчанию при загрузке Web-странички с локального компьютера вы получите сначала несколько предупреждающих сообщений. Сначала вы увидите диалоговое окно **Панель информации** (рис. 1.1).

Вам нужно нажать кнопку **OK** (если вы хотите, чтобы это окно у вас больше не появлялось, то установите соответствующий флажок). Но и это еще не все. Теперь вам придется щелкнуть по желтой полоске наверху страницы (но не на крестике в углу этой полоски!). Появится еще одно окно, где нужно выбрать первый пункт **Разрешить заблокированное содержимое...**. И наконец, в диалоговом окне **Предупреждение системы безопасности** нажать кнопку **Да**.

Не волнуйтесь, когда ваш документ будет выложен на сайт, данные предупреждения выводиться не будут, они появляются только при запуске сценариев на локальном компьютере. Впрочем, в *главе 13*, посвященной советам и хитростям, я расскажу, как можно обойти эту чересчур назойливую процедуру. Если же вы будете запускать примеры через Mozilla Firefox, то подобных сообщений не увидите.

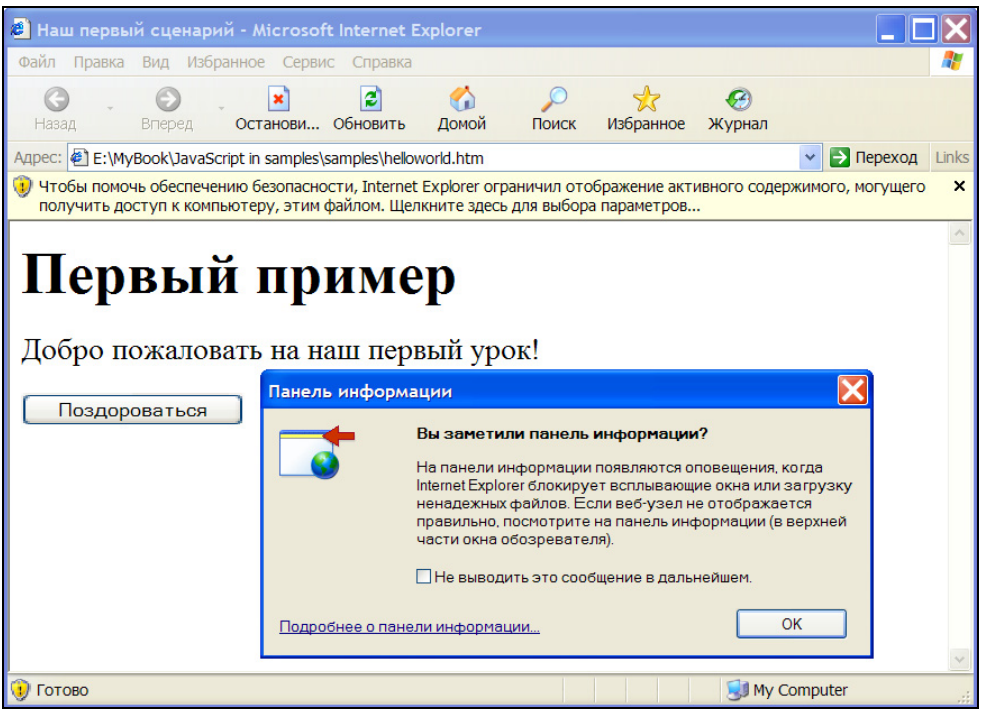

Рис. 1.1. Диалоговое окно **Панель информации**

1.7. Сценарий в действии

Наконец-то мы продрались через частокол предупреждений и можем увидеть написанный сценарий в действии. Сначала вы увидите окно с сообщением "Здравствуй, мир!" После нажатия кнопки **OK** откроется наша страница. Обратите внимание, что в третьей строчке выводится результат второго сценария. Осталось только нажать кнопку **Поздороваться**, чтобы запустить третий сценарий. На этом краткий курс по запуску сценариев JavaScript будем считать закрытым. В следующих главах мы начнем изучать уже конкретные примеры.

Примечание

Пример-знакомство находится в файле Chap01\helloworld.htm.

ГЛАВА 2

Информация о системе

2.1. Война браузеров

Вероятно, вам приходилось слышать о войне браузеров. Сначала было противостояние Netscape Navigator и Internet Explorer. Затем в борьбу вступил браузер норвежских разработчиков Opera. В настоящее время идет активная вербовка в ряды сторонников модного Mozilla Firefox. Я упомянул только популярные программы для серфинга по просторам Интернета, хотя существуют еще несколько проектов независимых разработчиков.

