Объект FORM

Введя в документ форму, помимо стандартной процедуры отправки формы на сервер, можно выполнить проверку введенной информации, очистить форму, изменить набор отображаемых элементов управления, их оформление и т.д. К каждому элементу формы обеспечивается доступ с помощью объектной модели динамического HTML.

Семейства форм

В динамическом HTML элементам форм отвечают объекты, которые формируют ряд важных семейств.

Семейство forms

Как известно, в документе может быть несколько форм, и ни одна из форм не может включать другую. То есть, все формы принадлежат одному иерархическому уровню, который определяется семейством `forms`. Это семейство является массивом упорядоченных, проиндексированных элементов, каждый из которых представляет определенный тег `<form>` в документе. Элементы массива `forms[j]` различаются индексом `j`, причем первым элементом является `forms[0]`.

Семейство elements

Массив `elements` является основным свойством объекта `form` и обеспечивает доступ к элементам управления. Данный массив представляет все элементы формы, пронумерованные от 0 до `n-1`, где `n` – длина массива. Длина массива определяется стандартным образом с помощью свойства `length`:

```
document.forms[j].elements.length
```

где `j` – индекс формы в массиве `forms` (для именованной формы можно использовать имя формы).

Рассмотрим простой сценарий:

```
<script language="javascript">
  document.write("<h2>Список имен элементов формы</h2>");
  //Определение количества элементов в форме
  ex=document.forms[0].elements.length;
  Вывод имен элементов управления на экран
  for(i = 0; i<ex; i++){
    document.write(document.forms[0].elements[i].name  +  ",
");
}
</script>
```

В результате действия данного сценария на экран будет выведен список имен всех элементов формы.

Обращение к форме

Обратиться к форме можно по ее индексу в массиве `forms`:

```
document.forms[0].
```

Если форма имеет имя (задан атрибут имени, например, `name="formGuest"`), то вместо индекса можно указать имя:

```
document.forms[formGuest].
```

Кроме того, к форме можно обратиться непосредственно по имени:

```
document.formGuest.
```

Тем не менее, лучше ссылаться на форму через семейство `forms`, поскольку сценарии при этом лучше читаются и более понятны:

```
document.forms.formGuest.
```

Свойства `document.forms.formGuest.name` и `document.formGuest.name` будут возвращать значения `formGuest`.

Свойства и методы объекта form

Стандартные свойства DHTML
Объект `form` поддерживает набор стандартных свойств динамического HTML:

- `className` – это свойство позволяет определить, к какому классу, определенному в списке CSS, относится данный объект формы. Так, для элемента `<form id="f1" class="require">` свойство `document.forms.f1.className` будет возвращать значение `require`.
- `document` – возвращает ссылку на объект документа и, следовательно, обеспечивает доступ к любому свойству документа. Например, из формы можно сослаться на URL документа следующим образом: `this.document.location`;
- `id` – это свойство возвращает идентификатор формы;
- `innerHTML` и `innerTEXT` – возвращают соответственно содержимое HTML-элемента и содержимое элемента, включая только текст без внутренних тегов;
- `isTextEdit` – это свойство принимает два булевских значения, позволяющих определить, предназначен ли данный элемент формы для ввода или редактирования текста. Так, свойство `this.isTextEdit` возвращает `true` только для элементов text, `texrarea`, `button` и `reset`. Для остальных элементов оно возвращает значение `false`;
- `lang` – свойство для определения языка, который используется в элементе формы;
- `language` – определяет язык сценария;
- `offsetHeight`, `offsetWidth` – возвращают высоту и ширину воспроизводимого элемента или самой формы;
- `offsetLeft`, `offsetTop` – возвращают координаты левого верхнего угла элемента;
- `offsetParent` – возвращают ссылку на родительский объект;
- `outerText`, `outerHTML`;
- `sourceIndex` – возвращает индекс элемента `<form>` в массиве семейства `all`;

- `style` – возвращает ссылку на объект стиля, который определен в теге `<form>` или в тегах элементов формы;
- `title` – задает вывод и содержание подсказки, которая всплывает на экране при прохождении указателя мыши над элементом формы.

Все перечисленные выше свойства применимы как к формам, так и к элементам форм.

Установка фокуса и выделение элемента (методы focus() и select())

Установить фокус на нужном элементе можно с помощью обработчика события `onload`, помещенного в тег `<body>`:

```
<body onload = "document.f1.userName.focus()">
```

где `f1` – имя формы, `userName` – имя элемента формы, `focus()` – метод установки фокуса.
Если требуется установить фокус на элементе ввода, причем его содержимое должно быть выделено, нужно воспользоваться методом `select()`:

```
<body onload = "document.f1.userName.focus();
document.f1.userName.select()">
```

Проверка ввода данных в форму

Проверка вводимых символов

Обработку вводимого символа можно выполнять с помощью события `onkeypress`, которое возникает при нажатии и отпускании любой клавиши. Свойство `keyCode` этого события возвращает код клавиши в кодировке `Unicode`. Арабским цифрам соответствует интервал от 48 до 57, буквам русского алфавита – интервал 1040 – 1103, прописным буквам латинского алфавита – интервал 65 – 90, строчным – 97 – 122. Для блокировки ввода неправильного символа в поле используется свойство `returnValue`. Если данному свойству присвоить значение `false`, введенный символ обрабатываться не будет.

