Работа с диалоговыми окнами

В рамках динамического HTML возможно создание нескольких типов окон. Выше было рассмотрено создание окон браузера, которые работают независимо от текущего окна. Такие окна называются *немодальными*.

Другие типы окон, к которым относятся различные диалоговые окна, обычно используются, когда пользователь должен сделать выбор для продолжения работы в приложении. При вызове любого из диалогов работа приложения в текущем окне останавливается, и браузер ожидает закрытия диалогового окна. Такие окна называются *модальными*.

Организация простых диалогов (методы alert, confirm и prompt)

Простейшие диалоговые окна запросов и сообщений можно создавать с помощью методов объекта `window`. Приведем полный перечень методов, отвечающих за вывод простых диалогов в текущее окно:

- `alert()` – отображает сообщение в простом диалоговом окне (рис.4, а), имеющем одну кнопку **ОК**. Диалоги `alert` не прерывают выполнение программы, однако `alert` – это именно тот тип назойливых сообщений, от которых пользователь обычно рад избавиться;
- `confirm(msg)` – отображает в диалоговом окне (рис.4, б) вопрос `msg`, на который нужно ответить «да-нет» (**ОК** или **Отмена**). Функция `confirm()` возвращает значение `true`, если нажата кнопка **ОК**, либо значение `false` при нажатии кнопки **Отмена**. Таким образом можно организовать ветвление в программе;
- `prompr()` – отображает диалоговое окно с текстовым полем, в которое пользователь должен ввести строку (рис.4, в). В диалоге имеются также кнопка **ОК** (возвращается значение текстовой величины) и кнопка **Отмена** (возвращается значение `null`).

Методы `confirm` и `prompt` не возвращают какого-либо значения, пока пользователь не закрыл диалоговое окно. Только после закрытия окна возможно продолжение работы.

Приведем в качестве примера сценарий, в котором использованы три рассмотренных метода вызова диалоговых окон:

```
<html>
  <head>
    <title>Диалоговые окна</title>
    <script language="javascript">
      alert('Клуб "Парус"');
      var msg="Являетесь ли Вы членом клуба?";
      if (confirm(msg)) {
      n = prompt("Ваше имя?","");
      document.write("Добро пожаловать, "+n+"!");
      }
      else
      alert("Вам необходимо зарегистрироваться");
    </script>
```

```
  </head>
  <body>
  <!--HTML-код страницы-->
  </body>
</html>
```

