

ВВЕДЕНИЕ В JAVASCRIPT ДЛЯ МАГА

© 1996, 1997 Стефан Кох (Stefan Koch)

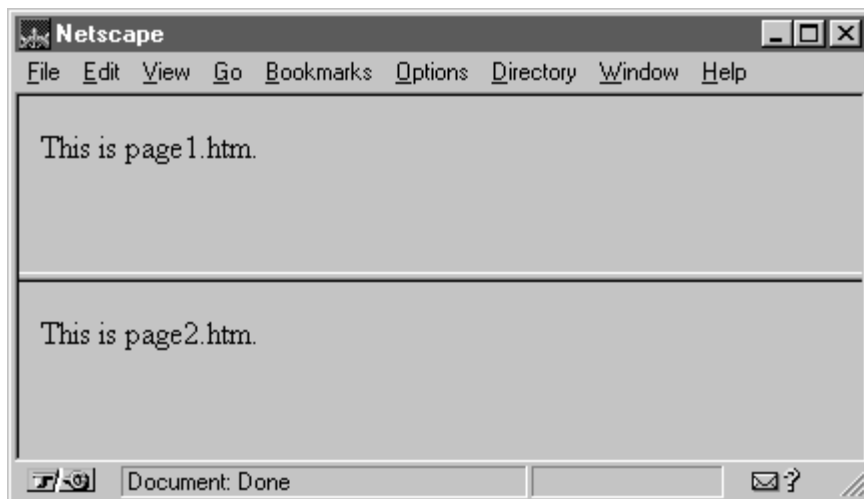
Часть 3: Фреймы

Создание фреймов

Один из часто задаваемых вопросов - как фреймы и JavaScript могут работать вместе. Сначала я хотел бы объяснить, что такое фреймы и для чего они могут использоваться. Затем мы рассмотрим, как можно использовать JavaScript совместно с фреймами. В общем случае окно браузера может быть разбито в несколько отдельных фреймов. Это означает, что фрейм определяется как некое выделенное в окне браузера поле в форме прямоугольника. Каждый из фреймов выдает на экран содержимое собственного документа (в большинстве случаев это документы HTML). Таким образом, Вы можете, к примеру, создать два фрейма. В первый такой фрейм Вы можете загрузить "домашнюю страницу" фирмы Netscape, а во второй - фирмы Microsoft. Хотя создание фреймов является задачей языка HTML, я бы хотел все же описать здесь основные моменты этого процесса. Для создания фреймов Вам необходимо два тэга: `<frameset>` и `<frame>`. HTML-страница, создающая два фрейма, в общем случае может выглядеть следующим образом:

```
<html>
<frameset rows="50%,50%">
  <frame src="page1.htm" name="frame1">
  <frame src="page2.htm" name="frame2">
</frameset>
</html>
```

В результате будут созданы два фрейма. Вы можете заметить, что во фрейме `<frameset>` мы используем свойство `rows`. Это означает, два наших фрейма будут расположены друг над другом. В верхний фрейм будет загружена HTML-страница `page1.htm`, а в нижнем фрейме разместится документ `page2.htm`. Окончательно созданная структура фреймов будет выглядеть следующим образом:

