Часть 2: **Документ HTML**

Иерархия объектов в JavaScript
В языке JavaScript все элементы на web-странице выстраиваются в иерархическую структуру. Каждый элемент предстает в виде объекта. И каждый такой объект может иметь определенные свойства и методы. В свою очередь, язык JavaScript позволит Вам легко управлять объектами web-страницы, хотя для этого очень важно понимать иерархию объектов, на которые опирается разметка HTML. Как это все действует, Вы сможете быстро понять на следующем примере. Рассмотрим простую HTML-страницу:

```
<html>
<head>
</head>
<body bgcolor=#ffffff>
<center>
<img src="home.gif" name="pic1" width=200 height=100>
</center>
<p>
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
e-Mail:
<input type="text" name="email" value=""><br><br>
<input type="button" value="Push me" name="myButton" onClick="alert('Yo')">
</form>
<p>
<center>
<img src="ruler.gif" name="pic4" width=300 height=15>
<p>
<a href="http://rummelplatz.uni-mannheim.de/~skoch/">My homepage</a>
</center>
</body>
</html>
```

Итак, мы имеем два рисунка, одну ссылку и некую форму с двумя полями для ввода текста и одной кнопкой. С точки зрения языка JavaScript окно браузера - это некий объект window. Этот объект также содержит в свою очередь некоторые элементы оформления, такие как строка состояния. Внутри окна мы можем разместить документ HTML (или файл какого-либо другого типа - однако пока мы все же ограничимся файлами HTML). Такая страница является ни чем иным, как объектом document. Это означает, что объект document представляет в языке JavaScript загруженный на настоящий момент документ HTML. Объект document является очень важным объектом в языке JavaScript и Вы будете пользоваться им многократно. К свойствам объекта document относятся, например, цвет фона для web-страницы. Однако для нас гораздо важнее то, что все без исключения объекты HTML являются свойствами объекта document. Примерами объекта HTML являются, к примеру, ссылка или заполняемая форма. На следующем рисунке иллюстрируется иерархия объектов, создаемая HTML-страницей из нашего примера:

Разумеется, мы должны иметь возможность получать информацию о различных объектах в этой иерархии и управлять ею. Для этого мы должны знать, как в языке JavaScript организован доступ к различным объектам. Как видно, каждый объект иерархической структуры имеет свое имя. Следовательно, если Вы хотите узнать, как можно обратиться к первому рисунку на нашей HTML-странице, то обязаны сориентироваться в иерархии объектов. И начать нужно с самой вершины. Первый объект такой структуры называется document. Первый рисунок на странице представлен как объект images[0]. Это означает, что отныне мы можем получать доступ к этому объекту, записав в JavaScript document.images[0]. Если же, например, Вы хотите знать, какой текст ввел читатель в первый элемент формы, то сперва должны

выяснить, как получить доступ к этому объекту. И снова начинаем мы с вершины нашей иерархии объектов. Затем прослеживаем путь к объекту с именем elements[0] и последовательно записываем названия всех объектов, которые минуем. В итоге выясняется, что доступ к первому полю для ввода текста можно получить, записав:

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

А теперь как узнать текст, введенный читателем? Чтобы выяснять, которое из свойств и методов объекта позволят получить доступ к этой информации, необходимо обратиться к какому-либо справочнику по JavaScript (например, это может быть документация, предоставляемая фирмой Netscape, либо моя книга). Там Вы найдете, что элемент, соответствующий полю для ввода текста, имеет свойство value, которое как раз и соответствует введенному тексту. Итак, теперь мы имеем все необходимое, чтобы прочитать искомое значение. Для этого нужно написать на языке JavaScript строку:

```
name= document.forms[0].elements[0].value;
```

Полученная строка заносится в переменную name. Следовательно, теперь мы можем работать с этой переменной, как нам необходимо. Например, мы можем создать выпадающее окно, воспользовавшись командой alert("Hi " + name). В результате, если читатель введет в это поле слово 'Stefan', то по команде alert("Hi " + name) будет открыто выпадающее окно с приветствием ' Hi Stefan '.