Появление каждого нового популярного браузера приносит головную боль владельцам Web-сайта. Существует очень большая проблема совместимости, несмотря на большие усилия сообщества выработать некие стандарты для отображения Web-страниц. В результате подобной несогласованности некоторые Web-страницы по-разному отображаются в различных браузерах.

Некоторые Web-программисты, отчаявшись угодить всем посетителям Web-сайта, просто выводят надпись: "Данная страничка предназначена для просмотра только в Internet Explorer". Естественно, фанаты других браузеров не остались в долгу и закрывают доступ для владельцев детища Microsoft. О том, как можно узнать информацию об установленном браузере и других компонентах системы, чтобы использовать ее в своих целях, и будет рассказано в этой главе.

2.2. Информация о системе и браузере

Всю необходимую информацию о запущенном браузере и системе у пользователя можно узнать при помощи объекта `navigator`. Каждый браузер имеет несколько общих методов и свойств данного объекта, а также несколько своих,

только ему присущих свойств. Я попытался составить небольшой перечень совместимости трех браузеров: Internet Explorer, Mozilla Firefox и Opera. Сначала перечислим общие для всех свойства и методы объекта `navigator`:

- `appCodeName` — кодовое имя браузера. Обычно используется Mozilla;
- `appName` — официальное имя браузера (Internet Explorer, Netscape, Opera);
- `appVersion` — версия браузера;
- `platform` — платформа, на которой работает браузер (обычно Win32);
- `cookieEnabled` — доступность сохранения cookie[1];
- `javaEnabled` — доступность на запуск сценариев JavaScript;
- `userAgent` — специальная строка для служебных целей.

Давайте на основе этой информации напишем сценарий, который будет работать в трех рассмотренных браузерах. Вставляем в теле документа следующий код (листинг 2.1).

Листинг 2.1. Получение информации о браузере и системе

```
<h1>Информация о браузере и системе</h1>
<script type = "text/javascript">
  var code = navigator.appCodeName;
  var name = navigator.appName;
  var vers = navigator.appVersion;
  var platform = navigator.platform;
  var cook = navigator.cookieEnabled;
  var je = navigator.javaEnabled();
  var ua = navigator.userAgent;
  document.write('Ваш браузер: ' + name +
                 '<br />Версия браузера: ' + vers +
                 '<br />Кодовое название браузера: ' + code +
                 '<br />Платформа: ' + platform +
                 '<br />Поддержка cookie: ' + cook +
                 '<br />Поддержка JavaScript: ' + je +
                 '<br />userAgent: ' + ua);
</script>
```

[1] Небольшой фрагмент данных о предыстории обращений пользователя к Web-сайту, автоматически создаваемый на компьютере пользователя.

При загрузке Web-страницы браузер покажет всю информацию о себе (рис. 2.1 и 2.2).

