

ГЛАВА

19

Сценарии и объекты

Джерри Хоникатт

В этой главе...

| | |
|--|----|
| Динамический HTML | 2 |
| Работа с элементами Web-страницы | 2 |
| Доступ к нескольким элементам в наборе | 3 |
| Оперативное изменение содержимого элемента | 3 |
| Обработка событий объектной модели | 4 |
| Соединение сценариев со стилями и классами | 8 |
| Управление свойством видимости объекта | 12 |
| Позиционирование элементов HTML на Web-странице | 20 |
| Чтение и запись cookie с использованием объектной модели | 23 |

Динамический HTML

Динамический HTML — следующая большая волна Web. Это комбинация HTML, стилей и сценариев, используемая для управления всеми элементами Web-страницы.

Различия между динамическим HTML Microsoft и Netscape огромны. Они так велики, что разработка совместимой с разными платформами Web-страницы с динамическим HTML почти невозможна. Самое большое различие между обеими версиями динамического HTML заключается в объектной модели. Объектная модель Microsoft идет намного дальше, чем у Netscape, — она включает каждый элемент на странице, обеспечивая ряд свойств, элементов, событий и наборов. Объектная модель Microsoft ближе к тому, что, как я ожидаю, консорциум W3C определит в качестве рекомендации. Поэтому все примеры, приведенные в этой книге, работают с браузером Internet Explorer, но некоторые не работают с Netscape Communicator.



WWW

Вы можете найти дополнительную информацию об объектной модели каждой компании на соответствующем узле Web. Вы можете также узнать о планируемой рекомендации объектной модели документа W3C на его Web-сервере (<http://www.w3.org/>). Далее перечислены некоторые URL с дополнительной информацией:

http://www.microsoft.com/msdn/sdk/inetsdk/help/dhtml/ref_objects/objects.htm#om40_objects
<http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/navobj.htm>
<http://www.w3.org/Press/DOM-core.html>

Работа с элементами Web-страницы

Динамический HTML определяет взаимодействие с Web-страницей, поэтому вам нужно иметь доступ к элементам на странице (обратите внимание, что Internet Explorer в настоящее время единственный браузер, который обеспечивает доступ ко всем элементам страницы).

Наиболее простой метод обращения к элементу состоит в том, чтобы использовать объект `all`, который можно индексировать порядковым номером, именем элемента или его идентификатором. Например, следующая строка JavaScript устанавливает `e` в качестве объектного представления элемента, идентификатор которого равен `Example`. Вы можете точно так же использовать имя элемента или индексировать его позицией во всей совокупности элементов:

```
e = document.all( "Example" );
```

Объектная модель документа обеспечивает два варианта для получения доступа к элементу страницы. Обратите внимание, что вы можете просто указать идентификатор элемента в качестве объекта, как показано во втором примере, если идентификатор в пределах Web-страницы и не находится в противоречии с именами из объектной модели или JavaScript.

```
e = document.all.Example  
e = Example
```

На заметку

Коллекции (collections) Microsoft и массивы (arrays) Netscape обеспечивают доступ к набору связанных элементов. Например, набор форм — массив, который содержит по элементу для каждой формы на Web-странице. Поскольку форма представляет собой порожденный объект документа, вы обращаетесь к этому набору следующим образом: `document.forms`. Вы можете индексировать набор форм номером, с помощью идентификатора элемента или имени, подобно следующему: `document.forms ("MyForm")`. Создав с помощью набора ссылку на объект, можно обращаться к любому из свойств этого объекта, к методам и событиям.

Доступ к нескольким элементам в наборе

В предыдущих примерах мы привязали `e` к элементу с идентификатором `Example`. Если Web-страница содержит только один элемент с таким идентификатором, вы можете обращаться к любому из его свойств следующим образом: `e.property`. Что же получится, если Web-страница содержит больше одного элемента с таким идентификатором? В этом случае `e` представляет собой набор, содержащий ссылки на соответствующие элементы.

Как показано в листинге 19.1, можно проверить переменную, чтобы узнать, содержит ли она одиночное значение или набор. Если свойство переменной `length` имеет значение `null`, то переменная содержит одиночное значение. Если же свойство не равно `null`, значит, переменная содержит массив величин, индексируемый порядковым номером. Вы можете обращаться к каждой величине в отдельности, получая доступ к свойствам каждой из них.

Листинг 19.1. Работа с наборами

```
<HTML>
<P ID=Test>Привет</P>
<P ID=Test>До свиданья</P>

<SCRIPT LANGUAGE=JAVASCRIPT>
var i, e;

e = document.all( "Test" );
if( e.length == null )
    alert( "В наборе один элемент" );
else
    for( i = 0; i < e.length; i++ )
        alert( i.toString() + ":" + e[i].innerText );
</SCRIPT>
</HTML>
```

Оперативное изменение содержимого элемента

Вы, вероятно, уже понимаете, что можно динамически вносить код в Web-страницу, когда броузер открывает ее. Это делается с помощью метода `document.write`. Проблема состоит в том, что этот метод нельзя применить для изменения содержимого Web-страницы после того, как броузер откроет ее. В объектной модели Internet Explorer каждый элемент содержит два интересных свойства, которые все же позволяют изменять содержимое после открытия (это не работает в Netscape Communicator):

- `innerText` содержит текстовое содержимое дескриптора контейнера без HTML;
- `innerHTML` содержит полное содержимое дескриптора контейнера, включая HTML.

В листинге 19.2 показан пример, в котором текст на Web-странице изменяется после того, как она была открыта. В начале файла вы видите дескриптор `<P>` с идентификатором `Test`. Когда пользователь нажимает кнопку, браузер вызывает обработчик события `ChangeText()`. Первая строка `ChangeText()` присваивает `innerText` значение, которое заставляет браузер полностью заменить текст в дескрипторе `<P>` на новое значение. Третья строка присваивает `innerHTML` значение, содержащее код HTML, который браузер аккуратно вставляет в контейнер `<P>`. При использовании и `innerText` и `innerHTML` вы можете изменять содержимое Web-страницы без обмена данными с сервером. На рис. 19.1 вы можете увидеть, как работает этот код в браузере.

Листинг 19.2. Динамическое изменение содержимого Web-страницы

```
<HTML>
  <P ID=Test>Example Text</P>

  <FORM>
    <INPUT ID=Click TYPE=BUTTON VALUE="Click"
onClick="ChangeText () ;">
  </FORM>

  <SCRIPT LANGUAGE=JAVASCRIPT>
    function ChangeText()
    {
      Test.innerText = "Динамически построенный текст";
      window.alert( "Новый текст" );

      Test.innerHTML = "Динамически построенный <STRONG> код
HTML</STRONG>";
      window.alert( "Новый код HTML" );
    }
  </SCRIPT>
</HTML>
```

Обработка событий объектной модели

Если вы использовали JavaScript, то уже знакомы с созданием обработчиков событий. Есть несколько вещей, которые нужно знать о моделях событий Internet Explorer и Communicator. Это касается событий, сгенерированных объектной моделью.

- Internet Explorer поддерживает концепцию, названную *event bubbling* (всплытие пузырьков событий), что означает, что “пузырьки события” всплывают сквозь иерархию страницы, пока не достигают верха. Если пользователь щелкает на слове `here`, как показано в листинге 19.3, событие направляется сначала к дескриптору `<A>`, затем всплывает к дескриптору `<P>`. Последние два объекта, к которым направляется событие, — документ (`document`) и окно (`window`). Листинг 19.4 представляет собой пример, который можно использовать с Internet Explorer, чтобы лучше понять всплытие; откройте этот файл в Internet Explorer и следите за появляющимися сообщениями.
- Communicator поддерживает концепцию, названную *event capturing* (перехватом события), которая является противоположностью всплытия. В ней события по-

являются на верху иерархии страницы и спускаются вниз. Если пользователь щелкнет на слове `here`, событие направляется сначала к объектам `document` и `window`, а затем — к дескриптору `<A>`. Обратите внимание, что Communicator пропускает дескриптор `<P>`, потому что он не поддерживает события.

