Операции с окнами (объект window)

При работе браузера всегда существует объект `window` – это текущее окно. Тем не менее, можно создать еще одно или даже несколько окон, которые будут существовать в рамках одной объектной модели. При этом ссылка всегда `window` относится к текущему окну, то есть к окну, содержащему исполняемый JavaScript-код. Обращаться к другим окнам можно с помощью имен, которые задаются при создании окна.

Создание нового окна браузера (метод open)

Чтобы создать новое окно, нужно применить метод `window.open()`. Типичные инструкции, вводящие этот метод, имеют следующий вид:

```
window.open("URL","Namewin","Features");
```

либо

```
var win=window.open("URL","name","features");
```

Согласно первой записи создается новый объект `window` без возможности ссылки на него. При использовании второй записи данный об объекте сохраняются в переменной `win`. То есть можно сослаться на созданный объект, использовать его методы и свойства.

При операциях в текущем окне в записи `window.open()` объект верхнего уровня (`window`) можно не указывать.

Метод `open()` создает окно, которое относится к типу немодальных окон. В отличие от модальных окон `alert()`, `prompt()`, `confirm()`, немодальные окна не зависят от текущего окна. Программа в немодальном окне выполняется независимо от других окон.

Аргументы метода open()

Метод `open` может иметь аргументы: `open("URL","Namewin","Features")`. Перечислим значения аргументов:

- `URL` обозначает `URL` документа, загружаемого в окне.
- `Name` вводит название окна. Имя окна `Namewin` будет присвоено свойству `name` объекта `win`.
- `Features` обозначает список необязательных опций, которые разделены запятой. В этом списке могут быть параметры размеров окна в пикселах (`width` и `height`) и возможности изменения размеров границы окна (`resizable`), а также параметров наличия панели инструментов (`toolbar`), меню (`menubar`), поля ввода адреса (`location`), строки состояния (`status`).

Все указанные параметры в методе `open()` являются необязательными. Если опущен в списке параметров указатель `URL`, то в новом окне браузера будет открыта пустая страница. Имя также можно не указывать, однако в этом случае нельзя будет использовать данное окно в качестве целевого (`target`) для последующих документов. Наконец, если не заданы параметры окна, используются значения, принятые по умолчанию.

Приведем несколько примеров создания окон. Например, нижеследующий сценарий приведет к созданию трех различных независимых окон:

```
<script language="javascript">
  var win1 = open("","Copy");
  var win = open("D:/js/about.html", "Version", "width=200,
height=150, toolbar=0,status=1, menubar=yes");
  var win3 = open("https://www.mail.ru","Session");
</script>
```

Переменная `win1` будет содержать только данные о пустом окне с именем `Copy`. В окно с именем `Version` (объект `win2`) будет загружен файл `D:/js/about.html`, а в окне `Session` (объект `win3`) будет открыта Web-страница https://www.mail.ru. Для объекта `win2` заданы размеры окна: `width=200`, `height=150`, а также предусмотрены меню (`menubar`) и строка состояния (`status`).

Чтобы включить тот или иной элемент окна (`toolbar`, `menubar`, `location`, `status`), используются значения параметров `yes`, `1` или само название элемента. Для отключения элемента – значения `no` или `0`. Например, для вывода панели инструментов можно использовать любую из следующих инструкций:

```
window.open("...","...","toolbar=1");
open("...","...","toolbar=yes");
open("...","...","toolbar");
```

Переключение между окнами (методы focus() и blur())

Если на экране открыто несколько окон, вы можете для переключения между ними воспользоваться методом `window.focus()`. Окно, получившее фокус, будет отличаться от других окон цветной подсветкой строки заголовка. Лишить окно фокуса можно с помощью метода `window.blur()`.

Вызов метода `focus()` можно связать с выполнением соответствующего обработчика событий – `onfocus`. При этом важно учитывать, что событие `onfocus` запускается, если фокус получает прежде не активизированное окно. Если же окно активно, то при вызове метода `focus()` уже не будет генерировать событие `onfocus`.