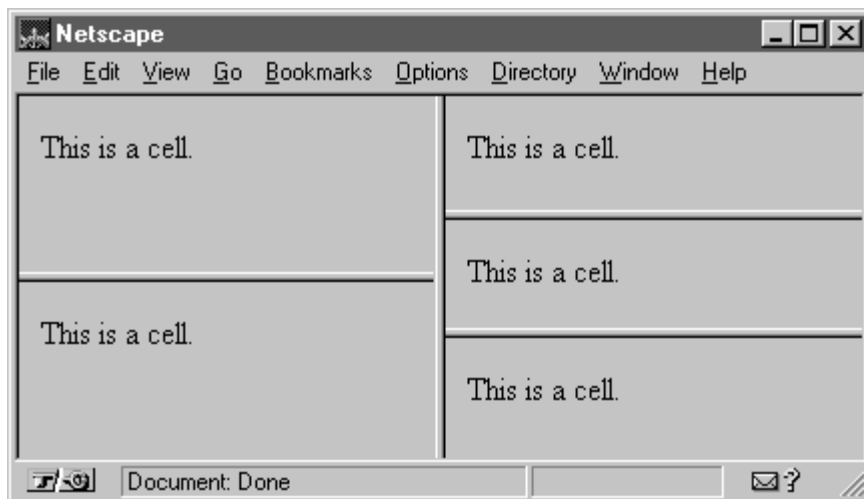


Если Вы хотите, чтобы документы располагались не друг над другом, а рядом то, Вам следует в тэге `<frameset>` писать *rows*, а *cols*. Фрагмент "50%,50%" сообщает, насколько велики должны быть оба получившихся окна. Вы имеете также возможность записать "50%,*", если не хотите утруждать себя расчетами, насколько велик должен быть второй фрейм, чтобы в сумме получалась все те же 100%. Вы можете также задать размер фрейма в пикселах, для чего достаточно после числа не ставить символ %. Любому фрейму можно присвоить уникальное имя, воспользовавшись в тэге `<frame>` атрибутом *name*. Такая возможность пригодится нам в языке JavaScript для доступа к фреймам.

При создании web-страниц Вы можете использовать несколько вложенных тэгов `<frameset>`. Следующий пример я нашел в документации, предоставляемой фирмой Netscape, (и слегка изменил его):

```
<frameset cols="50%,50%">
  <frameset rows="50%,50%">
    <frame src="cell.htm">
    <frame src="cell.htm">
  </frameset>
</frameset>
<frameset rows="33%,33%,33%">
  <frame src="cell.htm">
  <frame src="cell.htm">
  <frame src="cell.htm">
</frameset>
</frameset>
```

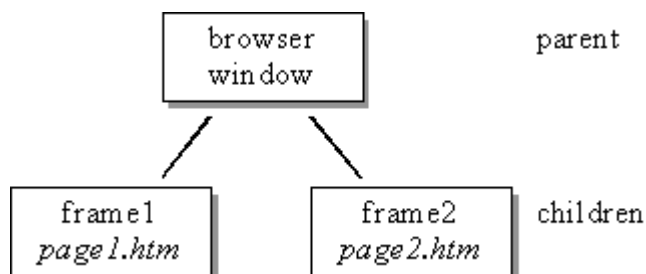
В результате созданная структура фреймов будет выглядеть следующим образом:



Вы можете задать толщину границы между фреймами, воспользовавшись в тэге `<frameset>` параметром `border`. Запись `border=0` означает, что Вы не хотите, чтобы между тэгами имелась какая-либо граница (в Netscape 2. x такой механизм не работает).

Фреймы и JavaScript

А теперь давайте посмотрим, как JavaScript “видит” фреймы, присутствующие в окне браузера. Для этой цели мы создадим два фрейма, как было показано в первом примере этой части описания. Как мы уже видели, JavaScript организует все элементы, представленные на web-странице, в виде некой иерархической структуры. То же самое относится и к фреймам. На следующем рисунке показана иерархия объектов, представленных в первом примере:



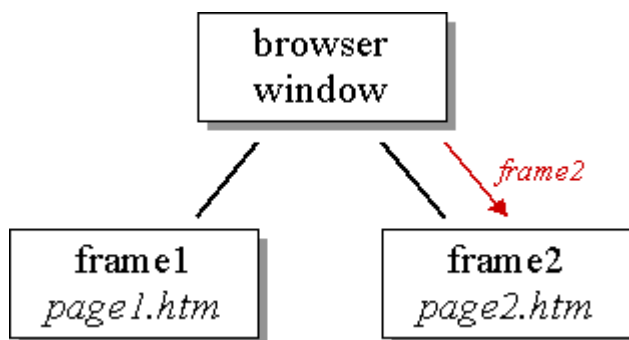
В вершине иерархии находится окно браузера (browser window). В данном случае он разбито на два фрейма. Таким образом, окно, как объект, является родоначальником, родителем данной иерархии (parent), а два фрейма - соответственно, его потомки (children). Мы присвоили этим двум фреймам уникальные имена - frame1 и frame2. И с помощью этих имен мы можем обмениваться информацией с двумя указанными фреймами.