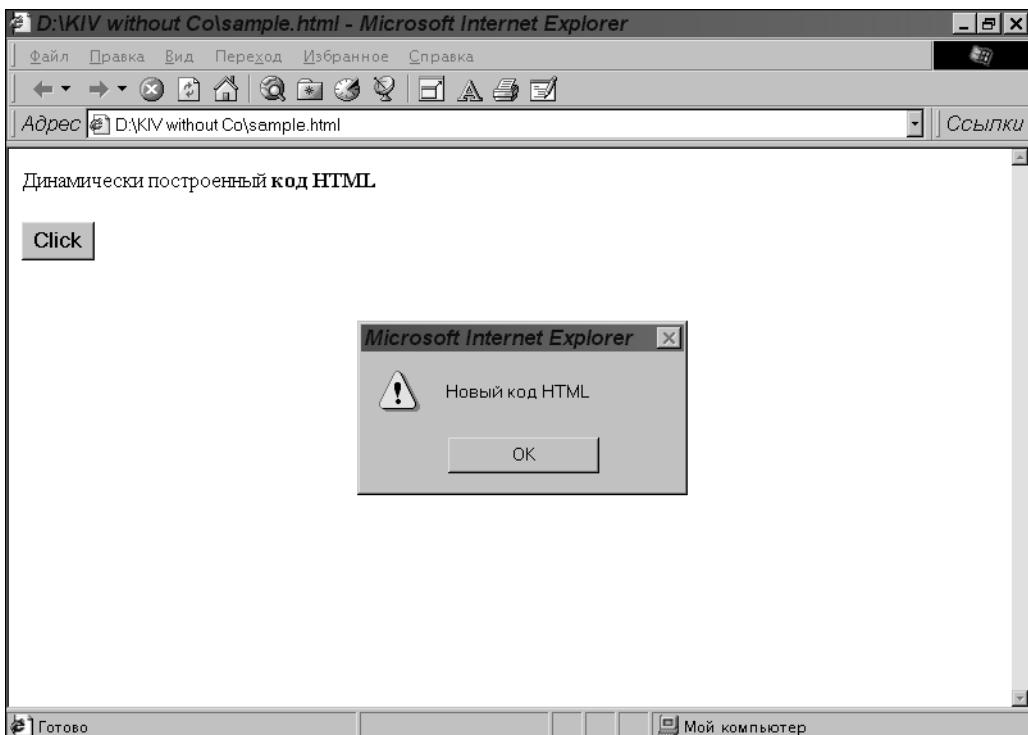


Рис. 19.1. Изменение содержимого страницы после ее загрузки в браузер

Листинг 19.3. Сравнение концепций всплытия событий и фиксации событий

```
<HTML>
  <BODY>
    <SCRIPT LANGUAGE=JAVASCRIPT>
      function Hello()
      {
        window.alert( "Вы щелкнули на заголовке" );
      }
    </SCRIPT>

    <P>Щелкните <A onClick="Hello();" href="">здесь</A>, и вы увидите
сообщение </P>
  <BODY>
</HTML>
```

Эти различия чрезвычайно важны, если вы создаете межплатформенные Web-страницы. Всплытие события и перехват события представляют собой методы, которые можно использовать, чтобы написать один обработчик события для группы объектов. Это очень удобно. Вы научитесь этому в следующих разделах.

ВНИМАНИЕ!

Если вы создаете межплатформенные Web-страницы, постарайтесь вообще не использовать всплытие событий или их фиксацию. Эти методы несовместимы, поэтому ваша страница не будет читаться в обоих браузерах.

Листинг 19.4. Пояснение всплытия события

```
<HTML>
<BODY onclick="alert( 'Щелк!' );">

<DIV onClick="alert('Снаружи');">
    <DIV onClick="alert('В середине');">
        <SPAN onClick="alert('Внутри');">Щелкните здесь</SPAN>
    </DIV>
</DIV>

</BODY>
</HTML>
```

Получение дополнительной информации о событии

Internet Explorer и Communicator предоставляют дополнительную информацию обработчикам события в виде объекта `event`. Этот объект доступен только внутри обработчика события. Объект `event` имеет различные свойства в Internet Explorer и Communicator. В табл. 19.1 показаны наиболее важные свойства объекта `event`, поддерживаемые обоими браузерами.

Таблица 19.1. Объект `event`

| Свойство | Описание |
|--------------------------------|---|
| Internet Explorer | |
| CancelBubble | Установите значение равным <code>true</code> , чтобы отменить всплытие |
| KeyCode | Код ASCII нажатой клавиши |
| ReturnValue | Установите значение равным <code>false</code> для отмены действия (например, посылки формы) |
| SrcElement | Объект HTML, который вызвал событие |
| X | Координата указателя мыши по оси x в момент наступления события |
| Y | Координата указателя мыши по оси y в момент наступления события |
| Netscape Communicator 4 | |
| target | Объект HTML, который вызвал событие |
| type | Тип события |
| pageX | Координата указателя мыши по оси x в момент наступления события |
| pageY | Координата указателя мыши по оси y в момент наступления события |
| which | Номер кнопки мыши или код ASCII клавиши |

Отмена события в Internet Explorer

Не всегда нужно, чтобы “пузырек события” всплывал к самому верху иерархии страницы. Например, если вы имеете обработчик для `onClick` в дескрипторах `<BODY>` и `<P>`, Internet Explorer вызовет оба, что, вероятно, вам не требуется.

Как показано в листинге 19.5, при щелчке пользователя на сообщении Щелкните здесь, событие сначала направляется к дескриптору `<P>`, который выдает сообщение: Вы щелкнули на абзаце. Затем событие автоматически всплывает к дескриптору `<BODY>`, который выдает сообщение: Вы щелкнули на контейнере `<BODY>`. В обработчике события установите `event.cancelBubble` равным `true`, чтобы предотвратить всплытие события выше определенного дескриптора. В листинге 19.5 уберите отметку комментария перед последней строкой в сценарии. Это предотвращает посылку события далее по иерархии документа.

Листинг 19.5. Отмена всплытия события

```
<HTML>
  <BODY onClick="alert('Вы щелкнули на контейнере <BODY>');">
    <P onClick="ClickMessage();">Щелкните здесь</P>
    <SCRIPT LANGUAGE=JAVASCRIPT>
      function ClickMessage()
      {
        alert( " Вы щелкнули на абзаце " );
        // window.event.cancelBubble = true;
      }
    </SCRIPT>
  </BODY>
</HTML>
```

Обработка события для группы объектов

Как отмечалось ранее, всплытие события делает возможным обработку события для группы связанных объектов. Например, если вы хотите создать эффект подсветки при перемещении для пяти блоков текста, то можете либо написать обработчик события для каждого блока, либо поместить все пять блоков внутри дескриптора `<DIV>` и написать один обработчик события для дескриптора `<DIV>`. Не забудьте, что объект `event` точно сообщает вам, какой объект генерировал событие, используя свойство `srcElement`. Вы должны прочитать это свойство, чтобы определить, какой объект в группе вызвал событие; затем вы обработаете его соответствующим образом.

Листинг 19.6 показывает пример использования одного обработчика события `onClick` для группы связанных объектов. Обратите внимание, что все дескрипторы `` имеют уникальные `ID` и включены внутрь одного дескриптора `<DIV>`. Дескриптор `<DIV>` связывает функцию `LinkClick()` с событием `onClick`. Внутри `LinkClick()` вы видите оператор `If`, который проверяет `srcElement.id`, чтобы вызвать код, соответствующий объекту. Если `ID` исходного элемента — 1, обработчик события выводит сообщение: Вы щелкнули на первом элементе; если `ID` — 2, обработчик события выводит: Вы щелкнули на втором элементе и так далее.

Листинг 19.6. Обработка событий для нескольких объектов с помощью одного обработчика

```
<HTML>
<BODY onclick="alert( 'Щелк!' );">
<SCRIPT LANGUAGE=JAVASCRIPT>

    function LinkClick()
    {
        if( window.event.srcElement.id == "1" )
        {
            window.alert( "Вы щелкнули на первом элементе" );
            window.event.cancelBubble = true;
        }
        else
            if( window.event.srcElement.id == "2" )
            {
                window.alert( "Вы щелкнули на втором элементе" );
                window.event.cancelBubble = true;
            }
        else
            if( window.event.srcElement.id == "3" )
            {
                window.alert( "Вы щелкнули на третьем элементе" );
                window.event.cancelBubble = true;
            }
    }
</SCRIPT>

<DIV onClick="LinkClick();">
    <SPAN ID="1">Первый элемент</SPAN><BR>
    <SPAN ID="2">Второй элемент</SPAN><BR>
    <SPAN ID="3">Третий элемент</SPAN>
</DIV>

</BODY>
</HTML>
```

Соединение сценариев со стилями и классами

Динамический HTML предоставляет возможность изменения стиля элемента внутри сценария. Internet Explorer 4.0 в настоящее время единственный броузер, который обеспечивает эту возможность, но после того, как W3C примет рекомендацию объектной модели документа, Communicator быстро последует его примеру.