Присвоение имени окну (свойство name)

Проще всего выполняются манипуляции с именем окна. Вообще говоря, все окна, создаваемые в браузере, остаются без имени, пока оно им не присвоено с помощью специальной директивы – присвоения строкового значения свойству `name` объекта `window`. Следующие пять строк приводят к одному и тому же результату, они присваивают имя `MyWin` текущему окну:

```
window.Window.name = "MyWin"
window.self.name = "MyWin"
window.name = "MyWin"
self.name = "MyWin"
name = "MyWin"
```

В этих записях использовано свойство `self` объекта `window`, которое возвращает окно. Убедиться в том, что любая из директив присваивает имя `MyWin`, можно с помощью пары строк в сценарии, например,

```
self.name = "MyWin";
alert(name);
```

которые в диалоге сообщения выведут имя окна «`MyWin`». Аналогично описанному способу присваивается имя фрейму.

Перемещение окна браузера

Для перемещения текущего окна браузера в объекте `window` предусмотрено два следующих метода:

- `window.moveTo()` – этот метод получает два параметра, отвечающих перемещению левого верхнего угла окна в точку с заданными координатами. Например, метод `window.moveTo(100,200)` переместит окно в положение, при котором левый верхний угол окна будет расположен на расстоянии 100 пикселов от левой кромки экрана и в 200 пикселов от верхней кромки экрана;
- `window.moveBy()` – этот метод осуществляет перемещение окна на заданное расстояние. Например, `window.moveTo(10,5)` переместит окно на 10 пикселов вправо и на 5 пикселов вниз. Причем возможно как положительные, так и отрицательные значения аргументов. Перемещение будет осуществляться при каждом выполнении метода.

Приведем как пример строку кода для выполнения перемещения методом `MoveBy()`:

```
<input type="button" value="Переместить"
onclick = "window.moveBy(10,5);">
```

Изменение размеров окна (методы resizeTo и resizeBy)

Объект `window` позволяет изменять размер текущего окна. Для этого предназначены два метода `window.resizeTo()` и `window.resizeBy()`, которые аналогичны методам перемещения окна.

- Метод `window.resizeTo()` приводит к изменению окна до указанных размеров.
- Метод `window.resizeBy()` дает приращения размеров окна на указанную величину.

Печать Web-страницы (метод Print)

Распечатать Web-документ можно с помощью кнопки **Печать** на панели браузера. Однако аналогичное средство имеется и в JavaScript, Оно представлено методом `window.print()`. Например, разместим на странице кнопку «Печать» с помощью кода:

```
<input type="button" value="Печать" onclick="window.print();">
```

При нажатии этой кнопки будет появляться обычное диалоговое окно печати.

Закрытие окна (метод close)

Окна, созданные методом `open`, могут быть закрыты методом `close`. Так, для закрытия окна используются следующие записи:

```
window.close();
window.close(Namewin);
```

Первая из этих директив закрывает окно, а вторая – окно с именем `Namewin`.

Отметим, что факт закрытия окна записывается в свойство `closed` объекта `window`. На первый взгляд весьма парадоксальная ситуация: окно закрыто, а его свойство еще существует. На самом деле объект `window` после закрытия окна полностью не разрушается: свойство `MyWin.closed` остается доступным (`MyWin` – переменная окна) и возвращает значение `true`.
В случае закрытия последнего окна браузера требуется подтверждение этой операции пользователем. Такая возможность предусмотрена для того, чтобы избежать «хулиганского» закрытия всех окон со стороны JavaScript-программы.

Информация о документе и окне браузера (объект location)

Tot одним дочерним объектом `window` является объект `location`/ Этот объект определяет точную ссылку (URL) на текущий Web-документ, загруженный в окно браузера. Обратиться к `location` можно, к примеру, следующей строкой кода:

```
window.alert("URL данного документа: "+window.location);
```

которая выводит окно сообщения с указанием URL текущего документа.

