

VOODOO'S INTRODUCTION TO JAVASCRIPT

© 1996, 1997 by Stefan Koch

About this tutorial

Online version

This tutorial is an introduction to JavaScript. I have started this tutorial as an online tutorial where you can test all examples immediately. As the tutorial grew larger a printable version was required. It can be quite exhausting to read long parts before the monitor. It is obvious that the printable version cannot substitute the online version completely. You can find the online version at <http://rummelplatz.uni-mannheim.de/~skoch/js/> or at <http://www.webconn.com/java/javascript/intro> (US mirror).

JavaScript book and examples

I have written a JavaScript book recently. It is called 'JavaScript - Einfuehrung, Programmierung und Referenz' and is written in german. I have build up a homepage for this book which can be found at <http://www.dpunkt.de/javascript/>

There you will find information about my book and some interesting JavaScript examples. The pages are both in german and english - so do not hesitate to have a look at the JavaScript examples even if you do not know any german.

Title: *JavaScript - Einfuehrung, Programmierung und Referenz* (german)

Author: *Stefan Koch*

Publisher: *dpunkt.verlag*

ISBN: *3-920993-64-0*

Homepage: <http://www.dpunkt.de/javascript/>

Part 1: First steps

What is JavaScript

JavaScript is a new scripting language which is being developed by Netscape. With JavaScript you can easily create interactive web-pages. This tutorial shows you what can be done with JavaScript - and more importantly *how* it is done.

JavaScript is not Java!

Many people believe that JavaScript is the same as Java because of the similar names. This is **not** true though. I think it would go too far at the moment to show you all the differences - so just memorize that JavaScript is *not* Java. For further information on this topic please read the

introduction provided by Netscape or my book :-)

Running JavaScript

What is needed in order to run scripts written in JavaScript? You need a JavaScript-enabled browser - for example the Netscape Navigator (since version 2.0) or the Microsoft Internet Explorer (MSIE - since version 3.0). Since these two browsers are widely spread many people are able to run scripts written in JavaScript. This is certainly an important point for choosing JavaScript to enhance your web-pages.

Of course you need a basic understanding of HTML before reading this tutorial. You can find many good online resources covering HTML. Best you make an online search for 'html' at Yahoo in order to get more information on HTML.

Embedding JavaScript into a HTML-page

JavaScript code is embedded directly into the HTML-page. In order to see how this works we are going to look at an easy example:

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
<script language="JavaScript">
  document.write("This is JavaScript!")
</script>
<br>
Back in HTML again.
</body>
</html>
```

At the first glance this looks like a normal HTML-file. The only new thing is the part:

```
<script language="JavaScript">
  document.write("This is JavaScript!")
</script>
```

This is JavaScript. In order to see this script working save this code as a normal HTML-file and load it into your JavaScript-enabled browser. Here is the output generated by the file (if you are using a JavaScript browser you will see 3 lines of output):

```
This is a normal HTML document.
This is JavaScript!
Back in HTML again.
```

I must admit that this script isn't very useful - this could have been written in pure HTML more easily. I only wanted to demonstrate the `<script>` tag to you. Everything between the `<script>` and the `</script>` tag is interpreted as JavaScript code. There you see the use of `document.write()` - one of the most important commands in JavaScript programming. `document.write()` is

used in order to write something to the actual document (in this case this is the HTML-document). So our little JavaScript program writes the text *This is JavaScript!* to the HTML-document.

Non-JavaScript browsers

What does our page look like if the browser does not understand JavaScript? A non-JavaScript browser does not know the `<script>` tag. It ignores the tag and outputs all following code as if it was normal text. This means the user will see the JavaScript-code of our program inside the HTML-document. This was certainly not our intention. There is a way for hiding the source code from older browsers. We will use the HTML-comments `<!-- -->`. Our new source code looks like this:

```
<html>
<body>
<br>
This is a normal HTML document.
<br>
  <script language="JavaScript">
    <!-- hide from old browsers

      document.write("This is JavaScript!")

