

ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

Часть 8: Объект Image

Изображения на web-странице

Рассмотрим теперь объект Image, который стал доступен, начиная с версии с 1.1 языка JavaScript (то есть с Netscape Navigator 3.0). С помощью объекта Image Вы можете вносить изменения в графические образы, присутствующие на web-странице. В частности, это позволяет нам создавать мультипликацию.

Заметим, что пользователи браузеров более старых версий (таких как Netscape Navigator 2.0 или Microsoft Internet Explorer 3.0 - т.е. использующих версию 1.0 языка JavaScript) не смогут запускать скрипты, приведенные в этой части описания. Или, в лучшем случае, на них нельзя будет получить полный эффект. Давайте сначала рассмотрим, как из JavaScript можно адресоваться к изображениям, представленным на web-странице. В рассматриваемом языке все изображения предстают в виде массива. Массив этот называется `images` и является свойством объекта `document`. Каждое изображение на web-странице получает порядковый номер: первое изображение получает номер 0, второе - номер 1 и т.д. Таким образом, к первому изображению мы можем адресоваться записав `document.images[0]`.

Каждое изображение в HTML-документе рассматривается в качестве объекта Image. Объект Image имеет определенные свойства, к которым и можно обращаться из языка JavaScript. Например, Вы можете определить, который размер имеет изображение, обратившись к его свойствам `width` и `height`. То есть по записи `document.images[0].width` Вы можете определить ширину первого изображения на web-странице (в пикселах).

К сожалению, отслеживать индекс всех изображений может оказаться затруднительным, особенно если на одной странице у Вас их довольно много. Эта проблема решается назначением изображениям своих собственных имен. Так, если Вы заводите изображение с помощью тэга

```

```

то Вы сможете обращаться к нему, написав `document.myImage` или `document.images["myImage"]`.

Загрузка новых изображений

Хотя конечно и хорошо знать, как можно получить размер изображения на web-странице, это не совсем то, чего бы мы хотели. Мы хотим осуществлять смену изображений на web-странице и для этого нам понадобится атрибут `src`. Как и в случае тэга ``, атрибут `src` содержит адрес представленного изображения. Теперь - в языке JavaScript 1.1 - Вы имеете возможность назначать новый адрес изображению, уже

загруженному в web-страницу. И в результате, изображение будет загружено с этого нового адреса, заменив на web-странице старое. Рассмотрим к примеру запись:

```

```

Здесь загружается изображение *img1.gif* и получает имя *myImage*. В следующей строке прежнее изображение *img1.gif* заменяется уже на новое - *img2.gif*:

```
document.myImage.src= "img2.src";
```

При этом новое изображение всегда получает тот же размер, что был у старого. И Вы уже не можете изменить размер поля, в котором это изображение размещается.

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

Упреждающая загрузка изображения

Один из недостатков такого подхода может заключаться в том, что после записи в *src* нового адреса начинается процесс загрузки соответствующего изображения. И поскольку этого не было сделано заранее, то еще пройдет некоторое время, прежде чем новое изображение будет передано через Интернет и встанет на свое место. В некоторых ситуациях это допустимо, однако часто подобные задержки неприемлемы. И что же мы можем сделать с этим? Конечно, решением проблемы была бы упреждающая загрузка изображения. Для этого мы должны создать новый объект *Image*. Рассмотрим следующие строки:

```
hiddenImg= new Image();  
hiddenImg.src= "img3.gif";
```

В первой строке создается новый объект *Image*. Во второй строке указывается адрес изображения, которое в дальнейшем будет представлено с помощью объекта *hiddenImg*. Как мы уже видели, запись нового адреса в атрибуте *src* заставляет браузер загружать изображение с указанного адреса. Поэтому, когда выполняется вторая строка нашего примера, начинается загрузка изображения *img2.gif*. Но как подразумевается самим названием *hiddenImg* (“скрытая картинка”), после того, как браузер закончит загрузку, изображение на экране не появится. Оно будет лишь сохранено в памяти компьютера (или точнее в кэше) для последующего использования. Чтобы вызвать изображение на экран, мы можем воспользоваться строкой:

```
document.myImage.src= hiddenImg.src;
```