Если Вы имеете дело с большими страницами, то процедура адресации к различным объектам по номеру может стать весьма запутынной. Например, придется решать, как следует обратиться к объекту document.forms[3].elements[17] document.forms[2].elements[18]? Во избежание подобной проблемы, Вы можете сами присваивать различным объектам уникальные имена. Как это делается, Вы можете увидеть опять же в нашем примере:

```
<form name="myForm">
Name:
<input type="text" name="name" value=""><br>
...
```

Эта запись означает, что объект forms[0] получает теперь еще и второе имя - myForm. Точно так же вместо elements[0] Вы можете писать name (последнее было указано в атрибуте name тэга <input>). Таким образом, вместо

```
name= document.forms[0].elements[0].value;
```

Вы можете записать

```
name= document.myForm.name.value;
```

Это значительно упрощает программирование на JavaScript, особенно в случае с большими web-страницами, содержащими множество объектов. (Обратите внимание, что при написании имен Вы должны еще следить и за положением регистра - то есть Вы не имеете права написать myform вместо myForm). В JavaScript многие свойства объектов доступны не только для чтения. Вы также имеете возможность записывать в них новые значения. Например, посредством JavaScript Вы можете записать в упоминавшееся поле новую строку.

Пример кода на JavaScript, иллюстрирующего такую возможность - интересующий нас фрагмент записан как свойство onClick второго тэга <input>:

```
<form name="myForm">
<input type="text" name="input" value="bla bla bla">
```

```
<input type="button" value="Write"
onClick="document.myForm.input.value= 'Yo!'; ">
```

Сейчас я не имею возможности описывать каждую деталь данного примера. Намного лучше будет, если Вы попробуете сами понять иерархию объектов в JavaScript, обратившись к справочнику по JavaScript. В заключение я написал небольшой пример. В нем Вы увидите, как используются различные объекты. Попытайтесь разобрать его, обращаясь за помощью к документации, предоставляемой фирмой Netscape, или - еще лучше - к моей книге по JavaScript..: -)

Исходный код скрипта:

```
<html>
<head>
<title>Objects</title>
<script language="JavaScript">
<!-- hide
function first() {
// создает выпадающее окно, где размещается
// текст, введенный в поле формы
alert("The value of the textelement is: " +
document.myForm.myText.value);
}
function second() {
// данна\я функци\я провер\яет состо\яние переключателей
var myString= "The checkbox is ";
// переключатель включен, или нет?
if (document.myForm.myCheckbox.checked) myString+= "checked"
else myString+= "not checked";
// вывод строки на экран
alert(myString);
}
// -->
</script>
</head>
<body bgcolor=lightblue>
<form name="myForm">
<input type="text" name="myText" value="bla bla bla">
<input type="button" name="button1" value="Button 1"
onClick="first()">
<br>
<input type="checkbox" name="myCheckbox" CHECKED>
<input type="button" name="button2" value="Button 2"
onClick="second()">
</form>
<p><br><br>
<script language="JavaScript">
<!-- hide
document.write("The background color is: ");
document.write(document.bgColor + "<br>");
document.write("The text on the second button is: ");
document.write(document.myForm.button2.value);
// -->
</script>
</body>
</html>
```

Объект location

Кроме объектов window и document в JavaScript имеется еще один важный объект - location. В этом объекте представлен адрес загруженного HTML-документа. Например, если Вы загрузили страницу http://www.xyz.com/page.html, то значение location.href как раз и будет соответствовать этому адресу. Впрочем, для нас гораздо более важно, что Вы имеете возможность записывать в location.href свои новые значения. Например, в данном примере кнопка загружает в текущее окно новую страницу:

```
<form>
<input type=button value="Yahoo"
onClick="location.href='http://www.yahoo.com'; ">
</form>
```