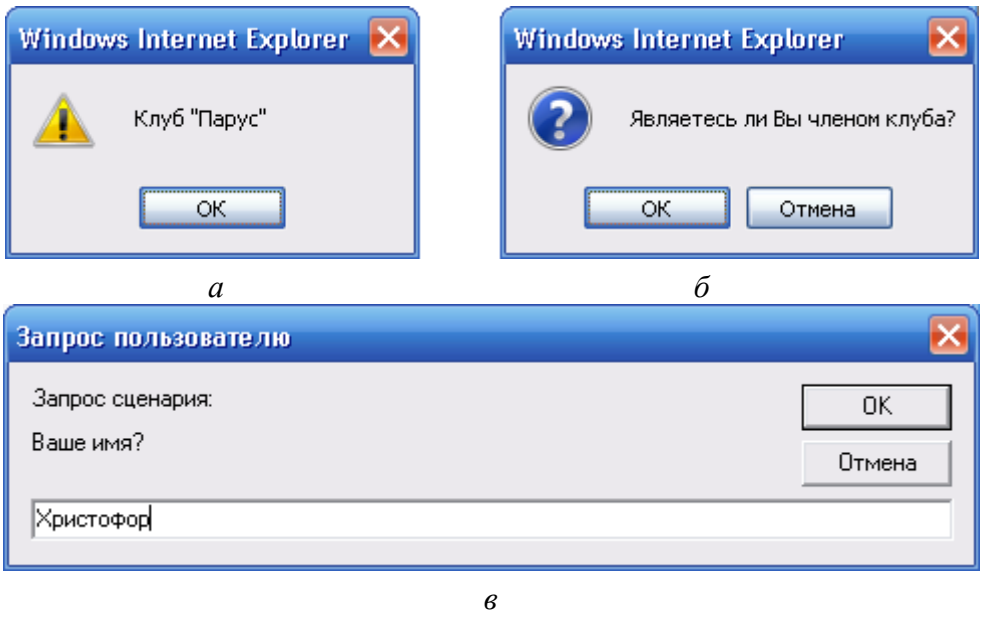

Рис. 4. Диалоговые окна – методы объекта `window`: *а –* `alert();` *б –* `confirm(msg);` *в –* `prompt()`

Имя «Христофор», введенное в последнем диалоге (рис. 4,в), будет использовано для вывода в текущем окне строки приветствия «Добро пожаловать, Христофор!». Следует обратить внимание, что в данном примере объект `window` в записи методов `alert`, `confirm` и `prompt` не указывается, поскольку эти методы относятся к текущему окну браузера.

Диалоговые окна `alert` можно использовать для отладки составленного кода. Для этого соответствующие инструкции нужно расставить в различных местах сценария для вывода промежуточных значений переменных, свойств объектов и т.д.

Создание индивидуального диалогового окна

Еще одним типом диалога, поддерживаемого в DHTML, является индивидуальное диалоговое окно. Такое окно, в отличие от описанных выше диалогов (рис.4), представляет собой Web-страницу в формате HTML с заданными размерами и своим URL.

Определение окна (метод showModalDialog)

Индивидуальное диалоговое окно создается с помощью метода `showModalDialog()`. Это окно является модальным и требует для продолжения работы ответа от пользователя. В индивидуальный диалог загружается файл HTML, однако диалог не является копией окна браузера. Это всего лишь средство просмотра. Поэтому в диалоговом окне нельзя, например, выделить текст или перемещаться по ссылкам, как в окне браузера.

Вызов метода `showModalDialog()` в общем случае выглядит как

```
showModalDialog("URL","varToDialog","features");
```

Первый аргумент "URL" – это указатель файла, в котором содержится код диалогового окна. Второй аргумент "varToDialog" отличается от соответствующего аргумента метода open: он определяет переменную (может быть массив), которая передается в диалоговое окно. Третий аргумент "features" – список параметров диалогового окна. Однако этот список оформляется несколько иначе, чем список параметров обычного окна браузера. Во-первых, параметры отделяются друг от друга точками с запятой. Во-вторых, размер и положение диалогового окна определяется следующими свойствами:

В аргументе "URL" указывается адрес файла, содержащего код диалога. Сокращение "varToDialog" обозначает список передаваемых аргументов, а "features" – параметры окна. В качестве параметров индивидуального диалогового окна обычно используюся следующие атрибуты:

- dialogWidth, dialogHeight – задают исходные ширину и высоту диалогового окна в относительных единицах em, например, dialogWidth:25em;
- dialogLeft; dialogTop – исходное положение левой и верхней границ окна;
- font (шрифт по умолчанию для диалогового окна), font-family (гарнитура шрифта), font-size (размер шрифта), font-style (стиль вывода символов). Все эти свойства могут принимать те же значения, что и свойства CSS;
- border – задает толщину границы диалогового окна путем присвоения значений thick (толстая) или thin (тонкая);
- center – выравнивает диалоговое окно по центру экрана, это свойство может иметь значения yes или no (или соответственно 0 или 1);
- maximize и minimize – определяет отображение кнопок разворачивания и сворачивания окна соответственно; этим свойствам могут присваиваться значения yes или no (или соответственно 0 или 1);
- help – определяет наличие кнопки справа в строке заголовка (значение параметра yes или 1) или отсутствие этой кнопки (значение no или 0).