С помощью скрипта можно решить следующую задачу: допустим посетитель активирует некую ссылку в первом фрейме, однако соответствующая страница должна загружаться не в этот же фрейм, а в другой. Примером такой задачи может служить составление меню (или навигационных панелей), где один фрейм всегда остается неизменным, но предлагает посетителю несколько различных ссылок для дальнейшего изучения данного сайта. Чтобы решить эту задачу, мы должны рассмотреть на три случая:

- *главное окно/фрейм получает доступ к фрейму-потомку*
- *фрейм-потомок получает доступ к родительскому окну/фрейму*
- *фрейм-потомок получает доступ к другому фрейму-потомку*

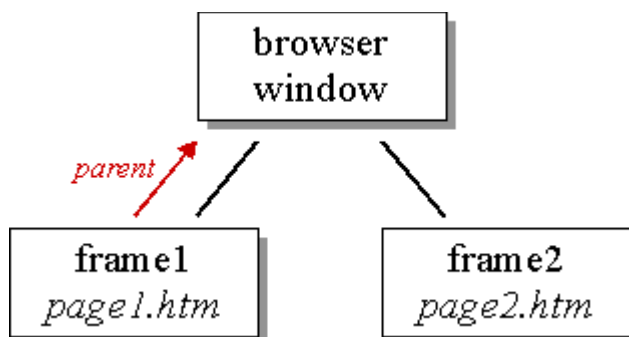
С точки зрения объекта “окно” (window) два указанных фрейма называются *frame1* и *frame2*. Как можно видеть на предыдущем рисунке, существует прямая взаимосвязь между родительским окном и каждым фреймом. Так образом, если Вы пишете скрипт для родительского окна - то есть для страницы, создающей эти фреймы - то можете обращаться к этим фреймам, просто называя их по имени. Например, можно написать:

frame2.document.write("Это сообщение передано от родительского окна.");



В некоторых случаях Вам понадобится, находясь во фрейме, получать доступу к родительскому окну. Например, это бывает необходимо, если Вы хотите при следующем переходе избавиться от фреймов. В таком случае удаление фреймов означает лишь загрузку новой страницы вместо содержавшей фреймы. В нашем случае это загрузка страницы в родительское окно. Сделать это нам поможет доступ к родительскому окну (или родительскому фрейму) из фреймов, являющихся его потомками. Чтобы загрузить новый документ, мы должны внести в *location.href* новый адрес URL. Поскольку мы хотим избавиться от фреймов, следует использовать объект *location* из родительского окна. (Напомним, что в каждый фрейм можно загрузить собственную страницу, то мы имеем для каждого фрейма собственный объект *location*). Итак, мы можем загрузить новую страницу в родительское окно с помощью команды:

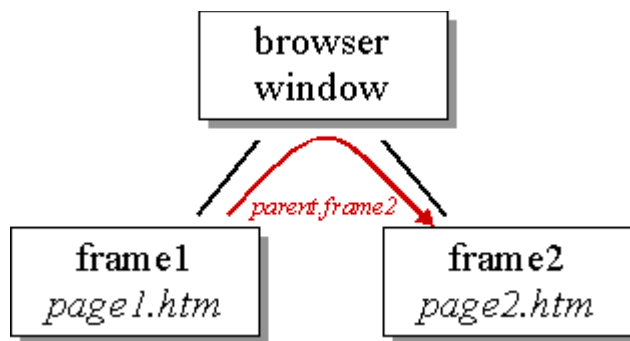
parent.location.href= "http://...";



И наконец, очень часто Вам придется решать задачу обеспечения доступа с одного фрейма-потомка к другому такому же фрейму-потомку. Итак, как можно, находясь в первом фрейме, записать что-либо во второй - то есть, которой командой следует воспользоваться на HTML-странице *page1.htm*? Как можно увидеть на нашем рисунке, между двумя этими фреймами нет никакой прямой связи. И потому мы не можем

просто так вызвать *frame2*, находясь в фрейме *frame1*, который просто ничего не знает о существовании второго фрейма. С точки же зрения родительского окна второй фрейм действительно существует и называется *frame2*, а к самому родительскому окну можно обратиться из первого фрейма по имени *parent*. Таким образом, чтобы получить доступ к объекту *document*, разместившемуся во втором фрейме, мы должны написать следующее,:

```
parent.frame2.document.write("Привет, это вызов из первого фрейма.");
```