Свойства объекта location

Свойства объекта `location` предоставляют информацию о URL документа, а методы объекта позволяют перезагружать документ и заменять текущий документ. Запишем общий формат URL, используя обозначения свойств объекта `location`:

```
Protocol://hostname:port/pathname?search#hash
```

Каждое из свойств объекта `location` является строкой для чтения/записи, которая содержит одну или более составляющих URL. Изменение значений свойств объекта `location` приводит к чтению нового URL браузером. Перечислим свойства объекта `location`:

- `protocol` – свойство, которое отвечает типу ресурса; возвращает значения: `http://`, `ftp://`, `file:`, `mailto:` и т.д.
- `hostname` – свойство, содержащее имя хоста;
- `port` – свойство, представляющее номер порта;
- `host` – это свойство, которое возвращает значение hostname, после которого через двоеточие указан номер порта (port);
- `pathname` – свойство, возвращающее путь к ресурсу;

- `search` – свойство, представляющее строку поиска, включающую метку ?;
- `hash` – составляющая URL, которая включает метку закладки #; это свойство возвращает имя якоря в HTML-файле;
- `href` – свойство, отвечающее полному URL.

К каждому из перечисленных свойств нужно обращаться с учетом положения объекта `location` в иерархии объектов. Например, на свойство `href` можно сослаться с помощью одной из следующих записей:

```
window.location.href
location.href
document.links[i].href
```

Перезагрузка и замена текущей страницы (методы объекта location)

Объект `location` имеет два метода: `reload()` и `replace()`. Метод `reload()` позволяет перезагрузить страницу, например, в случае изменения страницы. То есть `reload()` в этом случае вызывает то же действие, что и нажатие кнопки **Обновить** в браузере. Однако, если кнопка **Обновить** приводит к перезагрузке лишь в случае, когда страница была изменена, метод reload() позволяет выполнить перезагрузку даже в случае отсутствия изменений. Для этого при вызове метода используется булевский аргумент: `reload(force)`. Если `force` принимает значение `true`, перезагрузка производится в безусловном порядке, если же аргумент имеет значение `false`, страница перезагружается лишь при наличии изменений.

С помощью второго метода, `replace(url)`, можно заменить текущую страницу на другую, указанную в аргументе `url`. Этот метод часто оказывается полезным для выполнения переадресации на стороне клиента. Приведем пример документа со сценарием запроса о замене текущей страницы:

```
<html>
  <head>
    <title>Замена страницы</title>
  </head>
  <body>
    <!--Исходная страница-->
    <h1>Пример замены текущей страницы</h1>
    <script>
      if(confirm("Нужен ли прогноз погоды на сегодня?"))
location.replace("https://www.gismeteo.by");
    </script>
  </body>
</html>
```

В этом примере после загрузки исходной страницы появится диалоговое окно с запросом (рис. 1). Если нажать кнопку **OK**, текущая страница будет заменена на новую, указанную в качестве аргумента метода `location.replace()`. При нажатии кнопки **Отмена** или клавиши Esc замены текущей страницы не произойдет.

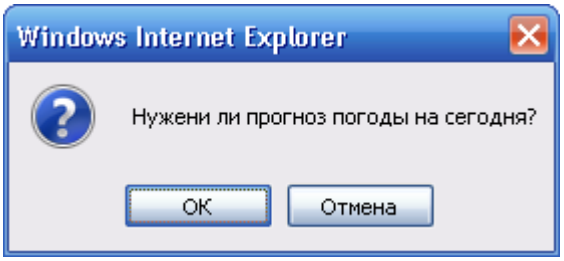

Рис. 1. Запрос на замену текущего окна

При вызове метода `replace()` происходит замена элемента объекта `history`. Поэтому важно иметь в виду, что переход к исходной странице с помощью кнопки **Назад** после действия `replace` будет невозможен.

Управление строкой состояния (свойства status, defaultStatus)

В нижней части окна браузера отображается строка состояния. Содержимое этой строки можно задать с помощью свойств `status` и `defaultStatus`. Первое из этих свойств (`status`) отвечает информации, которая отображается в строке состояния временно, а второе свойство (`defaultStatus`) – информация, которая отображается по умолчанию, то есть сразу после открытия окна до тех пор, пока свойство `defaultStatus` не изменится.