Вызов метода showModalDialog, в котором указываются значения параметров, может выглядеть следующим образом:

```
window.showModalDialog(“datalog.html”,”varToDialog”,
“dialodWidth:30em”; dialogHeight:20em; center:1; maximize;
border:thick”);
```

При отсутствии параметров в списке features будут назначены параметры окна по умолчанию. Если размер созданного окна недостаточен для размещения в нем содержимого HTML-документа (файл, указанный в ″URL″), то в окне автоматически появится полоса прокрутки.

В модальных окнах, создаваемых методами showModalDialog и open, по умолчанию отображаются только строка заголовка, строка состояния, кнопки **Закрыть** и **Справка.**

Вывод информации в диалог

Рассмотрим пример создания диалогового окна, выводящего на экран некоторую информацию в виде многострочного текстового поля (рис.5).

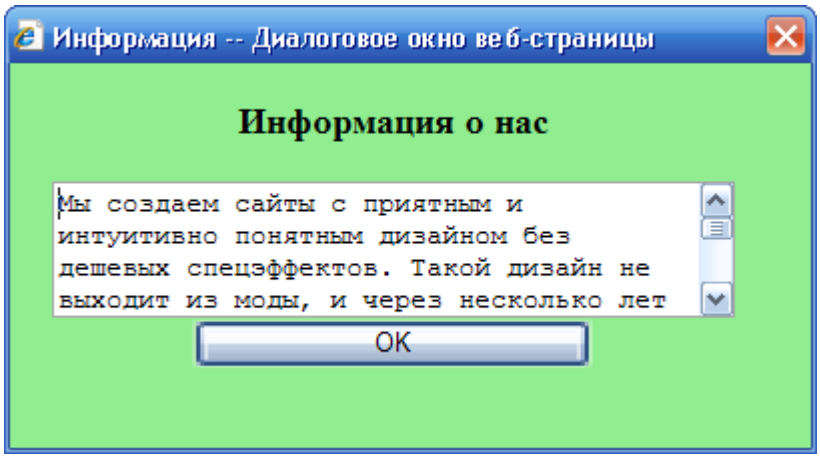

Рис.5. Пример диалога, созданного методом `showModalDialog`

Для вызова диалога введем соответствующий элемент в текущее окно браузера, например, кнопку «О нас». Если нажать эту кнопку, появится создаваемое диалоговое окно с информацией, например, о фирме. Приведем фрагмент кода этой страницы:

```
<html>
  <head>
    <title>Создание диалога</title>
    <script language="javascript">
      function ref() {
      window.showModalDialog("about.html","DialExamp",
"dialogWidth:25em; dialogHeight:12em; status=0;  ");
      }
    </script>
  </head>
  <body>
    <!--HTML-код страницы-->
    <INPUT type="button" value="О нас" onclick="ref();">
  </body>
</html>
```

HTML-код этого диалогового окна размещается в отдельном файле `about.html`, который указан в первом аргументе метода `showModalDialog`:

```
<html>
  <head>
    <title>Информация</title>
  </head>
  <body style="background:lightgreen">
    <center>
    <BR>
    <h3>Информация о нас</h3>
    <textarea rows=4 cols=40 name=comments>
```

```
        Мы создаем сайты с приятным и интуитивно понятным
        дизайном без дешевых спецэффектов. Такой дизайн не
        выходит из моды, и через несколько лет будет так же
        понятен и приятен вам и пользователям вашего сайта.
    </textarea>
    <INPUT type="button" style="width:15em" value="OK"
onclick="window.close()">
    </center>
  </body>
</html>
```

Для продолжения работы закройте диалог, щелкнув по кнопке **OK**, и вы вернетесь в исходное окно браузера. Обратите внимание, что метод `window.close` закрывает диалоговое окно, а не окно браузера. Если же документ `about.html` был открыт не в отдельном окне, а в текущем окне браузера (командой **Файл** – **Открыть**), то при закрытии окна щелчком по кнопке **OK** появится предупреждение о попытке закрытия файла самой Web-страницы.