Запишем для цифрового поля обработчик события `onkeypress`, который будет блокировать обработку всех символов, кроме цифр:

```
onkeypress="if((event.keyCode<48) ||
(event.keyCode>57))event.returnValue=false;"
```

Данный обработчик блокирует все символы, которые не являются арабскими цифрами.
Для буквенного поля обработчик задается в виде:

```
onkeypress="if((event.keyCode<1040) ||
(event.keyCode>1103))event.returnValue=false;"
```

Здесь обработчик блокирует все символы, которые не являются буквами русского алфавита.

Данный способ проверки данных полезен, когда необходимо проверять каждый символ. Более общую проверку – проверку данных в целом – можно выполнить при выходе из элемента управления.

Проверка при выходе из элемента управления

Такую проверку можно связать с событием `onchange`, которое наступает всякий раз при выходе пользователя из элемента управления (переводе фокуса на другой элемент). Если введенные данные окажутся неверными, это можно отобразить в появляющемся диалоговом окне или в изменении внешнего вида элемента.

Предположим, на странице необходимо организовать проверку, являются ли все введенные символы цифрами. Код документа будет иметь следующий вид:

```
<html>
  <head>
    <script language="javascript">
      function chkStr(){
      //Определение объекта - элемента управления
      var x=document.all.age;
      var num="0123456789";
      event.returnValue=true;
      //Проверка введенной строки (состоит ли она только из
цифр) elem.value - это введенная строка, а elem.value.length
- длина строки
      for (var i=0; i < x.value.length; i++)
      //Если символ не является числом, возвращается
значение false
      if (-1== num.indexOf(x.value.charAt(i)))
      event.returnValue=false;
      //Сброс значения поля перед повторным вводом
      if (!event.returnValue)
      x.value="";
      alert("Повторите ввод." + '\n'+"Возраст должен
набираться числом")
      }
    </script>
  </head>
  <body>
    <input id="age" type=text size=3 title="Введите возраст"
onchange="chkStr()">
  </body>
</html>
```

Здесь использованы следующие функции:

- `string.charAt(i)` – возвращает символ, расположенный в `i`-ой позиции строки string. Позиции символов нумеруются от нуля до `string.length-1`.
- `string.indexOf(x)` – возвращает номер позиции впервые встреченного символа в строке string. Если символ не найден, то возвращает значение –1.

Проверка данных при отправке формы

Приведем код страницы, содержащей форму с обязательными для заполнения текстовыми полями.

```
<html>
<head>
 <title>Проверка полноты данных</title>
 <style type="text/css">
  .att{background:red}
  div {font-family:"ms sans serif, sans-serif"}
 </style>
 <script language="javascript">
 //Функция проверки, не является ли элемент пустым
  function isEmpty(inText){
   for (var i=0; i < inText.length; i++)
    if (""!=inText.charAt(i))
     return false;
   return true;
  }
  //Функция просмотра всех полей
  function fullData(x) {
   for (var i=0; i<x.elements.length; i++)
    if ("att"==x.elements(i).className){
    alert("Необходимо заполнить все выделенные цветом
поля");
    return false;
    }
  }
   //Функция определения класса элемента
   function detClass(x) {
    x.className=isEmpty(x.value)?"att":"";
   }
   //Функция проверки, не является ли введенный символ
пробелом (код 32)
   function checkGap(x) {
   if (32!=event.keyCode)
    x.className="";
   }
   </script>
</head>
<body>
 <form name="userform" onsubmit="return fullData(this)">
  <div>Имя пользователя:</div>
  <input type=text class="att" onkeypress="checkGap(this)"
onchange="detClass(this)"><br>
  <div>Электронный адрес:</div>
  <input type=text class="att" onkeypress="checkGap(this)"
onchange="detClass(this)"><br>
  <div>Домашний адрес:</div>
  <input type=text size=45><br>
  <div>Возраст:</div>
  <input type=text size=3><br>
```

```
  <input type=submit value="Отправить">
 </form>
</body>
</html>
```

Обязательные поля формы окрашены в красный цвет. Фоновый цвет этих полей, принадлежащих классу `att`, задается стилевым свойством `background`. При вводе символа в обязательное поле, выполняется проверка, не является ли вводимый символ пробелом. Обработчик события `onkeypress` вызывает функцию `checkGap()`, которая «сбрасывает» класс `att` при вводе первого символа не пробела. В результате поле становится белым.

При выходе из обязательного поля ввода, выполняется обработка события `onchange` с помощью функции `detClass()`. Эта функция определяет принадлежность элемента классу `att`. Класс элемента устанавливается с помощью булевской функции пустого поля `isEmpty()`, которая возвращает значение `true`, если в поле нет символов, и `false` – в противном случае.

Для проверки формы перед ее отправкой сеть, в тег `<form>` помещен обработчик события `onsubmit`, которое наступает при нажатии кнопки **Отправить**. Для обработки данного события используется функция `fullData()`, которая просматривает все элементы формы и проверяет, нет ли среди них принадлежащих классу `att`. Если такие элементы имеются, то на экран выводится сообщение с предложением ввести данные в незаполненное поле. Функции `fullData()` присваивается значение `false`, и данные формы в сеть отправлены не будут.