Пример. Изменение содержимого строки состояния

Проиллюстрируем использование свойств `status` и `defaultStatus` строки состояния на примере, показанном на рис.2.

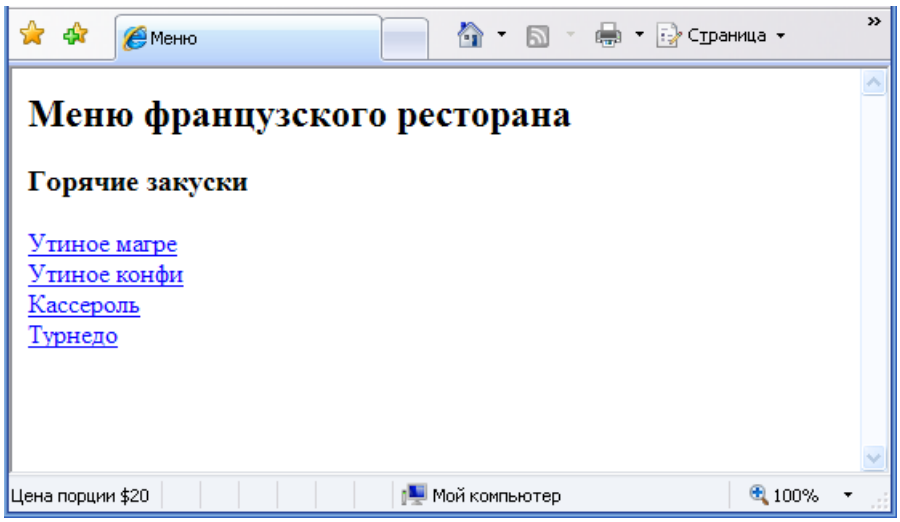

Рис.2. Пример изменения содержимого строки состояния

Листинг этого документа имеет вид:

```
<html>
  <head>
    <title>Меню</title>
  </head>
```

```
  <body>
    <h2>Меню французского ресторана</h2>
    <h3>Горячие закуски</h3>
    <a   href="megre.html"   onmouseover="window.status='Цена
порции $20'; return true;">Утиное магре</a>
    <br>
    <a  href="confite.html"  onmouseover="window.status='Цена
порции $15'; return true;">Утиное конфи</a>
    <br>
    <a href="casserole.html" onmouseover="window.status='Цена
порции $10'; return true;">Кассероль</a>
    <br>
    <a href="tournedos.html" onmouseover="window.status='Цена
порции $20'; return true;">Турнедо</a>
    <br>
    <script>
      window.defaultStatus="Сделайте заказ";
    </script>
  </body>
</html>
```

По умолчанию в строке состояния отображается сообщение «Сделайте заказ». Если же навести курсор мыши на название блюда, в строке состояния появится информация о его стоимости. В приведенном коде использован обработчик события `onmouseover`.

Пример. Цифровые часы в строке состояния

Опишем создание цифровых часов, которые будут выводить информацию в строку состояния. Чтобы построить часы, отображающие текущее время в часах, минутах и секундах, можно воспользоваться следующим кодом:

```
<html>
  <head>
    <title>Цифровые часы в строке состояния</title>
    <script>
      //Aeyrwbz добавления предшествующего нуля
      function ahead0(x){
        return(x<10)?"0"+x.toString():x;
      }
      //Функция отображения времени
      function clock(){
      //Определение даты и времени
        var datetime = new Date();
        var hour = datetime.getHours();
        var minut = datetime.getMinutes();
        var sec = datetime.getSeconds();
        var     time=     ahead0(hour)+':'+ahead0(minut)+':'+
ahead0(sec);
      //вывод времени в строку состояния
      defaultStatus=time;
      //Обновление содержимого строки состояния
      setTimeout("clock()",1000);
      }
```

```
    </script>
  </head>
  <body onload="clock();">
    <h2>Часы в строке состояния</h2>
  </body>
</html>
```

Цифровые часы запускаются сразу после загрузки документа, поскольку в теге `BODY` использован обработчик `onload`.