    // -->
  </script>
<br>
Back in HTML again.
</body>
</html>
```

The output in a non-JavaScript browser will then look like this:

```
This is a normal HTML document.
Back in HTML again.
```

Without the HTML-comment the output of the script in a non-JavaScript browser would be:

```
This is a normal HTML document.
document.write("This is JavaScript!")
Back in HTML again.
```

Please note that you cannot hide the JavaScript source code completely. What we do here is to prevent the output of the code in old browsers - but the user can see the code through 'View document source' nevertheless. There is no way to hinder someone from viewing your source code (in order to see how a certain effect is done).

Events

Events and event handlers are very important for JavaScript programming. Events are mostly caused by user actions. If the user clicks on a button a Click-event occurs. If the mousepointer

moves across a link a `MouseOver`-event occurs. There are several different events. We want our JavaScript program to react to certain events. This can be done with the help of event-handlers. A button might create a popup window when clicked. This means the window should pop up as a reaction to a `Click`-event. The event-handler we need to use is called `onClick`. This tells the computer what to do if this event occurs. The following code shows an easy example of the event-handler `onClick`:

```
<form>
<input type="button" value="Click me" onClick="alert('Yo')">
</form>
```

(The online version lets you test this script immediately)

There are a few new things in this code - so let's take it step by step. You can see that we create a form with a button (this is basically a HTML-problem so I won't cover it here). The new part is `onClick="alert('Yo')"` inside the `<input>` tag. As we already said this defines what happens when the button is pushed. So if a `Click`-event occurs the computer shall execute `alert('Yo')`. This is JavaScript-code (Please note that we do not use the `<script>` tag in this case). `alert()` lets you create popup windows. Inside the brackets you have to specify a string. In our case this is `'Yo'`. This is the text which shall be shown in the popup window. So our script creates a window with the contents `'Yo'` when the user clicks on the button.

One thing might be a little bit confusing: In the `document.write()` command we used double quotes `"` and in combination with `alert()` we use only single quotes `'` - why? Basically you can use both. But in the last example we wrote `onClick="alert('Yo')"` - you can see that we used both double and single quotes. If we wrote `onClick="alert("Yo")"` the computer would get confused as it isn't clear which part belongs to the `onClick` event-handler and which not. So you have to alternate with the quotes in this case. It doesn't matter in which order you use the quotes - first double quotes and then single quotes or vice versa. This means you can also write `onClick='alert("Yo")'`.

There are many different event-handlers you can use. We will get to know some during this tutorial - but not all. So please refer to a reference if you want to know what kind of other event-handlers do exist.

If you are using the Netscape Navigator the popup window will contain the text JavaScript alert. This is a security restriction. You can create a similar popup window with the `prompt()` method. This window accepts an input. A malicious script could imitate a system message and ask for a certain password. The text in the popup window shows that the window comes from your web browser and not from your operating system. As this is a security restriction you cannot remove this message.

Functions

We will use functions in most of our JavaScript programs. Therefore I will talk about this important concept already now. Basically functions are a way for bundling several commands together. Let's write a script which outputs a certain text three times. Consider the following approach:

```
<html>
<script language="JavaScript">
```

```
<!-- hide

document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");

document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");

document.write("Welcome to my homepage!<br>");
document.write("This is JavaScript!<br>");

// -->
</script>
</html>
```

This will write out the text

```
Welcome to my homepage!
This is JavaScript!
```

three times. Look at the source code - writing the code three times brings out the right result. But is this very efficiently? No, we can solve this better. How about this code which does the same:

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

function myFunction() {
  document.write("Welcome to my homepage!<br>");
  document.write("This is JavaScript!<br>");
}

myFunction();
myFunction();
myFunction();

// -->
</script>
</html>
```

In this script we define a function. This is done through the lines:

```
function myFunction() {
  document.write("Welcome to my homepage!<br>");
  document.write("This is JavaScript!<br>");
}
```

The commands inside the {} belong to the function *myFunction()*. This means that our two *document.write()* commands are bundled together and can be executed through a function call. In our example we have three function calls. You can see that we write *myFunction()* three times

just below the definition of the function. These are the three function calls. This means that the contents of the function is being executed three times. This is a very easy example of a function. You might wonder why functions are so important. While reading this tutorial you will certainly realize the benefits of functions. Especially variable passing makes our scripts really flexible - we will see what this is later on.

Functions can also be used in combination with event-handlers. Please consider this example:

```
<html>
<head>

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

function calculation() {
  var x= 12;
  var y= 5;

  var result= x + y;

  alert(result);
}

// -->
</script>

</head>
<body>

<form>
<input type="button" value="Calculate" onClick="calculation()">
</form>

</body>
</html>
```

(The online version lets you test this script immediately)

The button calls the function *calculation()*. You can see that the function does certain calculations. For this we are using the variables *x*, *y* and *result*. We can define a variable with the keyword *var*. Variables can be used to store different values - like numbers, text strings etc. The line *var result= x + y;* tells the browser to create a variable *result* and store in it the result of *x + y* (i.e. *5 + 12*). After this operation the variable *result* is *17*. The command *alert(result)* is in this case the same as *alert(17)*. This means we get a popup window with the number *17* in it.

©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