Объектная модель документа Internet Explorer предоставляет доступ ко всем стилям для каждого элемента на Web-странице. Вы можете обращаться к стилю элемента, используя свойство стиля, как показано в приведенной ниже строке кода. Test.style — объект стиля для элемента по имени Test; color — свойство, которое отражает атрибут color CSS.

```
Test.style.color = "red";
```

Приложение Б “Справочник свойств листов стилей” содержит все возможные атрибуты CSS. Объектная модель документа предоставляет доступ ко всем из них. При именовании атрибутов CSS в сценариях нужно соблюдать определенные правила, поскольку CSS использует тире в именах атрибутов, а это не поддерживается в JavaScript.

- Вводите прописной буквой первый символ, находящийся справа от каждого тире.
- Уберите все тире из имени стиля.

В листинге 19.7 показан пример Web-страницы, в которой фоновый и основной цвета дескриптора <P> изменяются “на ходу”. Пользователь вводит соответствующие названия фонового и основного цвета, после чего щелкает на Change. Обработчик события ChangeColor() устанавливает свойства backgroundColor и color для стиля элемента Test равным значениям, введенным пользователем.

Листинг 19.7. Изменение стиля элемента

```
<HTML>

<P ID="Test">Это обычный элемент HTML</P>
<HR>
<FORM>
    Введите название фонового цвета:
    <INPUT ID=Background TYPE=TEXT SIZE=20 VALUE="White"><BR>
    Введите название основного цвета:
    <INPUT ID=Color TYPE=TEXT SIZE=20 VALUE="Black">
    <INPUT TYPE=BUTTON VALUE="Change" onClick="ChangeColor(
        Background.value, Color.value );">
<FORM>

<SCRIPT LANGUAGE=JAVASCRIPT>

    function ChangeColor( Background, Color )
    {
        Test.style.backgroundColor = Background;
        Test.style.color = Color;
    }

</SCRIPT>

</HTML>
```

Изменение свойств стиля отдельного элемента не всегда является наиболее эффективным способом изменения внешнего вида элемента. Если вы изменяете ряд свойств при обработке события, можно определить два различных класса для элемента; затем изменяйте классы при обработке события, используя свойство className элемента. В листинге 19.8 показан соответствующий пример. Каждый раз, когда пользователь щелкает на Change, броузер вызывает обработчик события ChangeColor(). Обработчик ChangeColor() проверяет свойство className элемента Test, чтобы определить, какой стиль использовать для него. Если в текущий момент он установлен равным Normal, обработчик события изменяет его на Reverse и наоборот. Обратите внимание, что дескриптор <P> первоначально устанавливает свойство CLASS равным Normal.

Листинг 19.8. Изменение класса элемента

```
<HTML>

<STYLE>
.Normal {background-color: white; color: black}
.Reverse {background-color: black; color: white}
</STYLE>

<P ID="Test" CLASS="Normal">Это обычный элемент HTML</P>
<HR>
<FORM>
    Щелкните на кнопке, чтобы изменить стиль:
    <INPUT TYPE=BUTTON VALUE="Change" onClick="ChangeColor();">
<FORM>

<SCRIPT LANGUAGE=JAVASCRIPT>

    function ChangeColor()
    {
        if( Test.className == "Normal" )
            Test.className = "Reverse";
        else
            Test.className = "Normal";
    }

</SCRIPT>

</HTML>
```

Переопределение функционирования гиперссылок

Освоив работу с событиями и стилями, можно приступать к рассмотрению более полного примера. Пример, показанный в листинге 19.9, изменяет вид и работу гиперссылок на Web-странице. Когда пользователь перемещает указатель мыши по гиперссылке, она изменяет цвет. Когда указатель оказывается вне гиперссылки, цвет изменяется на первоначальный. Вот дополнительная информация об этом примере.

- Этот пример определяет два стиля: A и A.LitUp. A становится заданным по умолчанию стилем, используемым для всех дескрипторов <A> в Web-странице. A.LitUp — стиль, используемый для изменения внешнего вида гиперссылок при перемещении указателя мыши по ссылке.
- Для дескриптора <BODY> определены два обработчика события: RollOn(), который обрабатывает событие onMouseOver, и RollOff(), который обрабатывает событие onMouseOut. Так как эти обработчики событий определены в дескрипторе <BODY>, они будут обрабатывать любые события onMouseOver или onMouseOut для документа, которые не были отменены.
- RollOn() проверяет источник события onMouseOver. Если событие не сгенерировано из контейнера дескриптора <A>, он возвращает управление. В противном случае он изменяет класс стиля для исходного элемента на A.LitUp. Это вызывает эффект изменения цвета ссылки при перемещении пользователем указателя мыши по этой ссылке.

- RollOff() также проверяет источник события onmouseout. Если событие было генерировано дескриптором <A>, он изменяет класс стиля для исходного элемента на A. Это возвращает первоначальный цвет ссылке.

Листинг 19.9. Переопределение функционирования гиперссылок

```
<HTML>
  <BODY onmouseover="RollOn();" onmouseout="RollOff();">

<STYLE>
  A { color: blue; font-size: smaller; font-family: arial;
       font-weight: bold; text-decoration: none }
  A.LitUp { color: red; font-size: smaller;
             font-family: arial; font-weight: bold;
             text-decoration: underline }
</STYLE>

<SCRIPT LANGUAGE=JAVASCRIPT>
  function RollOn()
  {
    if( window.event.srcElement.tagName != "A" ) return;

    if( window.event.srcElement.className == "" )
      window.event.srcElement.className = "LitUp";
  }

  function RollOff()
  {
    if( window.event.srcElement.className == "LitUp" )
      window.event.srcElement.className = "";
  }
</SCRIPT>

Посетите <A HREF="http://rampages.onramp.net/~jerry"> Web-узел
Джерри </A>
Можете также посетить мои любимые Web-узлы:
<UL>
  <LI><A HREF="http://www.microsoft.com">Microsoft Corporation</A>
  <LI><A HREF="http://www.netscape.com">Netscape Corporation</A>
</UL>
</BODY>
</HTML>
```

Создание эффекта подсветки для инструментальных панелей и меню

В листинге 19.10 представлена программа, которая работает только с Internet Explorer 4.0. Вы можете использовать ее, чтобы создать меню или инструментальные панели, которые изменяются при перемещении по ним указателя мыши. Вот как это происходит.

- Два стиля определены для эффекта подсветки при перемещении: Normal и LitUp. Первый только изменяет указатель на изображение руки, чтобы пользователь знал, что он находится над чем-то, на чем можно щелкнуть. Второй — инвертирует цвет объекта.

- Каждый элемент в инструментальной панели включен в дескриптор . Каждый дескриптор имеет уникальный идентификатор. Все элементы в инструментальной панели включены в один дескриптор <DIV>, который определяет обработчики событий для onMouseOver и onMouseOut. События, сгенерированные в дескрипторе всплывают до дескриптора <DIV>.
- Обработчики событий RollOn() и RollOff() переключают стиль для объекта, который сгенерировал событие таким образом, что цвет объекта инвертируется при перемещении пользователем указателя мыши по этому объекту, как показано на рис. 19.2.