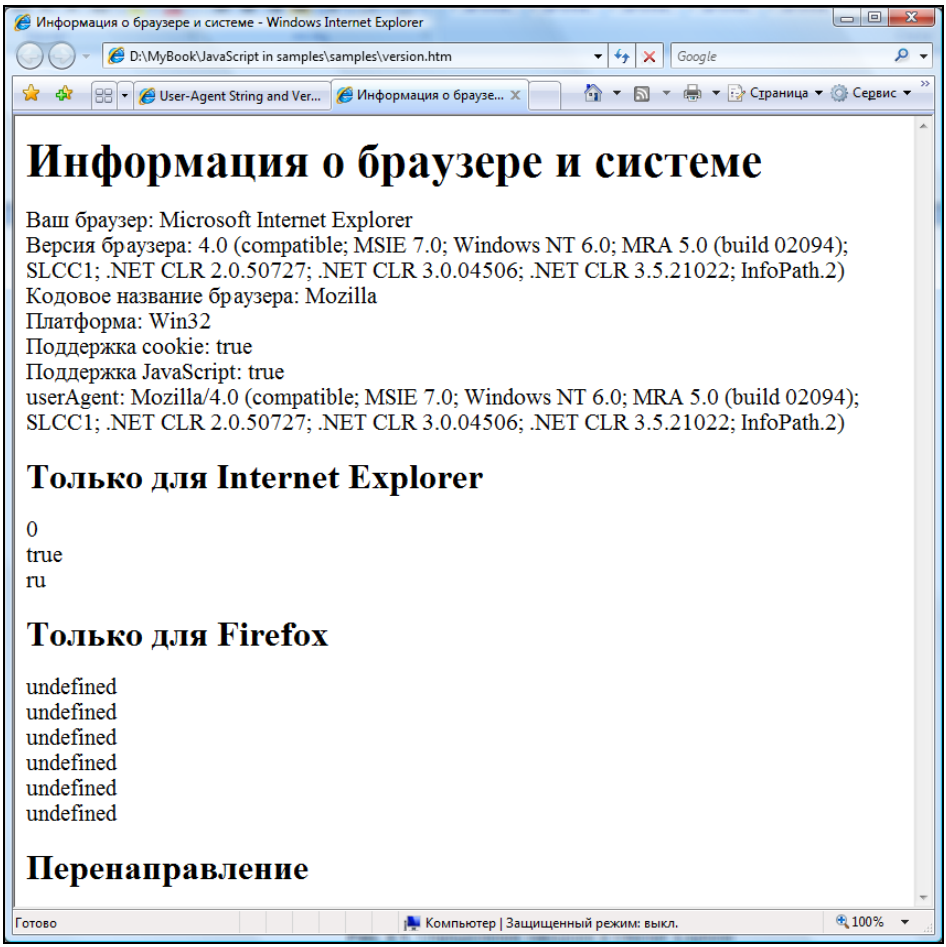

Рис. 2.1. Определение настроек в Internet Explorer

Для любознательных читателей сообщу, что Internet Explorer также поддерживает несколько своих свойств и методов:

- `appMinorVersion`;
- `online`;
- `systemLanguage`.

Я не стану приводить расшифровку данных свойств, поищите эту информацию в документации.

А мы напишем сценарий, предназначенный только для Internet Explorer (листинг 2.2).

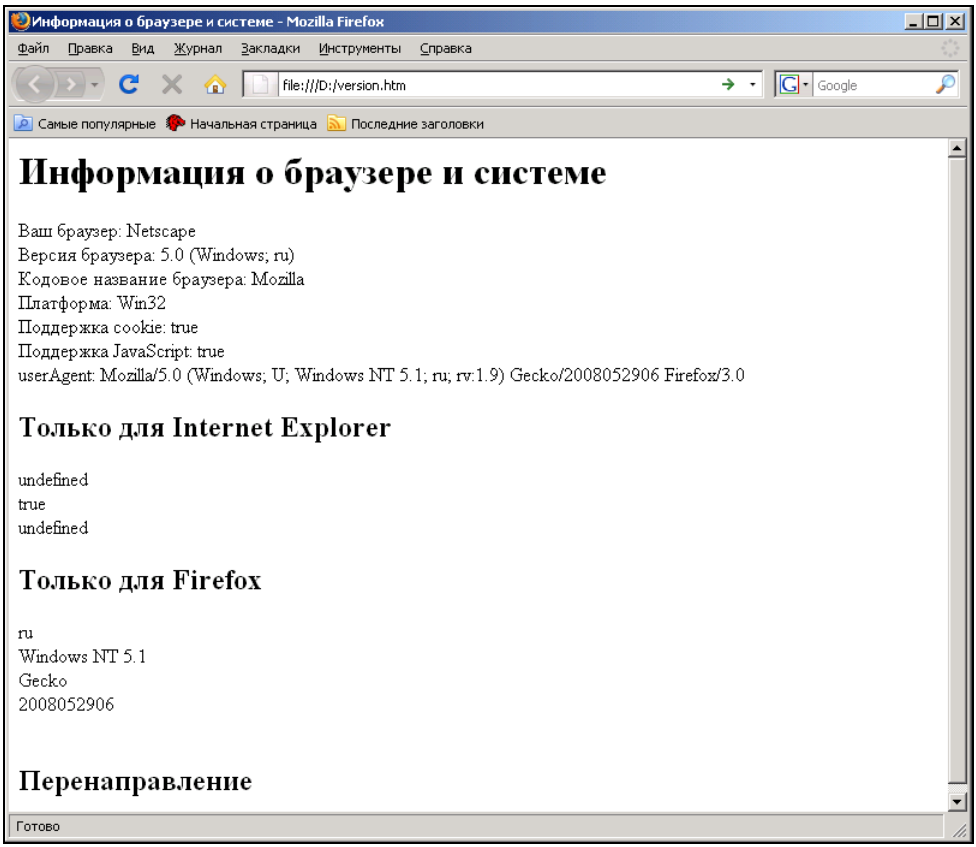

Рис. 2.2. Определение настроек в Mozilla Firefox

Листинг 2.2. Свойства, доступные только в Internet Explorer

```
<h2>Только для Internet Explorer</h2>
<script type = "text/javascript">
  document.write(navigator.appMinorVersion +
```

```
                '<br />' + navigator.onLine +
                '<br />' + navigator.systemLanguage);
</script>
```

А теперь приведу список свойств, присущих только Mozilla Firefox:

- `language`;
- `oscpu`;
- `product`;
- `productSub`;
- `vendor`;
- `vendorSub`.