Навигационные панели

Давайте рассмотрим, как создаются навигационные панели. В одном фрейме мы создаем несколько ссылок. Однако, если посетитель активирует какую-либо из них, соответствующая страница будет помещена не в тот же самый фрейм, а в соседний. Сперва нам необходимо написать скрипт, создающий указанные фреймы. Такой документ выглядит точно так же, как и тот, что мы рассматривали ранее в этой части описания:

frames3.htm

```
<html>
<frameset rows="80%,20%">
  <frame src="start.htm" name="main">
  <frame src="menu.htm" name="menu">
</frameset>
</html>
```

Здесь *start.htm* - это та страница, которая первоначально будет показана в главном фрейме (*main*). У нас нет никаких специальных требований к содержимому этой страницы. Следующая web-страница будет загружена во фрейм "*menu*":

menu.htm

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

```
function load(url) {
  parent.main.location.href= url;
```

```

}

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

<a href="javascript:load('first.htm')">first</a>
<a href="second.htm" target="main">second</a>
<a href="third.htm" target="_top">third</a>

</body>
</html>

```

Здесь Вы можете увидеть несколько способов загрузки новой страницы во фрейм *main*. В первой ссылке для этой цели используется функция *load()*. Давайте посмотрим, как это делается:

```
<a href="javascript:load('first.htm')">first</a>
```

Как Вы можете видеть, вместо явной загрузки новой страницы мы предлагаем браузеру выполнить некую команду на языке JavaScript - для этого мы всего лишь должны воспользоваться параметром *javascript* вместо обычного *href*. Далее, внутри скобок можно увидеть *'first.htm'*. Эту строка передается в качестве аргумента функции *load()*. Сама же функция *load()* определяется следующим образом:

```

function load(url) {
    parent.main.location.href= url;
}

```

Здесь Вы можете увидеть, что внутри скобок написано *url*. Это означает, что в нашем примере строка *'first1.htm'* при вызове функции заносится в переменную *url*. И эту новую переменную теперь можно использовать при работе внутри функций *load()*. Позже мы познакомимся с другими примерами использования важной концепции переменных.

Во второй ссылке присутствует параметр *target*. На самом деле это уже не относится к JavaScript. Это одна из конструкций языка HTML. Как видно, мы всего лишь указываем имя необходимого фрейма. Заметим, что в этом случае мы не обязаны ставить перед именем указанного фрейма слово *parent*, что, честно говоря, несколько смущает. Причина такого отступления от правил кроется в том, что параметр *target* - это функция языка HTML, а не JavaScript. И на примере третьей ссылки Вы можете видеть, как с помощью *target* можно избавиться от фреймов.

А если Вы хотите избавиться от фреймов с помощью функции *load()*, то Вам необходимо написать в ней лишь *parent.location.href= url*.

Итак, который способ Вам следует выбрать? Это зависит от вашего скрипта и от того, что собственно Вы хотите сделать. Параметр *target* использовать очень просто. Вы можете воспользоваться им, если хотите всего лишь загрузить новую страницу в другой фрейм. Решение на основе языка JavaScript (примером этого служит первая ссылка) обычно используется, если Вы хотите, чтобы при активизации ссылки произошло несколько вещей. Одна из наиболее часто возникающих проблем такого

рода состоит в том, чтобы разом загрузить две страницы в два различных фрейма. И хотя Вы могли бы решить эту задачи с помощью параметра *target*, использование функции JavaScript в этом случае более предпочтительно. Предположим, мы имеем три фрейма с именами *frame1*, *frame2* и *frame3*. Допустим посетитель активирует ссылку в *frame1*. И мы хотим, чтобы при этом в два других фрейма загружались две различные web-страницы. В качестве решения этой задачи Вы можете, например, воспользоваться функцией:

```
function loadtwo() {  
    parent.frame1.location.href= "first.htm";  
    parent.frame2.location.href= "second.htm";  
}
```

Если же Вы хотите сделать функцию более гибкой, то можете воспользоваться возможностью передачи переменной в качестве аргумента. Результат будет выглядеть как:

```
function loadtwo(url1, url2) {  
    parent.frame1.location.href= url1;  
    parent.frame2.location.href= url2;  
}
```

После этого можно организовать вызов функции: *loadtwo("first.htm", "second.htm")* или *loadtwo("third.htm", "forth.htm")*. Очевидно, передача аргументов делает вашу функцию более гибкой. В результате Вы можете использовать ее многократно и в различных контекстах.

©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>