

ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

Часть 4: Окна и динамически создаваемые документы

Создание окон

Открытие новых окон в браузере - грандиозная возможность языка JavaScript. Вы можете либо загружать в новое окно новые документы (например, те же документы HTML), либо (динамически) создавать новые материалы. Посмотрим сначала, как можно открыть новое окно, потом как загрузить в это окно HTML-страницу и, наконец, как его закрыть. Приводимый далее скрипт открывает новое окно браузера и загружает в него некую web-страничку:

```
<html>
<head>
<script language="JavaScript">
<!-- hide

function openWin() {
    myWin= open("bla.htm");
}

// -->
</script>
</head>
<body>

<form>
<input type="button" value="Открыть новое окно" onClick="openWin()">
</form>

</body>
</html>
```

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

В представленном примере в новое окно с помощью метода *open()* записывается страница *bla.htm*.

Заметим, что Вы имеете возможность управлять самим процессом создания окна. Например, Вы можете указать, должно ли новое окно иметь строку статуса, панель инструментов или меню. Кроме того Вы можете задать размер окна. Например, в

следующем скрипте открывается новое окно размером 400x300 пикселей. Оно не имеет ни строки статуса, ни панели инструментов, ни меню.

```
<html>
<head>
<script language="JavaScript">
<!-- hide

function openWin2() {
  myWin= open("bla.htm", "displayWindow",
    "width=400,height=300,status=no,toolbar=no,menubar=no");
}

// -->
</script>
</head>
<body>

<form>
<input type="button" value="Открыть новое окно" onClick="openWin2()">
</form>

</body>
</html>
```

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Как видите, свойства окна мы формулируем в строке "width=400,height=300,status=no,toolbar=no,menubar=no". Обратите внимание также и на то, что Вам не следует помещать в этой строке символы пробела!

Список свойств окна, которыми Вы можете управлять:

directories	yes no
height	<i>количество пикселей</i>
location	yes no
menubar	yes no
resizable	yes no
scrollbars	yes no
status	yes no
toolbar	yes no
width	<i>количество пикселей</i>

В версии 1.2 языка JavaScript были добавлены некоторые новые свойства (то есть в Netscape Navigator 4.0). Вам не следует пользоваться этими свойствами, готовя материалы для Netscape 2.x, 3.x или Microsoft Internet Explorer 3.x, поскольку эти браузеры не понимают языка 1.2 JavaScript. Новые свойства окон:

alwaysLowered	yes no
alwaysRaised	yes no
dependent	yes no

hotkeys	yes no
innerWidth	количество пикселей (заменяет width)
innerHeight	количество пикселей (заменяет height)
outerWidth	количество пикселей
outerHeight	количество пикселей
screenX	количество пикселей
screenY	количество пикселей
titlebar	yes no
z-lock	yes no

Вы можете найти толкование этих свойств в описании языка JavaScript 1.2. В дальнейшем я для некоторых из них дам разъяснение и примеры использования. Например, теперь с помощью этих свойств Вы можете определить, в каком месте экрана должно находиться вновь открываемое окно. Работая со старой версией языка JavaScript, Вы не смогли бы этого сделать.

Имя окна

Как видите, открывая окна, мы должны использовать три аргумента:

```
myWin= open("bla.htm", "displayWindow",
"width=400,height=300,status=no,toolbar=no,menubar=no");
```

А для чего нужен второй аргумент? Это имя окна. Ранее мы видели, как оно использовалось в параметре target. Так, если Вы знаете имя окна, то можете загрузить туда новую страницу с помощью записи

```
<a href="bla.html" target="displayWindow">
```

При этом Вам необходимо указать имя соответствующего окна (если же такого окна не существует, то с этим именем будет создано новое). Обратите внимание, что *myWin* - это вовсе не имя окна. Но только с помощью этой переменной Вы можете получить доступ к окну. И поскольку это обычная переменная, то область ее действия - лишь тот скрипт, в котором она определена. А между тем, имя окна (в данном случае это *displayWindow*) - уникальный идентификатор, которым можно пользоваться с любого из окон браузера.

