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®

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

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).

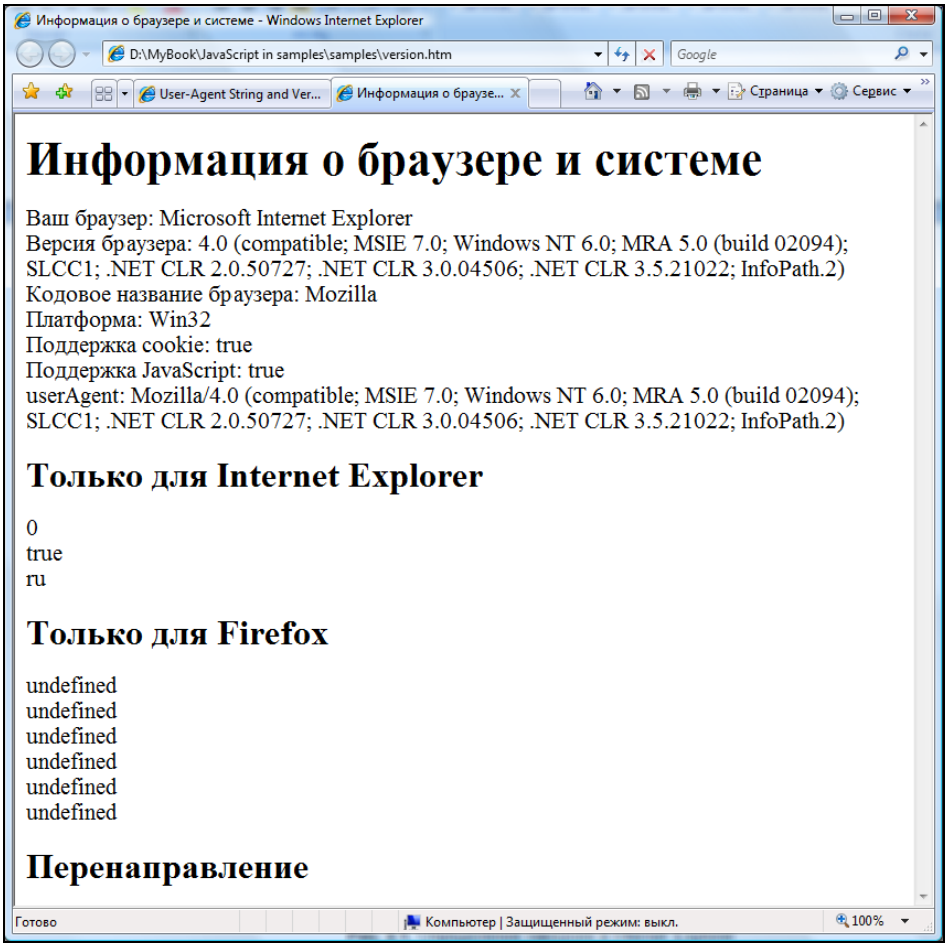

Рис. 2.1. Определение настроек в Internet Explorer

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).

