

VOODOO'S INTRODUCTION TO JAVASCRIPT

© 1996, 1997 by Stefan Koch

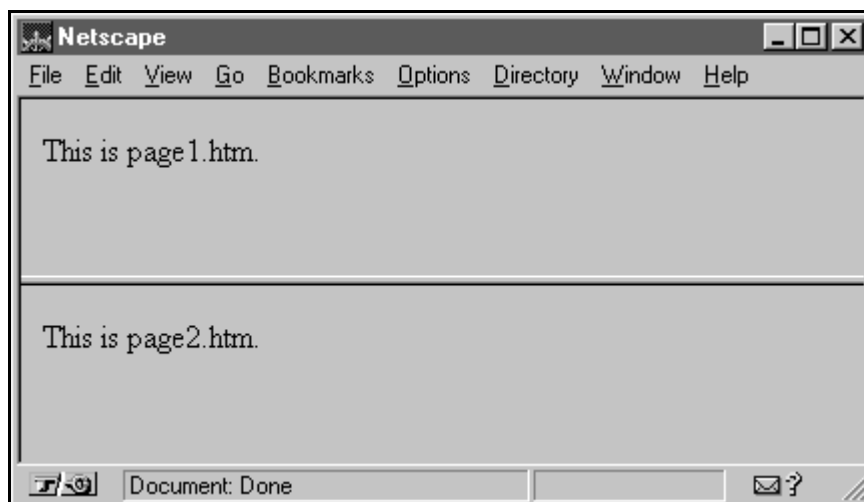
Part 3: Frames

Creating frames

An often asked question is how frames and JavaScript work together. First I want to explain what frames are and what they can be used for. After this we will see how we can use JavaScript in combination with frames. The browser window can be split up into several frames. This means a frame is a square area inside the browser window. Each frame displays its own document (most of the time HTML-documents). So you can for example create two frames. In the first frame you load the homepage of Netscape and in the second frame you load the homepage of Microsoft. Although creating frames is a HTML-problem I want to describe the basic things. For creating frames you need two tags: `<frameset>` and `<frame>`. A HTML-page creating two frames might look like this:

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

This will produce two frames. You can see that we use the `rows` property in the `<frameset>` tag. This means the two frames lie above each other. The upper frame loads the HTML-page `page1.htm` and the lower frame displays the document `page2.htm`. The created frame-structure looks like this:



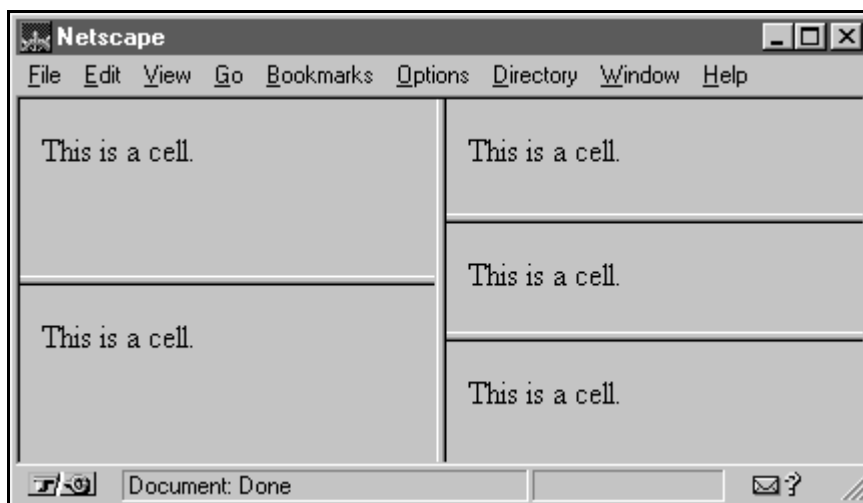
If you want to have columns instead of rows you write `cols` instead of `rows` in the `<frameset>`

tag. The "50%,50%" part specifies how large the two windows are. You can also write "50%,*" if you do not want to calculate how large the second frame must be in order to get 100%. You can specify the size in pixels by omitting the % symbol. Every frame gets an unique name with the name property in the <frame> tag. This will help us when accessing the frames through JavaScript.