А теперь сценарий, предназначенный для Mozilla Firefox (листинг 2.3).

Листинг 2.3. Свойства, работающие только в Mozilla Firefox

```
<h2>Только для Firefox</h2>
<script type = "text/javascript">
  document.write(navigator.language +
                '<br />' + navigator.oscpu +
                '<br />' + navigator.product +
                '<br />' + navigator.productSub +
                '<br />' + navigator.vendor +
                '<br />' + navigator.vendorSub);
</script>
```

Как видите, различия у браузеров довольно значительны. А ведь мы рассмотрели только один объект `navigator`. С другими объектами такая же история.

2.3. Определение браузера

Итак, мы увидели, что каждый браузер имеет свои особенности. А так ли это важно? Очень важно. Загрузив Web-страницу со сценарием, в котором со-

держатся не поддерживаемые браузером свойства, пользователь попросту не увидит ваш замысел. Существуют два подхода к решению проблемы совместимости браузеров. Первый подход немного утомительный. Разработчик пишет два варианта для разных браузеров и затем, на основе полученной информации, направляет пользователя на нужную Web-страницу. Вот небольшой пример такого сценария (листинг 2.4).

Листинг 2.4. Перенаправление пользователя на нужную Web-страницу

```
<script type = "text/javascript">
  var ver = navigator.appVersion;
  if (ver.indexOf("MSIE") != -1)
  {
    alert ("У вас запущен Internet Explorer");
    // window.location.href="ie.htm";
  }
  else
    alert ("У вас запущен Firefox");
    // window.location.href="firefox.htm";
</script>
```

Примечание

Готовые примеры находятся в файле Chap02\version.htm.

Я привел упрощенный сценарий, сделав допущение, что в мире существуют только два браузера. В реальной жизни, конечно, потребуется доработка; кроме того, у рассмотренного сценария есть один недостаток — требуется создание дополнительных Web-страниц под разные браузеры. Второй подход — использование оператора условия `if`. Рассмотрим на примере, для чего воспользуемся свойством `appName`. У Internet Explorer при вызове данного свойства возвращается строка `Microsoft Internet Explorer`, у браузера Mozilla Firefox — `Netscape` (листинг 2.5).

Листинг 2.5. Другой способ определения браузера

```
<script type = "text/javascript">
  var errorText = 'Не могу определить глубину цвета';
```

```
  if (navigator.appName.indexOf('Netscape') != -1)
    if (navigator.appVersion.substr(0, 1) > 3)
      document.write('Глубина цвета: ' + window.screen.pixelDepth)
    else
      document.write(errorText);

  if (navigator.appName.indexOf('Microsoft') != -1)
    if (navigator.appVersion.substr(0, 1) > 3)
      document.write('Глубина цвета: ' + window.screen.colorDepth)
    else
      document.write(errorText);
</script>

<NOSCRIPT>
  Увы, ваш браузер не поддерживает выполнение сценариев,
  поэтому я не могу определить настройки вашего монитора.
</NOSCRIPT>
```

В этом примере сначала определяется название браузера, затем его версия, после чего, с помощью поддерживаемых свойств, выводится информация. Как видно из этого сценария, у Netscape Navigator 4.0 для определения глубины цвета монитора используется свойство `pixelDepth`, а у Internet Explorer — свойство `colorDepth`.

> ***Примечание***
>
> Хорошая новость для разработчиков. Браузер Mozilla Firefox поддерживает оба эти свойства.

Обратите внимание, что нам нет острой необходимости помнить всю возвращаемую строку. С помощью функции `indexOf` мы определяем позицию в строке одного слова. Этого вполне достаточно, чтобы сценарий был работоспособным. Только не забывайте, что не только Mozilla Firefox имеет имя приложения Netscape. Существует, к примеру, еще и браузер Netscape Navigator.

> ***Примечание***
>
> Пример сценария находится в файле Chap02\version2.htm.

Хотя мы с вами договорились, что в этой книге все примеры тестируются только на Internet Explorer 7.0 и Mozilla Firefox 3.0.*x*, я все-таки приведу небольшой пример определения версии используемого браузера. Немного теории. Написать полностью универсальную функцию, определяющую браузер, установленный у пользователя, невозможно. Слишком много их развелось, и у каждого разработчика браузера свое видение объектной модели. Можно остановиться только на самых популярных и пренебречь остальными.