Обмен данными между диалоговым окном и документом

Для вывода информации в HTML-документ можно использовать диалоговые окна. Если требуется вводить небольшой объем данных, то можно воспользоваться простым немодальным окном, вызываемым с помощью метода `prompt`. Однако для передачи множества значений в исходный документ более подходит индивидуальное диалоговое окно. Это окно позволяет организовать обмен данными между пользователем и приложением. В диалоге могут вводиться значения в поля ввода, выбираться опции из списка, устанавливаться флажки и переключатели и т.д.

При операциях с диалоговыми окнами следует помнить два основных свойства объектной модели окна: `window.returnValue` (возврат значений из диалогового окна в приложение) и `window.dialogArguments` (передача данных из приложения в диалоговое окно).

Обработка событий окна

Напомним, что к событиям окна относятся события, связанные с загрузкой и выгрузкой документа. Рассмотрим обработку этих событий в динамическом HTML.

Событие onLoad для различных элементов

После анализа документа и его разметки браузер переходит к загрузке внедренных элементов. Проследить последовательность загрузки различных элементов и всего документа можно с помощью события `onLoad`.

Для определения очередности наступления событий загрузки рассмотрим пример документа, содержащего изображение. Установим обработчик события `onload` в тегах `<IMG>` и `<BODY>`:

```
<html>
  <head>
```

```
    <title>Последовательность событий onload</title>
  </head>
  <body onload="alert('Наступление onload для BODY');">
    <!--HTML-код страницы-->
    <IMG src="rose.jpg" onload="alert('Наступление onload для
IMG');">
    <!--HTML-код страницы-->
  </body>
</html>
```

Обработчики событий `onload` будут захватываться по мере загрузки элементов документа. Сначала всплывет сообщение «Наступление `onload` для `IMG`», которое сигнализирует о том, что загружено изображение. После этого появится сообщение «Наступление `onload` для `BODY`», сигнализирующее о загрузке всего документа. Для объекта `window` не существует элемента – прямого аналога в коде HTML (как, например, для изображения). Однако по поведению элемента `BODY` можно сказать, что событие `onload` для объекта `window` возникает, когда документ целиком проанализирован и все его элементы (включая и все рисунки) загружены.

Проверка загрузки документа

Чтобы избежать обращения к кому-либо элементу HTML до окончания загрузки документа, можно поступить следующим образом. Воспользуемся событием `onload` объекта `document`, которое возникает когда документ целиком проанализирован и выполнена его начальная загрузка. В тег `<BODY>` введем обработчик события `onload()`, для которого определим новое свойство объекта `window`. Это свойство, назовем его `isLoaded`, будет принимать значение true лишь в случае загрузки документа:

```
<BODY onload=”window.isLoaded=true;”>
```

Теперь при запуске других обработчиков событий можно проверять значение `window.isloaded` с тем, чтобы быть уверенными, что документ полностью загружен и все содержащиеся в нем объекты определены. Приведем пример такой проверки для элемента `button`, построив документ по следующей схеме:

```
<html>
  <head>
    <title>Запуск обработчика с проверкой</title>
    <script>
      window.isLoaded=false;
      //Любая функция обработчика события
      funcrion inform(){
      alert("Документ загружен");
      }
    </script>
  </head>
  <body onload="window.isLoaded=true;">
    <!--HTML-код страницы-->
    <INPUT type="button" name=rest value="Сведения о
загрузке" onclick="if(window.isLoaded)inform();">
    <!--HTML-код страницы-->
  </body>
```

```
</html>
```

Нажатие кнопки «Сведения о загрузке» будет обрабатываться лишь в случае значения `window.isLoaded=true`, то есть для загруженного документа. В принципе подобные проверки значения флага `window.isLoaded` можно установить во всех обработчиках событий, которые должны исполняться при условии полной загрузки документа.