Создание окон

Вы можете также закрывать окна с помощью языка JavaScript. Чтобы сделать это, Вам понадобится метод close(). Давайте, как было показано ранее, откроем новое окно. И загрузим туда очередную страницу:

```
<html>
<script language="JavaScript">
<!-- hide
```

```
function closeIt() {
    close();
}

// -->
</script>

<center>
<form>
<input type=button value="Close it" onClick="closeIt()">
</form>
</center>

</html>
```

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Если теперь в новом окне Вы нажмете кнопку, то оно будет закрыто. *open()* и *close()* - это методы объекта *window*. Мы должны помнить, что следует писать не просто *open()* и *close()*, а *window.open()* и *window.close()*. Однако в нашем случае объект *window* можно опустить - Вам нет необходимости писать префикс *window*, если Вы хотите всего лишь вызвать один из методов этого объекта (и такое возможно только для этого объекта).

Динамическое создание документов

Теперь мы готовы к рассмотрению такой замечательной возможности JavaScript, как динамическое создание документов. То есть Вы можете разрешить Вашему скрипту на языке JavaScript самому создавать новые HTML-страницы. Более того, Вы можете таким же образом создавать и другие документы Web, такие как VRML-сцены и т.д. Для удобства Вы можете размещать эти документы в отдельном окне или фрейме. Для начала мы создадим простой HTML-документ, который покажем в новом окне. Рассмотрим следующий скрипт.

```
<html>
<head>
<script language="JavaScript">
<!-- hide

function openWin3() {
    myWin= open("", "displayWindow",
        "width=500,height=400,status=yes,toolbar=yes,menubar=yes");

    // открыть объект document для последующей печати
    myWin.document.open();

    // генерировать новый документ
    myWin.document.write("<html><head><title>On-the-fly");
    myWin.document.write("</title></head><body>");
    myWin.document.write("<center><font size=+3>");
```

```

myWin.document.write("Данный документ HTML был создан ");
myWin.document.write("с помощью JavaScript!");
myWin.document.write("</font></center>");
myWin.document.write("</body></html>");

// закрыть документ - (но не окно!)
myWin.document.close();
}

// -->
</script>
</head>
<body>

<form>
<input type=button value="On-the-fly" onClick="openWin3()">
</form>

</body>
</html>

```

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Давайте рассмотрим функцию `winOpen3 ()`. Очевидно, мы сначала открываем новое окно браузера. Поскольку первый аргумент функции `open()` - пустая строка (`""`), то это значит, что мы не желаем в данном случае указывать конкретный адрес URL. Браузер должен только не обработать имеющийся документ - JavaScript обязан создать дополнительно новый документ.

В скрипте мы определяем переменную `myWin`. И с ее помощью можем получать доступ к новому окну. Обратите пожалуйста внимание, что в данном случае мы не можем воспользоваться для этой цели именем окна (`displayWindow`). После того, как мы открыли окно, наступает очередь открыть для записи объект `document`. Делается это с помощью команды:

```

// открыть объект document для последующей печати
myWin.document.open();

```

Здесь мы обращаемся к `open()` - методу объекта `document`. Однако это совсем не то же самое, что метод `open()` объекта `window`! Эта команда не открывает нового окна - она лишь готовит `document` к предстоящей печати. Кроме того, мы должны поставить перед `document.open()` приставку `myWin`, чтобы получить возможность писать в новом окне.

В последующих строках скрипта с помощью вызова `document.write()` формируется текст нового документа:

```

// генерировать новый документ
myWin.document.write("<html><head><title>On-the-fly");
myWin.document.write("</title></head><body>");
myWin.document.write("<center><font size=+3>");
myWin.document.write("This HTML-document has been created ");
myWin.document.write("with the help of JavaScript!");

```

```
myWin.document.write("</font></center>");  
myWin.document.write("</body></html>");
```

Как видно, здесь мы записываем в документ обычные тэги языка HTML. То есть мы фактически генерируем разметку HTML! При этом Вы можете использовать абсолютно любые тэги HTML.

По завершении этого мы обязаны вновь закрыть документ. Это делается следующей командой:

```
// закрыть документ - (но не окно!)  
myWin.document.close();
```

Как я уже говорил, Вы можете не только динамически создавать документы, но и по своему выбору размещать их в том или ином фрейме. Например, если Вы получили два фрейма с именами *frame1* и *frame2*, а теперь во *frame2* хотите сгенерировать новый документ, то для этого в *frame1* Вам достаточно будет написать следующее:

```
parent.frame2.document.open();  
  
parent.frame2.document.write("Here goes your HTML-code");  
  
parent.frame2.document.close();
```

Динамическое создание VRML-сцен

Чтобы продемонстрировать гибкость языка JavaScript, давайте теперь попытаемся динамически создать сцену на языке VRML. Напомним, что аббревиатура VRML расшифровывается как язык моделирования виртуальной деальности. То есть это язык для создания трехмерных сцен. Можно, например, взять очки виртуальной реальности и насладиться прогулкой по таким сценам ... Возьмем самый простой пример - голубой куб. Тем не менее, чтобы рассмотреть его, понадобится программная приставка VRML к Вашему браузеру (plug-in). Предлагаемый Вашему вниманию скрипт не проверяет, а доступен ли браузеру plug-in VRML (впрочем сделать это - вовсе не проблема).