Итак, Internet Explorer используют в своей модели конструкции типа `document.all`, а Netscape Navigator — `document.layers`. Обе эти конструкции не являются стандартом, поэтому избегайте их использования на своих Web-страницах. Старые версии браузера Opera тоже грешили нестандартной конструкцией `document.all.item`. Но последние версии всех этих браузеров сделали шаг вперед и поддерживают правильную конструкцию `document.getElementById` (Mozilla Firefox изначально пытался следовать стандартам). Поэтому во многих сценариях используется приблизительно такой код (листинг 2.6).

Листинг 2.6. Еще один пример определения браузеров

```
var bNN4 = document.layers;
var bIE4 = document.all && document.all.item;
// проверка поддержки document.all.item необходима для отсечения Opera
var bW3CDOM = document.getElementById;
var bDOMBrowser = bNN4 || bIE4 || bW3CDOM;
```

Я привел только маленький кусочек кода сценария. Применение этого кода в реальной программе останется вам в качестве домашнего задания.

2.4. Свойства экрана

В этом разделе мы рассмотрим параметры, позволяющие определить некоторые свойства экрана.

- `availHeight`, `availWidth` — позволяют узнать рабочую высоту и ширину экрана в пикселах без учета панели задач и других панелей, постоянно занимающих место на экране, например, панели MS Office или ICQ.
- `height`, `width` — позволяют вычислить высоту и ширину экрана.

К счастью, эти параметры поддерживаются обоими нашими браузерами. Вот пример, позволяющий вычислить доступные размеры для окна браузера и раскрывающий его в соответствии с полученными результатами (листинг 2.7).

Листинг 2.7. Развернуть окно на максимально возможные размеры

```
<script type = "text/javascript">
  function fullWindow()
  {
    window.moveTo(0, 0);
    window.resizeTo(screen.availWidth, screen.availHeight);
  }
</script>

</head>
<body onload="fullWindow()">
```

ПРИМЕЧАНИЕ

Пример сценария находится в файле Chap02\fullscreen.htm.

Раз уж мы можем узнать размеры монитора пользователя, то можем создавать при необходимости разные Web-страницы под различные разрешения монитора. Вот самый простой сценарий, который приходит в голову (листинг 2.8).

Листинг 2.8. Определение разрешения

```
<script type = "text/javascript">
  var msg="У вашего монитора очень высокое разрешение";

  if (screen.width == 640 || screen.height == 480){
    location=("640x480.htm");}
  else if (screen.width == 800 || screen.height == 600){
    location=("800x600.htm");}
  else if (screen.width == 1024 || screen.height == 768){
    location=("1024x768.htm");}
```

```
  else if(screen.width == 1152 || screen.height == 864){
    location=("1152x864.htm");}
  else if (screen.width == 1280 || screen.height == 720){
    location=("1280x720.htm");}
  else if (screen.width > 1280 || screen.height > 720){
    alert(msg);
  }
</script>
```

Вам придется заранее подготовить пять разных Web-страниц под каждое из популярных разрешений монитора: 640x480.htm, 800x600.htm, 1024x768.htm, 1152x864.htm и 1280x720.htm.

> ***ПРИМЕЧАНИЕ***
>
> Пример сценария, определяющего разрешение экрана, находится в файле Chap02\screen.htm.

2.5. Размеры документа

Кроме размеров экрана, иногда требуется знать размеры самого документа в браузере. Здесь тоже приходится писать разный код для браузеров, т. к. используются различные свойства (листинг 2.9).

Листинг 2.9. Получение размеров документа в браузере

```
<script type = "text/javascript">
  var W; // ширина документа в браузере
  var H; // высота документа в браузере

  function checkSize()
  {
    if(document.all)
    {
      W = document.body.clientWidth;
      H = document.body.clientHeight;
    }
```

```
    else
    {
      W = innerWidth;
      H = innerHeight;
    }
  }
</script>
// Вызываем функцию в нужном месте для получения размеров документа
<script type = "text/javascript">
  checkSize();
  alert("Высота: " + H);
  alert("Ширина: " + W);
</script>
```

ПРИМЕЧАНИЕ

Пример сценария, определяющего размер документа в браузере, находится в файле Chap02\browsersize.htm.

2.6. Информация об операционной системе

Чтобы получить информацию об операционной системе посетителя вашей Web-страницы, можно воспользоваться свойством `appVersion` объекта `navigator` (листинг 2.10).

Листинг 2.10. Получение информации об операционной системе

```
function getOS()
{
  if (navigator.appVersion.indexOf('Windows')>=0) return 'Windows';
  if (navigator.appVersion.indexOf('Linux')>=0)   return 'Linux';
  if (navigator.appVersion.indexOf('Sun')==0)     return 'SunOS';
}
```