Листинг 19.10. Создание эффекта подсветки при перемещении для инструментальных панелей и меню

```
<HTML>
<BODY>

<SCRIPT LANGUAGE=JAVASCRIPT>
    function RollOn()
    {
        if( window.event.srcElement.className == "Normal" )
            window.event.srcElement.className = "LitUp";
    }

    function RollOff()
    {
        if( window.event.srcElement.className == "LitUp" )
            window.event.srcElement.className = "Normal";
    }
</SCRIPT>

<STYLE>
.Normal { cursor: hand }
.LitUp { cursor: hand; color: white; background-color: black }
</STYLE>

<DIV onMouseOut="RollOff(); onMouseOver="RollOn();">
    <SPAN CLASS="Normal">Первый элемент </SPAN>
    <SPAN CLASS="Normal">Второй элемент</SPAN>
    <SPAN CLASS="Normal">Третий элемент </SPAN>
</DIV>

</BODY>
</HTML>
```

Управление свойством видимости объекта

Вы можете делать некоторые элементы на Web-странице видимыми или невидимыми, используя свойство display объекта style. Установите это свойство равным none для того, чтобы браузер скрыл объект:

```
Element.style.display = "none";
```

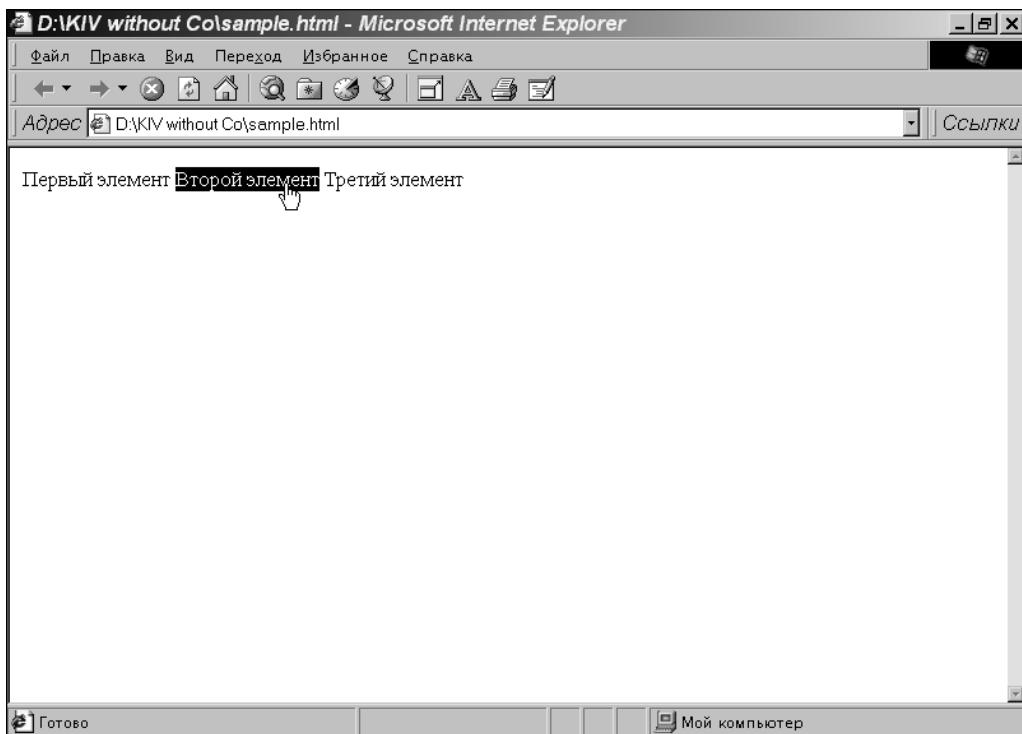


Рис. 19.2. Элемент изменяет цвет, когда пользователь перемещает указатель мыши поверх него. Обратите также внимание на изменение формы указателя

Установите это свойство равным пустой строке, как показано в следующей строке кода, и броузер отобразит элемент.

```
Element.style.display = "";
```

Скрытие элемента заставляет броузер повторно отобразить Web-страницу, поэтому там, где находится невидимый элемент, не образуется пустое место. На рис. 19.3 показана страница до и после того, как элемент был скрыт.

Отображение и сокрытие частей формы

Вы можете скрыть дополнительные поля формы от начинаящего пользователя и дать возможность более подготовленному пользователю добраться до них с помощью специальной кнопки. Это подход, который во многих случаях применяется в Windows 95 и Windows NT. Вы видели диалоговое окно в Windows с кнопкой, помеченной Advanced или More? Когда пользователь нажимает такую кнопку, диалоговое окно разворачивается, отображая большее количество полей.

Листинг 19.11 представляет собой пример такой формы на Web-странице. Последняя часть формы, определенной в этом примере, содержит дескриптор `<DIV>` с идентификатором `More`, который содержит ряд полей, не отображаемых первоначально, потому что дескриптор `<DIV>` скрыт. Обработчик события `OnClick` для кнопки `FEEDBACK_MORE`, которую вы видите в форме, называется `OpenMore()`. Эта функция переключает флагок по имени `MoreIsUp`, который отслеживает состояние скрытой области в форме и устанавливает свойство `display` дескриптора `<DIV>` соответствующим образом. Обратите внимание, что он также изменяет метку на кнопке, чтобы отразить состояние `MoreIsUp`, что можно увидеть на рис. 19.4.

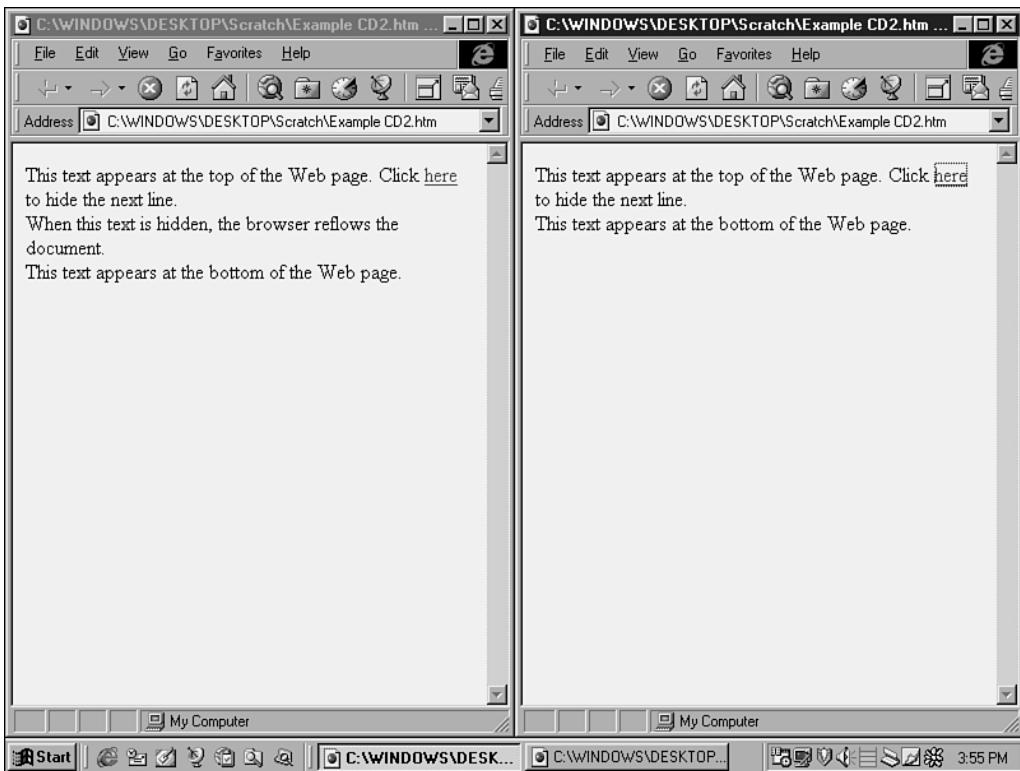


Рис. 19.3. При скрытии элемента браузер отображает Web-страницу в таком виде, будто этого элемента нет вообще

Листинг 19.11. Отображение и сокрытие частей Web-страницы

```

<HTML>
<SCRIPT LANGUAGE=JAVASCRIPT>

MoreIsUp = false;

// Вывести скрытую форму, если она еще не выведена.
// В противном случае, скрыть ее.
// Изменить также метку на кнопке, чтобы отразить
// текущее состояние скрытой формы.

function OpenMore()
{
    MoreIsUp = !MoreIsUp;
    More.style.display = MoreIsUp ? "" : "none";
    FEEDBACK.FEEDBACK_MORE.value = MoreIsUp ? "Less<<" : "More>>";
}
</SCRIPT>

</HEAD>
<BODY>

<FORM NAME=FEEDBACK METHOD=POST ACTION="mailto:jerry@honeycutt.com">