Переходы между Web-страницами (объект history)

Выполнять переходы между Web-страницами, ранее просматривавшимися пользователем, позволяет объект `history`, который является одним из дочерних объектов окна браузера. `history` представляет собой строковый массив только для чтения, составленный из URL открывавшихся страниц.

Свойства и методы, применяемые для переходов

Доступ к информации о ранее отображавшихся страницах обеспечивается следующими свойствами объекта `history`:

- `history.current` – содержит строку только для чтения, являющуюся адресом текущего Web-документа;
- `history.next` – содержит значение URL документа, который открывался после данного в списке загружавшихся документов; это свойство можно использовать лишь после повторного открытия текущего документа;
- `history.previous` – содержит значение URL документа, который открывался перед данным документом;
- `history.length` – определяет количество адресов в списке в списке загружавшихся документов (в течение текущего сеанса работы).

Приведем также перечень методов объекта `history`:

- `go()` – переход к ранее посещавшееся странице из списка `history`. В качестве аргумента метода используется целое число (положительное для перехода вперед, отрицательное для перехода назад).
- `back()` – отвечает переходу к предыдущей странице, то есть вызов этого метода приводит к тому же результату, что нажатие кнопки **Назад** на панели браузера. Использование метода эквивалентно записи:

  ```
  history.go(-1);
  ```

- `forward()` – выполняет переход вперед в списке посещавшихся страниц. вызов метода эквивалентен записи:

  ```
  history.go(1).
  ```

Создание кнопок «Назад» и «Вперед»

В создаваемые web-страницы часто добавляются кнопки, позволяющие пользователю легко перемещаться по списку открывавшихся документов. Для размещения этих кнопок можно использовать методы объекта history, а в качестве самих кнопок – изображения, заранее заготовленные в виде отдельных графических файлов. Приведем пример документа, содержащего кнопки **Назад** и **Вперед**:

```
<html>
  <head>
    <title>Методы и свойства объекта history</title>
  </head>
  <body>
    <center>
    <h1>Кнопки<br>
    Назад и Вперед</h1>
    <a href="javascript:history.go(-1)">
    <img border=0 src="back.png"></a>
    <a href="javascript:history.go(1)">
    <img border=0 src="forward.png"></a>
    <br><br>
    <INPUT type="button" value="Длина списка history"
onclick="alert(history.length);">
  </body>
</html>
```

На рис. 3 можно видеть, как выглядит эта страница в окне браузера Internet Explorer. Обе кнопки работают, когда текущей является не первая и не последняя открывавшаяся страница.

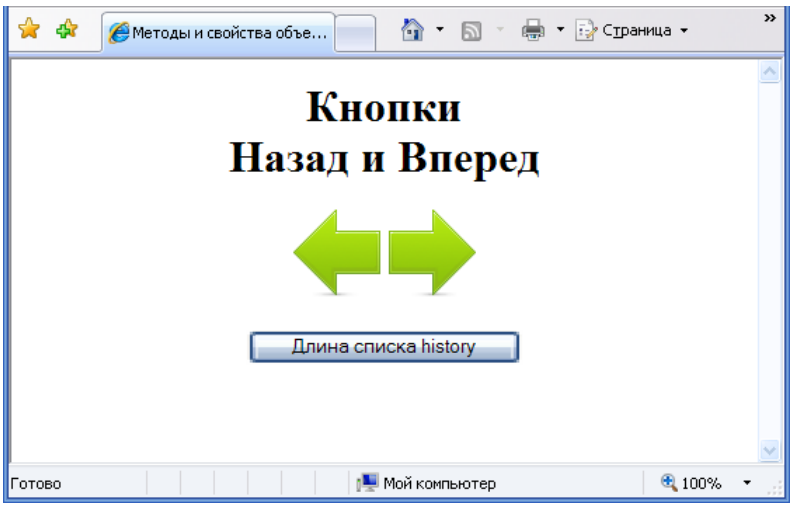

Рис.3. Пример использования методов и свойств объекта history

Если вы хотите получить информацию о количестве записей в журнале, нажмите кнопку Длина списка history. Затем в появившемся окне сообщения нажмите кнопку **ОК**, чтобы снова стали доступными кнопки перехода между страницами.