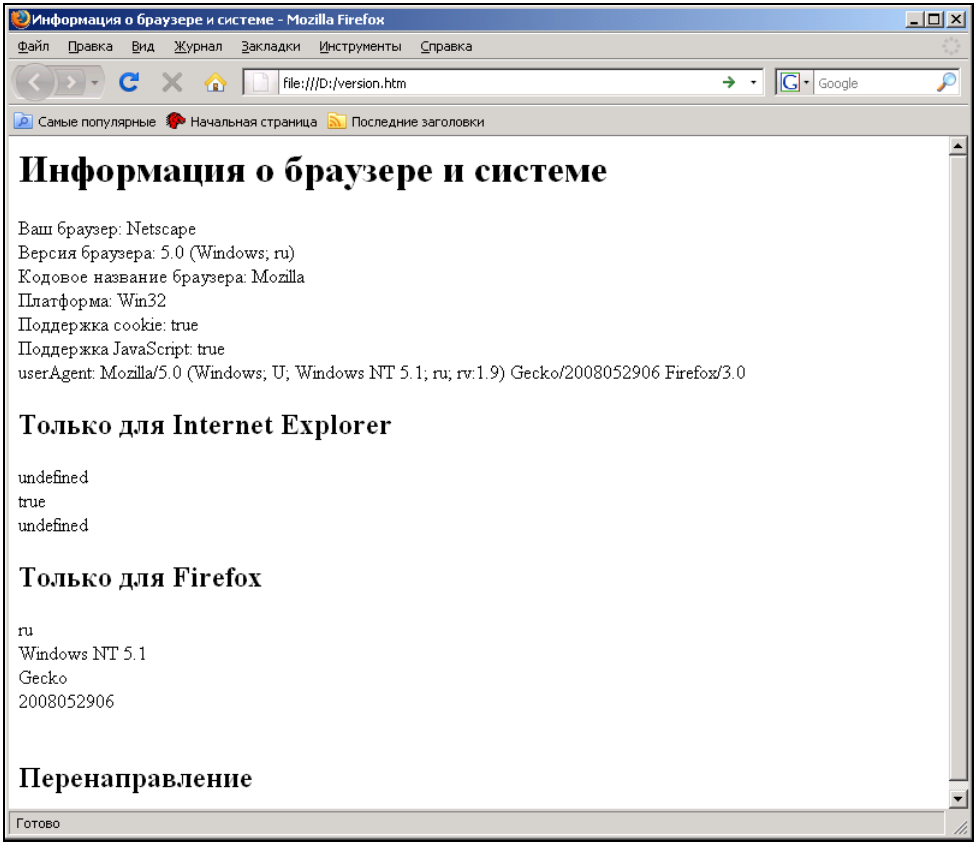

Рис. 2.2. Определение настроек в Mozilla Firefox

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';
}
```

Вы можете самостоятельно расширить список известных операционных систем. В приведенном примере я ограничился тремя наиболее известными операционными системами.

2.7. Свойство *userAgent*

Давайте подробнее рассмотрим свойство `userAgent` объекта `navigator`, с помощью которого можно извлечь много полезной информации. Когда вы посещаете Web-страницу, то ваш браузер посылает строку `userAgent` серверу, по которому сервер определяет ваш браузер, номер версии и другие детали о вашей системе. Данная информация помогает серверу подстроиться под работу вашего браузера. Строка, возвращаемая свойством `userAgent`, у всех браузеров уникальная. Например, у браузера Internet Explorer 7.0, запущенного в системе Windows Vista, свойство возвращает очень длинную строку

```
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0; MRA 5.0 (build 02094);
SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.04506; .NET CLR 3.5.21022;
InfoPath.2)
```

Примечание

Чтобы быстро получить строку `userAgent` в своем браузере, наберите прямо в адресной строке следующее (с соблюдением регистра символов): **javascript:alert(navigator.userAgent)**.

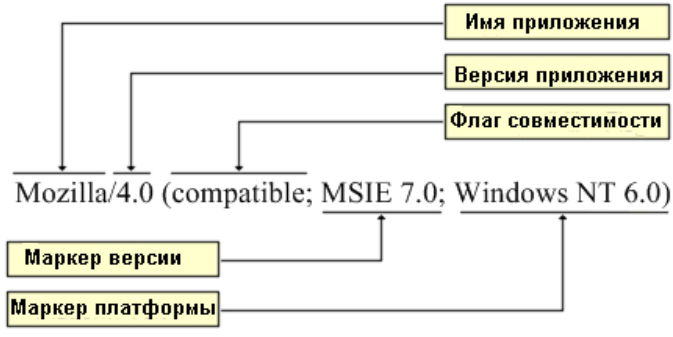

Рис. 2.3. Строка `userAgent`