(online-версия руководства позволит Вам проверить этот скрипт немедленно)

Исходный код скрипта:

```
<html>  
<head>  
<script language="JavaScript">  
<!-- hide  
  
function vrmlScene() {  
    vrml= open("", "displayWindow",  
        "width=500,height=400,status=yes,toolbar=yes,menubar=yes");  
  
    // открыть document для последующего вывода информации  
    vrml.document.open("x-world/x-vrml");
```

```

vr= vrml.document;

// создать сцену VRML
vr.writeln("#VRML V1.0 ascii");

// Освещение
vr.write("Separator { DirectionalLight { ");
vr.write("direction 3 -1 -2.5 } ");

// Камера
vr.write("PerspectiveCamera { position -8.6 2.1 5.6 ");
vr.write("orientation -0.1352 -0.9831 -0.1233 1.1417 ");
vr.write("focalDistance 10.84 } ");

// Куб
vr.write("Separator { Material { diffuseColor 0 0 1 } ");
vr.write("Transform { translation -2.4 .2 1 rotation 0 0.5 1 .9 } ");
vr.write("Cube { } }");

// Закрывать document - (но не окно!)
vrml.document.close();
}

// -->
</script>
</head>
<body>

<form>
<input type=button value="VRML on-the-fly" onClick="vrmlScene()">
</form>

</body>
</html>

```

Как видно, текст скрипта совершенно такой же, как и в предыдущем примере. Сперва открывается новое окно. Затем мы открываем document для вывода него информации. Рассмотрим поподробнее соответствующую команду:

```

// открыть document для последующего вывода информации
vrml.document.open("x-world/x-vrml");

```

В предыдущих примерах мы не указывали в скобках ничего. Что же тогда означает новая запись "x-world/x-vrml"? На самом же деле, с помощью этой инструкции мы задаем тип MIME для документа, который хотим создать. То есть, тем самым мы сообщаем браузеру, какого типа данные будут ему сейчас переданы. Если же мы в этом месте не определили в скобках конкретный тип MIME, то по умолчанию для нового документа будет выбран тип "text/html" (а это как раз и есть тип MIME для файлов HTML).

(Есть несколько способов выяснить, что же означает тот или иной тип MIME - в самом же браузере содержится список распознаваемых MIME. Вы можете извлечь этот список из пунктов меню option или preference.)

Для создания трехмерной сцены мы должны составить инструкцию `vrml.document.write()`. Но поскольку это кажется слишком длинным, то мы просто определяем переменную `vr = vrml.document`. И затем вместо `vrml.document.write()` мы пишем просто `vr.write()`.

Закончив это, мы можем писать обычные инструкции на языке VRML. Я не собираюсь описывать здесь элементы сцен VRML. А для желающих познакомиться с ними в Интернет имеется несколько хороших источников информации. Обычный же текст на языке VRML выглядит следующим образом:

```
#VRML V1.0 ascii
```

```
Separator {
```

```
    DirectionalLight { direction 3 -1 -2.5 }
```

```
    PerspectiveCamera {  
        position -8.6 2.1 5.6  
        orientation -0.1352 -0.9831 -0.1233 1.1417  
        focalDistance 10.84  
    }
```

```
Separator {  
    Material {  
        diffuseColor 0 0 1  
    }  
    Transform {  
        translation -2.4 .2 1  
        rotation 0 0.5 1 .9  
    }  
    Cube {}  
}  
}
```

А это как раз и есть тот код, который мы выводим на экран с помощью команды `document.write()`.

Впрочем, совершенно бессмысленно динамически создать сцену, которую с тем же успехом можно загрузить и как обычный VRML-файл.

Интереснее будет, если Вы, например, сделаете форму, где пользователь будет иметь выбор из различных объектов - например, между сферой, цилиндром, конусом и т.д. - а JavaScript на основе этих данных всего лишь сгенерирует соответствующую трехмерную сцену (например, так я поступаю в своей книге о JS).