```

```

<TABLE CELLPADDING=10>
<TR>
  <TD VALIGN=TOP>
    <B>Адрес вашей электронной почты:</B><BR>
    <INPUT NAME=FEEDBACK_MAIL TYPE=TEXT SIZE=40>
  </TD>
  <TD VALIGN=TOP>
    <B>Как вы нашли наш узел:</B><BR>
    <SELECT NAME=FEEDBACK_HOW SIZE=1>
      <OPTION VALUE=1>AltaVista
      <OPTION VALUE=2>Excite
      <OPTION VALUE=3>Lycos
      <OPTION VALUE=4>Yahoo!
      <OPTION VALUE=5>WebCrawler
      <OPTION VALUE=6>Friend
      <OPTION VALUE=7>Other Link
    </SELECT>
  </TD>
<TR>
  <TD VALIGN=TOP ROWSPAN=2>
    <B>Что вы думаете о нашем узле:</B><BR>
    <TEXTAREA NAME=FEEDBACK_MEMO COLS=45 ROWS=8>
    </TEXTAREA>
  </TD>
  <TD VALIGN=TOP>
    <B>Наш рейтинг?</B><BR>
    <TABLE BORDER=1>
      <TR ALIGN=CENTER>
        <TH></TH><TH>Yes</TH><TH>No</TH>
      </TR>
      <TR ALIGN=LEFT>
        <TD>Загрузка быстрая?
        <TD>
          <INPUT NAME=FEEDBACK_SPEED TYPE=RADIO>
        </TD>
        <TD>
          <INPUT NAME=FEEDBACK_SPEED TYPE=RADIO>
        </TD>
      </TR>
      <TR ALIGN=LEFT>
        <TD>Интересная графика?
        <TD>
          <INPUT NAME=FEEDBACK_GRAPHIC TYPE=RADIO>
        </TD>
        <TD>
          <INPUT NAME=FEEDBACK_GRAPHIC TYPE=RADIO>
        </TD>
      </TR>
      <TR ALIGN=LEFT>
        <TD>Содержимое вам подходит?
        <TD>
          <INPUT NAME=FEEDBACK_CONTENT TYPE=RADIO>
        </TD>
      </TR>
    </TABLE>
  </TD>

```

```

        <TD>
            <INPUT NAME=FEEDBACK_CONTENT TYPE=RADIO>
        </TD>
    </TR>
</TABLE>
</TD>
</TR>
<TR ALIGN=RIGHT>
<TD>
    <TABLE WIDTH=100%>
        <TD ALIGN=LEFT>
            <INPUT NAME=FEEDBACK_MORE TYPE=BUTTON VALUE="More>>" 
                OnClick="OpenMore () ">
        </TD>
        <TD>
            <INPUT NAME=FEEDBACK_RESET TYPE=RESET VALUE=Clear>
            <INPUT NAME=FEEDBACK_SUBMIT TYPE=SUBMIT VALUE=Submit>
        </TD>
    </TABLE>
</TD>
</TR>
</TABLE>

<!-- Этот участок содержит скрытую часть формы, которую
пользователь увидит, если щелкнет на More&gt;&gt;. Обработчик события
в начале этого файла показывает скрытую часть формы. --&gt;

&lt;DIV ID=More STYLE="display: none"&gt;
    &lt;TABLE CELLPADDING=10&gt;
        &lt;TR&gt;
            &lt;TD&gt;
                &lt;B&gt;Введите URL вашей начальной страницы:&lt;/B&gt;&lt;BR&gt;
                &lt;INPUT NAME=FEEDBACK_URL TYPE=TEXT SIZE=60&gt;
            &lt;/TD&gt;
            &lt;TD&gt;
                &lt;B&gt;Введите ваш телефонный номер:&lt;/B&gt;&lt;BR&gt;
                &lt;INPUT NAME=FEEDBACK_PHONE TYPE=TEXT SIZE=32&gt;
            &lt;/TD&gt;
        &lt;/TR&gt;
    &lt;/TABLE&gt;
&lt;/DIV&gt;

&lt;/FORM&gt;

&lt;/BODY&gt;
&lt;/HTML&gt;
</pre>

```

Создание сворачивающейся/ развертывающейся иерархической структурь

В листинге 19.12 используется свойство `display` для формирования сворачивающейся иерархической структуры. Первоначально видны разделы самого высокого уровня. Если пользователь щелкает на заголовке свернутого раздела, браузер разворачивает его.

чивает следующий уровень иерархической структуры. Если пользователь щелкает на заголовке развернутого раздела, браузер сворачивает его содержимое.

Вы можете формировать иерархическую структуру, используя любые дескрипторы HTML и вкладывая дескрипторы подразделов в дескрипторы родительских разделов. В этом примере используется дескриптор , автоматически выравнивающий уровни иерархической структуры. Для того чтобы сценарии работали, не забывайте давать каждому дескриптору соответствующий идентификатор. Вот как выглядят идентификаторы для простой иерархической структуры, представляющей основной раздел, два подраздела и три подраздела третьего уровня:

Иерархическая структура

```
Раздел1
    Раздел11
    Раздел12
    Раздел13
Раздел2
    Раздел21
    Раздел22
    Раздел23
```

The screenshot shows a Microsoft Internet Explorer window displaying a web form. The title bar reads "D:\KIV without Co\sample.html - Microsoft Internet Explorer". The menu bar includes "Файл", "Правка", "Вид", "Переход", "Избранное", and "Справка". The toolbar contains standard icons for back, forward, search, and file operations. The address bar shows the URL "D:\KIV without Co\sample.html". The page content is a form with the following fields:

- "Адрес вашей электронной почты:" (Email address) - A text input field.
- "Как вы нашли наш узел:" (How did you find our site?) - A dropdown menu set to "AltaVista".
- "Что вы думаете о нашем узле:" (What do you think about our site?) - A large text area for comments.
- "Наш рейтинг?" (Our rating?) - A section containing three questions with "Yes" and "No" radio buttons:
 - Загрузка быстрая? (Load fast?)
 - Интересная графика? (Interesting graphics?)
 - Содержимое вам подходит? (Content suits you?)
- "Less<<" and "Clear" buttons.
- "Submit" button.
- "Ведите URL вашей начальной страницы:" (Enter URL of your home page) - A text input field.
- "Ведите ваш телефонный номер" (Enter your phone number) - A text input field.

Рис. 19.4. Форма со скрываемыми элементами

Для того чтобы идентифицировать элементы, которые принадлежат иерархической структуре, присвойте значение `Outline` свойству `CLASS` каждого элемента. Вы должны также установить свойство `display` каждого подэлемента равным `none` для того, чтобы его сначала не было видно. Вот как выглядит типичная строка в иерархической структуре:

```
<P ID="Outline12" CLASS="Outline" STYLE="display: none">Подраздел</P>
```

И последнее, что вы должны сделать, — поместить всю иерархическую структуру в дескриптор `<DIV>`. Дескриптор должен сопоставить функцию `OutlineClick()` событию `onClick`. Так как событие всплывает от каждого элемента иерархической структуры к дескриптору `<DIV>`, требуется один сценарий, чтобы обработать событие на всех уровнях иерархической структуры.

Сценарий содержит три функции, организующие иерархическую структуру.

- `OutlineClick()`. Эта функция — обработчик события для дескриптора `<DIV>`, который окружает всю иерархическую структуру. Функция сначала проверяет, принадлежит ли объект, вызвавший событие, классу `Outline`, связанному с ним. Затем она проверяет раздел на наличие подразделов, используя набор `children`. Если объект имеет подразделы и они в данный момент не отображены, то для отображения всех подразделов она вызывает функцию `Expand()`, передавая собственный идентификатор в качестве параметра.
- `Expand()`. Эта функция начинается с установки счетчика равным 1 и прибавления значения счетчика к идентификатору, взятому в качестве параметра. Если элемент с этим объединенным идентификатором существует, функция показывает его. Затем функция увеличивает счетчик на 1 и повторяет процесс до тех пор, пока поиск подразделов для данного раздела не завершается.
- `Collapse()`. Эта функция немного более трудна для понимания. В связи с тем, что она должна сворачивать любое число уровней иерархической структуры, она является рекурсивной. Она принимает идентификатор уровня, который нужно свернуть, в качестве параметра. Если уровень не имеет никаких подуровней, происходит просто возврат. Если уровень имеет подуровни, каждый из них сворачивается. Для того чтобы обеспечить сворачивание всех элементов, находящихся на более низком уровне, функция рекурсивно вызывает себя с идентификатором порожденного элемента.