You can have several nested <frameset> tags. I've found this example in the documentation provided by Netscape (I just modified it a little bit):

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

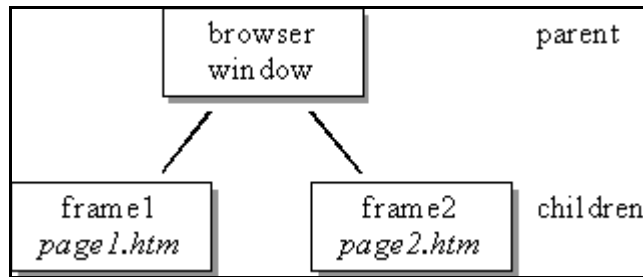
The created frame structure looks like this:



You can set the size of the border through the border property in the <frameset> tag. *border=0* means that you do not want to have a border (does not work with Netscape 2.x).

Frames and JavaScript

Now we want to have a look at how JavaScript 'sees' the frames in a browser window. For this we are going to create two frames as shown in the first example of this part. We have seen that JavaScript organizes all elements on a webpage in a hierarchy. This is the same with frames. The following image shows the hierarchy of the first example of this part:



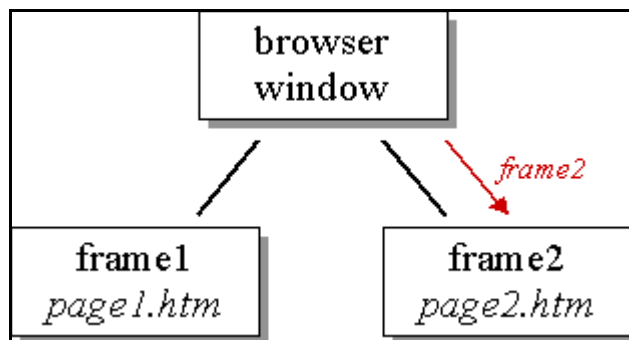
At the top of the hierarchy is the browser window. This window is split up into two frames. The window is the parent in this hierarchy and the two frames are the children. We gave the two frames the unique names *frame1* and *frame2*. With the help of these names we can exchange information between the two frames.

A script might have to solve the following problem: The user clicks on a link in the first frame - but the page shall be loaded in the second frame rather than in the first frame. This can for example be used for menubars (or navigationbars) where one frame always stays the same and offers several different links to navigate through a homepage. We have to look at three cases:

- *parent window/frame accesses child frame*
- *child frame accesses parent window/frame*
- *child frame accesses another child frame*

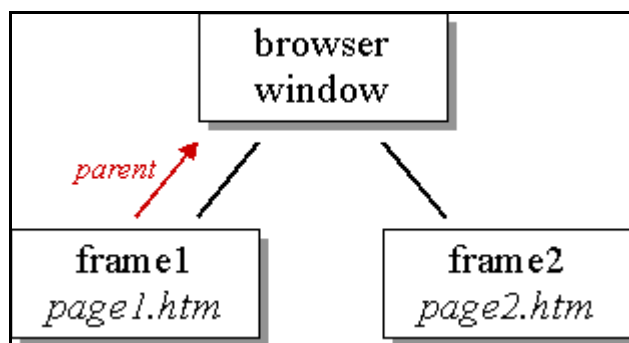
From the window's point of view the two frames are called *frame1* and *frame2*. You can see in the image above that there is a direct connection from the parent window to each frame. So if you have a script in the parent window - this means in the page that creates the frames - and you want to access the frames you can just use the name of the frame. For example you can write:

```
frame2.document.write("A message from the parent window.");
```