Но теперь изображение уже немедленно извлекается из кэша и показывается на экране. Таким образом, сейчас мы управляли упреждающей загрузкой изображения. Конечно браузер должен был к моменту запроса закончить упреждающую загрузку, чтобы необходимое изображение было показано без задержки. Поэтому, если Вы должны предварительно загрузить большое количество изображений, то может иметь место задержка, поскольку браузер будет занят загрузкой всех картинок. Вы всегда должны учитывать скорость связи с Интернет - загрузка изображений не станет быстрее, если пользоваться только что показанными командами. Мы лишь пытаемся чуть раньше

загрузить изображение - поэтому и пользователь может увидеть их раньше. В результате и весь процесс пройдет более гладко.

Если у Вас есть быстрая связь с Интернет, то Вы можете не понять, к чему весь этот разговор. О какой задержке все время говорит этот парень? Прекрасно, но еще остаются люди, имеющие более медленный модем, чем 14.4 (Нет, это не я. Я только что заменил свой на 33.6, да ...).

Изменение изображений в соответствии с событиями, инициируемыми самим читателем

Вы можете создать красивые эффекты, используя смену изображений в качестве реакции на определенные события. Например, Вы можете изменять изображения в тот момент, когда курсор мыши попадает на определенную часть страницы.

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

Исходный код этого примера выглядит следующим образом:

```
<a href="#"
onMouseOver="document.myImage2.src='img2.gif'"
onMouseOut="document.myImage2.src='img1.gif'">
</a>
```

При этом могут возникнуть следующие проблемы:

- Читатель пользуется браузером, не имеющим поддержки JavaScript 1.1.
- Второе изображение не было загружено.
- Для этого мы должны писать новые команды для каждого изображения на web-странице.
- Мы хотели бы иметь такой скрипт, который можно было бы использовать во многих web-страницах вновь и вновь, и без больших переделок.

Теперь мы рассмотрим полный вариант скрипта, который мог бы решить эти проблемы. Хотя скрипт и стал намного длиннее - но написав его один раз, Вы не больше будете беспокоиться об этих проблемах. Чтобы этот скрипт сохранял свою гибкость, следует соблюдать два условия:

- Не оговаривается количество изображений - не должно иметь значения, сколько их используется, 10 или 100
- Не оговаривается порядок следования изображений - должна существовать возможность изменять этот порядок без изменения самого кода

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

Рассмотрим скрипт (я внес туда некоторые комментарии):

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

```

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

// *****
// Script from Stefan Koch - Voodoo's Intro to JavaScript
//   http://rummelplatz.uni-mannheim.de/~skoch/js/
//   JS-book: http://www.dpunkt.de/javascript
//   You can use this code if you leave this message
// *****

// ok, у нас браузер с поддержкой JavaScript
var browserOK = false;
var pics;

// -->
</script>

<script language="JavaScript1.1">
<!-- hide

// браузер с поддержкой JavaScript 1.1!
browserOK = true;
pics = new Array();

// -->
</script>

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

var objCount = 0; // количество изображений на web-странице

function preload(name, first, second) {

// предварительная загрузка изображений и размещение их в массиве

if (browserOK) {
pics[objCount] = new Array(3);
pics[objCount][0] = new Image();
pics[objCount][0].src = first;
pics[objCount][1] = new Image();
pics[objCount][1].src = second;
pics[objCount][2] = name;
objCount++;
}
}

function on(name){
if (browserOK) {

```

```

    for (i = 0; i < objCount; i++) {
        if (document.images[pics[i][2]] != null)
            if (name != pics[i][2]) {
                // вернуть в исходное состояние все другие изображения
                document.images[pics[i][2]].src = pics[i][0].src;
            } else {
                // показывать вторую картинку, поскольку курсор пересекает данное изображение
                document.images[pics[i][2]].src = pics[i][1].src;
            }
        }
    }
}