Листинг 19.12. Создание сворачивающейся/развертывающейся иерархической структуры

```
<HTML>

<SCRIPT LANGUAGE=JAVASCRIPT>
function OutlineClick()
{
    var Source, Targets, i;

    Source = window.event.srcElement;
    if( Source.className == "Outline" )
    {
        Targets = document.all(Source.id).children;
        if( Targets.length != 0 && Targets[0].style.display == "none")
            Expand( Source.id );
        else
            Collapse( Source.id );

        window.event.cancelBubble = true;
    }
}

function Expand( Level )
{
```

```

var i, Target;

i = 1;
Target = document.all( Level + i.toString() );
while( Target != null )
{
    Target.style.display = "";
    i++;
    Target = document.all( Level + i.toString() );
}
}

function Collapse( Level )
{
    var i, Target;

    if( document.all( Level ).children.length == 0 )
        return false;
    else
    {
        i = 1;
        Target = document.all( Level + i.toString() );
        while( Target != null )
        {
            Collapse( Target.id );
            Target.style.display = "none";
            i++;
            Target = document.all( Level + i.toString() );
        }
    }
}
</SCRIPT>

<STYLE>
.Outline { cursor: hand }
</STYLE>

<DIV onClick="OutlineClick();">
    <UL ID="OL" CLASS="Outline" >
        Раздел (щелкните для развертывания раздела)
        <UL ID="OL1" CLASS="Outline" STYLE="display: none">
            Первый раздел
            <P ID="OL11" CLASS="Outline" STYLE="display: none">
                Можно создавать сколь угодно сложную иерархическую
                структуру.<BR>
                Для этого необходимо использовать соответствующие<BR>
                идентификаторы и включить каждый элемент раздела<BR>
                в класс Outline, тогда структура будет работать сама.<BR>
                Скопируйте сценарии из этого примера в ваш файл HTML.<BR>
                Вы можете использовать этот сценарий с любыми дескрипторами
                HTML.<BR>
                Здесь используется дескриптор UL для обеспечения <BR>
                автоматического отступа каждого уровня.
            </P>
            <UL ID="OL12" CLASS="Outline" STYLE="display: none">
                Подраздел первого раздела
            </UL>
            <UL ID="OL13" CLASS="Outline" STYLE="display: none">
                Второй подраздел первого раздела
            </UL>
        </UL>
    </UL>
</DIV>

```

```

<UL ID="OL131" CLASS="Outline" STYLE="display: none">Test
1</UL>
<UL ID="OL132" CLASS="Outline" STYLE="display: none">Test
2</UL>
</UL>
</UL>
</DIV>

</HTML>

```

Позиционирование элементов HTML на Web-странице

Internet Explorer и Communicator используют позиционирование CSS, о котором вы узнали в главе 18 “Размещение элементов HTML”. Вы можете написать сценарий для позиционирования элемента, применяя объект `style`. Используйте свойства `left`, `posLeft`, `top`, `posTop` и `zIndex` для того, чтобы управлять позицией элементов сценария.

Например, используйте следующий код, чтобы изменить позицию элемента на абсолютную позицию 30, 45:

```

Element.style.left = 30;
Element.style.top = 45;

```



Избегайте применения слоев Netscape — применяйте вместо них позиционирование с помощью каскадных листов стиля. Они обладают такими же возможностями и совместимы как с Internet Explorer, так и с Communicator.

Примеры в следующих разделах показывают, как использовать позиционирование для создания удивительных эффектов на Web-странице. Обратите внимание, что в большинстве примеров позиционирования используется таймер. В объектной модели документа вы можете устанавливать таймер, который вызывает событие в указанное время. В приведенной ниже строке кода показано, как установить таймер. Первый параметр для `setTimeout()` — имя функции, которую таймер должен вызывать по истечении времени, указанного таймеру, в данном случае — `ding()`. Второй параметр — число миллисекунд для установки таймера. Не забудьте, что в одной секунде имеется 1000 миллисекунд. Третий параметр — язык программирования, используемый функцией, указанной в первом параметре.

```
Timer = window.setTimeout( "Ding()", 500, "JAVASCRIPT" );
```

Создание простой мультиплексии

Создание простых мультиплексий с помощью комбинации позиционирования CSS и таймера объектной модели документа показано в листинге 19.13. Объект, который мы оживляем в этом примере, — дескриптор `<DIV>`, находящийся в конце листинга. Этот дескриптор имеет идентификатор `Ani` и его свойство `position` установлено равным `absolute`.

При каждом событии таймера, связанный с ним обработчик события ding() перемещает дескриптор <DIV> немного дальше. Переменные Hdir и Vdir указывают горизонтальное и вертикальное направления движения объекта. Для того чтобы, например, вычислить горизонтальное смещение, Hdir умножается на 10, и результат добавляется к текущей левой позиции в качестве смещения. Обработчик события должен снова установить таймер, потому что таймер работает только один раз после установки.

В этом примере имеется одна особенность. Функция ChangeDirection() связана с событием onClick документа. Когда пользователь щелкает на экране, ChangeDirection() проверяет объект event, чтобы определить, где именно пользователь щелкнул. ChangeDirection() изменяет Hdir и Vdir, чтобы указать направление относительно текущей позиции объекта, на котором сделан щелчок.

Листинг 19.13. Создание простой мультипликации

```
<HTML>
<BODY onClick="ChangeDirection()">
<SCRIPT LANGUAGE=JAVASCRIPT>

    var Timer;
    var Hdir = 1;
    var Vdir = 1;

    function Ding()
    {
        Ani.style.posTop += Vdir * 10;
        Ani.style.posLeft += Hdir * 10;
        Timer = window.setTimeout( "Ding()", 500, "JAVASCRIPT" );
    }

    Timer = window.setTimeout( "Ding()", 500, "JAVASCRIPT" );

    function ChangeDirection()
    {
        if( window.event.clientX < Ani.style.posLeft )
            Hdir = -1;
        else
            Hdir = 1;

        if( window.event.clientY < Ani.style.posTop )
            Vdir = -1;
        else
            Vdir = 1;;
    }
</SCRIPT>

<DIV ID=Ani STYLE="position: absolute; left: 0; top: 0">
    Этот текст передвигается по экрану.
</DIV>

</BODY>
</HTML>
```



Вы можете создавать намного более сложную мультипликацию, чем показанная в этом примере. Заполните массив storyboard, который указывает позицию и время, соответствующее этой позиции. Функция ding() может устанавливать позицию и время, используя storyboard. Это дает вам полный контроль над движением объекта по экрану.

Создание оперативной справки для гиперссылок

Вы можете предоставить пользователю справку в стиле ToolTip для гиперссылок на вашей Web-странице (или фактически для любого элемента на ней). Получение справки происходит следующим образом.

- Если указатель мыши находится на гиперссылке более секунды, то рядом с ней открывается окно справки.
- Окно справки немедленно исчезает, когда пользователь перемещает указатель вне ссылки.
- Если указатель находится на ссылке более десяти секунд, окно справки исчезает само.

В листинге 19.14 демонстрируется применение этой возможности (рис. 19.5). Единственный стиль, определенный в этом примере, `Help`, описывает, как выглядит окно справки. В данном случае это черный текст на желтом фоне. Кроме того, элемент окружен рамкой. Дескриптор `<DIV>`, который вы видите в начале листинга, представляет собой элемент, который используется в качестве окна справки. Это не приводит к повторному выводу Web-страницы, поэтому окно справки выглядит, скорее, как всплывающее окно.

Каждая ссылка на Web-странице связана с событиями `onMouseOver` и `onMouseOut`. Событие `OnMouseOver` связано с функцией `FlyOver()`, которая принимает в качестве параметра текст, печатающийся в окне справки. Событие `OnMouseOut` связано с функцией `ClearFlyOver()`, которая прячет окно справки. Вот как эти функции работают при выводении справки.