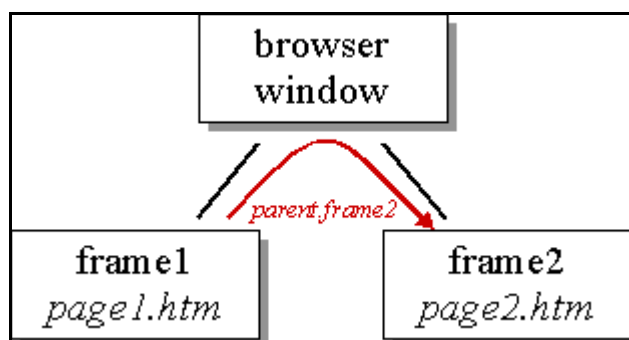
Sometimes you want to access the parent window from a frame. This is needed for example if you want to remove the frames. Removing the frames just means to load a new page instead of the page which created the frames. This is in our case the page in the parent window. We can access the parent window (or parent frame) from the child frames with `parent`. In order to load a new document we have to assign a new URL to `location.href`. As we want to remove the frames we have to use the location-object of the parent window. As every frame can load its own page we have a different location-object for each frame. We can load a new page into the parent window with the command:

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



Very often you will be faced with the problem to access one child frame from another child frame. So how can you write something from the first frame to the second frame - this means which command do you have to use in the HTML-page called *page1.htm*? In our image you can see that there is no direct connection between the two frames. This means we cannot just call *frame2* from the frame *frame1* as this frame does not know anything about the existence of the second frame. From the parent window's point of view the second frame is called *frame2* and the parent window is called *parent* seen from the first frame. So we have to write the following in order to access the document-object of the second frame:

```
parent.frame2.document.write("Hi, this is frame1 calling.");
```



Navigationbars

Let's have a look at a navigationbar. We will have several links in one frame. If the user clicks on these links the pages won't show up in the same frame - they are loaded in the second frame. First we need a script which creates the frames. This document looks like the first example we had in this part:

frames3.htm

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

The *start.htm* page is the entry page which will be displayed in the main frame at the beginning.

There are no special requirements for this page. The following page is loaded into the frame *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>
```

Here you can see different ways for loading a new page into the frame *main*. The first link uses the function *load()*. Have a look at how this function is called:

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

You can see that we can let the browser execute JavaScript code instead of loading another page - we just have to use *javascript:* in the *href* property. You can see that we write *'first.htm'* inside the brackets. This string is passed to the function *load()*. The function *load()* is defined through:

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

There you can see that we write *url* inside the brackets. This means that the string *'first1.htm'* is stored in the variable *url*. Inside the *load()* function we can now use this variable. We will see further examples of this important concept of variable passing later on.

The second link uses the *target* property. Actually this isn't JavaScript. This is a HTML-feature. You see that we just have to specify the name of the frame. Please note that we must not put *parent* before the name of the frame. This might be a little bit confusing. The reason for this is that *target* is HTML and not JavaScript. The third link shows you how to remove the frames with the *target* property.

If you want to remove the frames with the *load()* function you just have to write *parent.location.href= url* inside the function.

So which way should you choose? This depends on your script and what you want to do. The

target property is very simple. You might use it if you just want to load the page in another frame. The JavaScript solution (i.e. the first link) is normally used if you want to do several things as a reaction to the click on the link. One common problem is to load two pages at once in two different frames. Although you could solve this with the target property using a JavaScript function is more straightforward. Let's assume you have three frames called *frame1*, *frame2* and *frame3*. The user clicks on a link in *frame1*. Then you want to load two different pages in the two other frames. You can use this function for example:

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

If you want to keep the function more flexible you can use variable passing here as well. This looks like this:

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

Then you can call this function with *loadtwo("first.htm", "second.htm")* or *loadtwo("third.htm", "forth.htm")*. Variable passing makes your function more flexible. You can use it over and over again in different contexts.

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