```

```

function off(){
    if (browserOK) {
        for (i = 0; i < objCount; i++) {
            // вернуть в исходное состояние все изображения
            if (document.images[pics[i][2]] != null)
                document.images[pics[i][2]].src = pics[i][0].src;
        }
    }
}

```

// заранее загружаемые изображения - Вы должны здесь указать
 // изображения, которые нужно загрузить заранее, а также объект Image,
 // к которому они относятся (первый аргумент). Именно эту часть
 // нужно корректировать, если Вы хотите использовать скрипт
 // применительно к другим изображениям (конечно это не освобождает
 // Вас от обязанности подредактировать в документе также и раздел body)

```

preload("link1", "img1f.gif", "img1t.gif");
preload("link2", "img2f.gif", "img2t.gif");
preload("link3", "img3f.gif", "img3t.gif");

```

```

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

```

```

<body>
<a href="link1.htm" onMouseOver="on('link1')"
  onMouseOut="off()">
  </a>

  <a href="link2.htm" onMouseOver="on('link2')"
    onMouseOut="off()">
    </a>

  <a href="link3.htm" onMouseOver="on('link3')"

```

```
onMouseOut="off() ">
</a>
</body>
</html>
```

Данный скрипт помещает все изображения в массив `pics`. Создает этот массив функция `preload()`, которая вызывается в самом начале. Вызов функции `preload()` выглядит просто так:

```
preload("link1", "img1f.gif", "img1t.gif");
```

Это означает, что скрипт должен загрузить с сервера два изображения: `img1f.gif` и `img1t.gif`. Первое из них - это та картинка, которая будет представлена, пока курсор мыши не попадает в область изображения. Когда же пользователь помещает курсор мыши на изображение, то появляется вторая картинка. При вызове функции `preload()` в качестве первого аргумента мы указываем слово `"link1"` и тем самым задаем на web-странице объект `Image`, которому и будут принадлежать оба предварительно загруженных изображения. Если Вы посмотрите в нашем примере в раздел `<body>`, то обнаружите изображение с тем же именем `link1`. Мы пользуемся не порядковым номером, а именно имя изображения для того, чтобы иметь возможность переставлять изображения на web-странице, не переписывая при этом сам скрипт.

Обе функции `on()` и `off()` вызываются посредством программ обработки событий `onMouseOver` и `onMouseOut`. Поскольку сам элемент `image` не может отслеживать события `MouseOver` и `MouseOut`, то мы обязаны сделать на этих изображениях еще и ссылки. Можно видеть, что функция `on()` возвращает все изображения, кроме указанного, в исходное состояние. Делать это необходимо потому, что в противном случае выделенными могут оказаться сразу несколько изображений (дело в том, что событие `MouseOut` не будет зарегистрировано, если пользователь переместит курсор с изображения сразу за пределы окна).

Изображения - без сомнения могучее средство улучшения Вашей web-страницы. Объект `Image` дает Вам возможность создавать действительно сложные эффекты. Однако заметим, что не каждое изображение или программа JavaScript способно улучшить Вашу страницу. Если Вы пройдетесь по Сети, то сможете увидеть множество примеров, где изображения использованы самым ужасным способом. Не количество изображений делает Вашу web-страницу привлекательной, а их качество. Сама загрузка 50 килобайт плохой графики способна вызвать раздражение. При создании специальных эффектов с изображениями с помощью JavaScript помните об этом и ваши посетителями/клиентами будут чаще возвращаться на Ваши страницы.

©1996,1997 by Stefan Koch
e-mail: skoch@rumms.uni-mannheim.de
<http://rummelplatz.uni-mannheim.de/~skoch/>
Моя книга по JavaScript: <http://www.dpunkt.de/javascript>