- `FlyOver()`. Эта функция устанавливает интервал таймера в одну секунду и вызывает функцию `DoFlyOver()` после того, как интервал истекает. Она также записывает текст справки и координаты указателя мыши в глобальные переменные, потому что эти значения не доступны вне события `onMouseOver`.
- `DoFlyOver()`. Эта функция определяет текст в дескрипторе `<DIV>` справки, затем устанавливает позицию дескриптора `<DIV>` с маленьким смещением от указателя и выводит справку, присваивая свойству `display` значение, равное пустой строке. Далее она устанавливает таймер, который вызывает `ClearFlyOver()` приблизительно через десять секунд.
- `ClearFlyOver()`. Эта функция прячет окно справки, устанавливая свойство `display` равным `none`.

Листинг 19.14. Предоставление справки по гиперссылке

```
<HTML>
<STYLE>
.Help {
    position: absolute; posTop: 0; posLeft: 0;
    border-width: thin; border-style: solid;
    background-color: yellow; color: black;
    height=20; width=240;
}
</STYLE>
```

```

<DIV ID=FOArea CLASS="Help" STYLE="display: none">
</DIV>

<SCRIPT LANGUAGE=JAVASCRIPT>
  var HelpX, HelpY, HelpText;
  var ToShow, ToClear;

  function DoFlyOver()
  {
    if( ToClear != -1 ) window.clearTimeout( ToClear );

    FOArea.innerText = HelpText;
    FOArea.style.posLeft = HelpX + 8;
    FOArea.style.posTop = HelpY + 8;
    FOArea.style.display = "";

    ToClear = setTimeout( "ClearFlyOver()", 10000, "JAVASCRIPT" );
  }

  function ClearFlyOver()
  {
    FOArea.style.display = "none";
  }

  function FlyOver( Text )
  {
    HelpText = Text;
    HelpX = window.event.clientX;
    HelpY = window.event.clientY;
    ToShow = setTimeout( "DoFlyOver()", 1000, "JAVASCRIPT" );
  }
</SCRIPT>

<A onMouseOut="ClearFlyOver();"
onMouseOver="FlyOver( 'Открыть начальную страницу Джерри' );"
HREF="http://rampages.onramp.net/~jerry">Джерри Хоникатт</A><BR>

<A onMouseOut="ClearFlyOver();"
onMouseOver="FlyOver( 'Открыть страницу Microsoft' );"
HREF="http://www.microsoft.com">Microsoft</A>

</HTML>

```

Чтение и запись cookie с использованием объектной модели

Большинством прикладных программ, которые вы запускаете на своем компьютере, запоминаются параметры последней настройки; их можно восстановить, открыв соответствующее диалоговое окно. К сожалению, формы HTML не делают этого. Некоторые браузеры “помнят” параметры настройки в течение текущего сеанса, однако когда вы закрываете браузер, параметры настройки теряются.

Вы можете имитировать запоминание настроек с помощью файла cookie. Этот файл применяют для сохранения информации на компьютере пользователя; такую

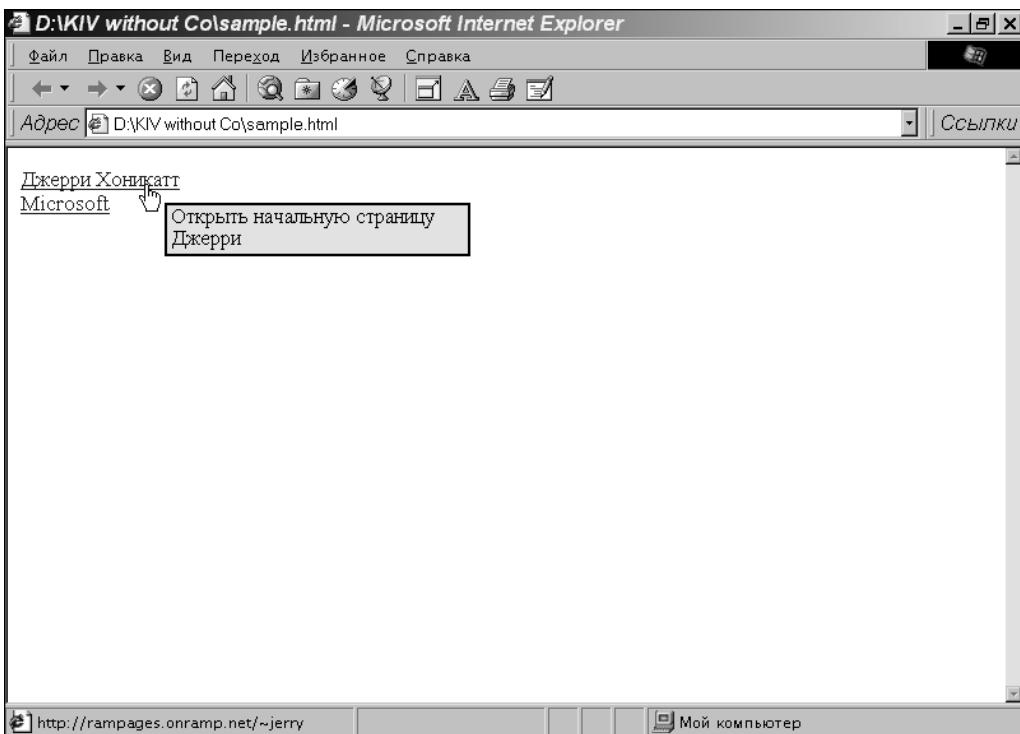


Рис. 19.5. Вы можете предоставить оперативную справку для любого дескриптора на Web-странице

информацию можно прочитать и использовать во время повторного сеанса (правда, зачастую это вызывает у некоторых пользователей большой испуг). Встроенные объекты броузера не обеспечивают все функции, необходимые для использования cookie — вам потребуются дополнительные функции записи и чтения cookie. Сценарий, показанный в листинге 19.15, содержит все функции, которые нужны для применения cookie в HTML-документах.

- `SetCookie()`. Эта функция сохраняет значение в cookie документа с именем `Name` и значением `Value`. Она устанавливает дату устарения равной дате, содержащейся в `Expire`. Если вы не указываете дату устарения, значение остается корректным только для текущего сеанса. Обратите внимание, что дата должна форматироваться следующим образом:
День недели, DD-MMM-YYYY HH:MM:SS GMT
- `GetCookie()`. Эта функция ищет величину по имени `Name` и возвращает ее значение. Она проверяет каждое поле в cookie документа, пока не находит нужное. Затем она вызывает `GetValue()` для чтения значения.
- `GetValue()`. Эта функция читает значение поля, которое расположено с указанным сдвигом `Offset` от начала.

Остальные функции в этом сценарии, `WriteCookies()` и `GetCookies()`, являются специфическими для формы в листинге 19.15. `WriteCookies()` записывает значение cookie для каждого поля в форме. `GetCookies()` читает все значения из cookie документа и устанавливает значения полей в форме.

Теперь у вас есть функции, необходимые для сохранения значения формы в cookie документа, которые вы должны соединить. Наилучшее место в сценарии для

записи значений в cookie находится в функции проверки правильности формы: `IsValid()`. Добавьте следующие две строки HTML в конец этой функции (перед последним оператором `return`); затем, когда пользователь посыпает правильно заполненную форму, функция проверки правильности сохранит значения формы в cookie.

```
If (blnValid)
    WriteCookies ();
```

А как насчет предварительного заполнения формы значениями из cookie? Это просто. Измените дескриптор `<BODY>` таким образом, как показано далее. Эта строка связывает событие окна `OnLoad` с функцией `GetCookies()`, которая предварительно заполняет форму значениями, взятыми из cookie каждый раз, когда пользователь загружает документ HTML, содержащий форму.

```
<BODY OnLoad="GetCookies()">
```

Листинг 19.15. Чтение и запись cookie с использованием объектной модели

```
<HTML>
<SCRIPT LANGUAGE=JAVASCRIPT>

MoreIsUp = false;

// Вывести скрытую форму, если она еще не выведена. В противном
//случае, спрятать ее.
// Изменить также метку на кнопке, чтобы отразить текущее состояние
//скрытой формы.

function OpenMore()
{
    MoreIsUp = !MoreIsUp;
    More.style.display = MoreIsUp ? "" : "none";
    FEEDBACK.FEEDBACK_MORE.value = MoreIsUp ? "Less<<" : "More>>";
}
</SCRIPT>

<SCRIPT LANGUAGE=JAVASCRIPT>

// Прочитать значение cookie с указанным смещением.

function GetValue( Offset )
{
    var End = document.cookie.indexOf (";", Offset);
    if( End == -1 )
        End = document.cookie.length;

    // Возвратить часть cookie, начиная со смещения
    // и заканчивая ";".

    return unescape( document.cookie.substring( Offset, End) );
}

// Возврат значения cookie с заданным именем.

function GetCookie( Name )
{
    var Len = Name.length;
```

```

// Проверка всех подстрок, имеющих длину, равную длине имени cookie
// на совпадение. Если найдено, читать значение и возвратить его.

var i = 0;
while( i < document.cookie.length )
{
    var j = i + Len + 1;
    if( document.cookie.substring( i, j ) == (Name + "=") )
        return GetValue( j );
    i = document.cookie.indexOf( " ", i ) + 1;
    if( i == 0 )
        break;
}
return null;
}

// Создать или изменить cookie, указывая имя и значение. Имя и значение
// необходимы, а дата устаревания нет. Отметьте, что, если дата
// устаревания не указана, cookie будет существовать только в течение
// текущего сеанса.

function SetCookie( Name, Value, Expire )
{
    document.cookie = Name + "=" + escape( Value ) + ";expires=" + Expire;
}

// Записать все cookie для формы FEEDBACK.

function WriteCookies()
{
    var Expire = "Friday, 25-Feb-2000 12:00:00 GMT";

    with( document.FEEDBACK )
    {
        SetCookie( "Mail", FEEDBACK_MAIL.value, Expire );
        SetCookie( "How", FEEDBACK_HOW.selectedIndex, Expire );
        SetCookie( "Memo", FEEDBACK_MEMO.value, Expire );
        SetCookie( "Speed", FEEDBACK_SPEED[0].checked ? "1" : "0", Expire
    );
        SetCookie( "Content", FEEDBACK_CONTENT[0].checked ? "1" : "0",
        Expire );
        SetCookie( "Graphic", FEEDBACK_GRAPHIC[0].checked ? "1" : "0",
        Expire );
    }
}

// Заполнить формы значениями из cookie.

function GetCookies()
{
    with( document.FEEDBACK )
    {
        FEEDBACK_MAIL.value = GetCookie( "Mail" );
        FEEDBACK_HOW.selectedIndex = GetCookie( "How" );
        FEEDBACK_MEMO.value = GetCookie( "Memo" );
        FEEDBACK_SPEED[0].checked = (GetCookie( "Speed" ) == "1");
        FEEDBACK_SPEED[1].checked = (GetCookie( "Speed" ) == "0");
        FEEDBACK_CONTENT[0].checked = (GetCookie( "Content" ) == "1");
    }
}

```

```

        FEEDBACK_CONTENT[1].checked = (GetCookie( "Content" ) == "0");
        FEEDBACK_GRAPHIC[0].checked = (GetCookie( "Graphic" ) == "1");
        FEEDBACK_GRAPHIC[1].checked = (GetCookie( "Graphic" ) == "0");
    }
}

function IsValid()
{
    blnValid = true;

    with( document.FEEDBACK )
    {
        if( FEEDBACK_MAIL.value == "" )
        {
            window.alert( "Введите, пожалуйста, почтовый адрес" );
            blnValid = false;
        }

        if( !(FEEDBACK_SPEED[0].checked || FEEDBACK_SPEED[1].checked) ||
            !(FEEDBACK_CONTENT[0].checked || FEEDBACK_CONTENT[1].checked) ||
            !(FEEDBACK_GRAPHIC[0].checked || FEEDBACK_GRAPHIC[1].checked) )
        {
            window.alert( "Пожалуйста, выберите Yes или No для каждой оценки" );
            blnValid = false;
        }
    }

    if( blnValid )
        WriteCookies();

    return blnValid;
}

```

</SCRIPT>

<BODY onLoad="GetCookies();">

<FORM NAME=FEEDBACK METHOD=POST ACTION="" onSubmit="return IsValid();">

| | |
|--|---|
| Адрес вашей электронной почты: <input name="FEEDBACK_MAIL" size="40" type="text"/> | Как вы нашли наш узел?: <select name="FEEDBACK_HOW" size="1"> <option value="1">AltaVista</option> <option value="2">Excite</option> <option value="3">Lycos</option> <option value="4">Yahoo!</option> <option value="5">WebCrawler</option> <option value="6">Friend</option> <option value="7">Other Link</option> </select> |
|--|---|

```

<TD VALIGN=TOP ROWSPAN=2>
    <B>Что вы думаете о нашем узле?:</B><BR>
    <TEXTAREA NAME=FEEDBACK_MEMO COLS=45 ROWS=8>
    </TEXTAREA>
</TD>
<TD VALIGN=TOP>
    <B>Ваша оценка?</B><BR>
    <TABLE BORDER=1>
        <TR ALIGN=CENTER>
            <TH></TH><TH>Yes</TH><TH>No</TH>
        </TR>
        <TR ALIGN= CENTER>
            <TD ALIGN=LEFT>
                Загрузка достаточно быстрая?
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_SPEED TYPE=RADIO>
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_SPEED TYPE=RADIO>
            </TD>
        </TR>
        <TR ALIGN= CENTER>
            <TD ALIGN=LEFT>
                Рисунки интересные?
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_GRAPHIC TYPE=RADIO>
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_GRAPHIC TYPE=RADIO>
            </TD>
        </TR>
        <TR ALIGN= CENTER>
            <TD ALIGN=LEFT>
                Содержимое подходящее?
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_CONTENT TYPE=RADIO>
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_CONTENT TYPE=RADIO>
            </TD>
        </TR>
    </TABLE>
</TD>
</TR>
<TR ALIGN=RIGHT>
    <TD>
        <TABLE WIDTH=100%>
            <TD ALIGN=LEFT>
                <INPUT NAME=FEEDBACK_MORE TYPE=BUTTON VALUE="More>>" On-
Click="OpenMore () ">
            </TD>
            <TD>
                <INPUT NAME=FEEDBACK_RESET TYPE=RESET VALUE=Clear>
                <INPUT NAME=FEEDBACK_SUBMIT TYPE=SUBMIT VALUE=Submit>
            </TD>
        </TABLE>
    </TD>

```

```

        </TD>
    </TR>
</TABLE>

<!-- Этот участок содержит скрытую часть формы, которую
пользователь видит, если щелкнет на More-->. Обработчик
события в начале этого файла показывает уровень. -->

<DIV ID=More STYLE="display: none">
    <TABLE CELLPADDING=10>
        <TR>
            <TD>
                <B>Введите адрес URL вашей начальной страницы: </B><BR>
                <INPUT NAME=FEEDBACK_URL TYPE=TEXT SIZE=60>
            </TD>
            <TD>
                <B>Номер вашего телефона: </B><BR>
                <INPUT NAME=FEEDBACK_PHONE TYPE=TEXT SIZE=32>
            </TD>
        </TR>
    </TABLE>
</DIV>

</FORM>

</BODY>
</HTML>

```

Различия между Internet Explorer и Communicator

W3C находится в процессе стандартизации объектной модели документа. Internet Explorer и Communicator используют различные объектные модели, что затрудняет создание совместимых сценариев. Различие этих объектных моделей связано с концепцией, которую производители называют динамическим HTML.

- Microsoft. Объектная модель документа предоставляет доступ к каждому элементу на Web-странице, включая свойства листа стилей. Так же и каждый элемент предоставляет доступ к соответствующим свойствам, методам, событиям и наборам.
- Netscape. Объектная модель документа не идет так далеко, как у Microsoft. Она предоставляет доступ к ограниченному числу элементов и не предоставляет доступ к свойствам листа стилей вообще. При использовании Communicator вы не можете изменять внешний вид элементов после того, как Web-страница загружена.

Оба браузера предоставляют доступ к традиционным объектам: window, navigator, document, history, location, anchor, applet, area, form, element, options, image и link. Communicator также предоставляет доступ к дополнительному объекту layer, который позволяет работать из программ с собственными слоями браузера. Internet Explorer предоставляет доступ к ряду объектов, которые являются уникальными, включая all, cell, rows, children, embeds, plugins, external, filters, rules, style, styleSheet, styleSheets, screen, scripts, TextRange и userProfile.