JavaScript – Diverses

Inhalt:

1. Die Historie, Objekt „history“
2. „Keydown“-Event-Handler – jQuery
3. Diverses….

1)Die Historie, Objekt „history“

Das Objekt „history“ arbeitet mit der Historie des Browsers. Man kann:

- Mit der Methode „back()“ die vorherige Seite anzeigen lassen
- Mit der Methode „forward()“ die nächste Seite anzeigen lassen, falls man sich schon in der Historie bewegt hat
- Mit der Methode „go()“ zu einer belieben Seite in der Historie springen, dabei mit einem negativen Wert als Parameter zu einer der vorherigen Seiten und mit einem positiven Wert zu einer der nächsten Seiten.
- Die Eigenschaft „length“ zeigt die aktuelle Länge der Historie an.

4.handler_history.html

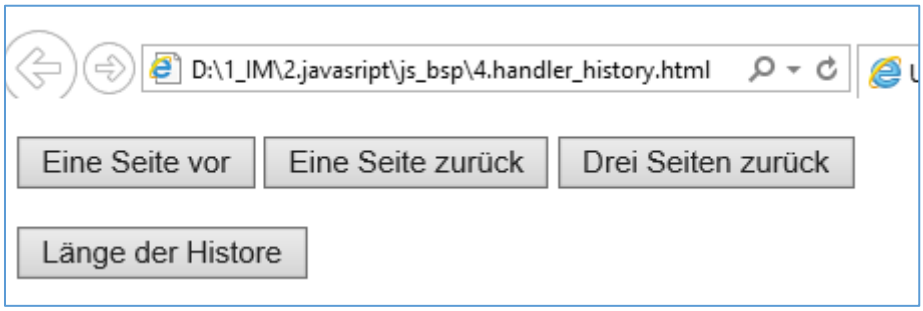
Code | Teilen | Entwurf | Live-Ansicht | Titel: Unbenanntes Dokument

Keine Syntaxfehler.

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Unbenanntes Dokument</title>
6  <script>
7  function meinHandler(id, ereignis, funktion) {
8      if(window.addEventListener)
9      document.getElementById(id)
10     .addEventListener(ereignis, funktion, false);
11     else if(window.attachEvent)
12     document.getElementById(id)
13     .attachEvent("on" + ereignis, funktion);
14 }
15 </script>
16 </head>
17
18 <body>
19 <p><input id="idVorwaerts" type="button" value="Eine Seite vor">
20    <input id="idZurueck" type="button" value="Eine Seite zurück">
21    <input id="idGo" type="button" value="Drei Seiten zurück"></p>
22 <p><input id="idLaenge" type="button" value="Länge der Histore"></p>
23 <script ></script>
24 <script>
25   meinHandler("idVorwaerts", "click", function() { history.forward(); } );
26   meinHandler("idZurueck", "click", function() { history.back(); } );
27   meinHandler("idGo", "click", function() { history.go(-3); } );
28   meinHandler("idLaenge", "click", function() { alert(history.length); } );
29 </script>
30 </body>
31 </html>
```

Im Head wird geprüft, ob der Browser die jeweilige Methode kennt. Man spricht von der sogenannten Browserweiche. Für den Internet Explorer gilt „attachEvent()“ und für andere Browser „addEventListener()“.

Ergebnis:



Quelle:

Thomas Theis, in: Einstieg in JavaScript, 2015 Galileo Press, Bonn, S.201-202, 112-116

2)„Keydown“-Event-Handler – jQuery

JavaScript kann die Tastatur durch „Keyboard-Events“ (Tastaturereignisse) überwachen. Sobald der Nutzer eine Taste auf der Tastatur drückt, wird ein Tastaturereignis erzeugt. Dabei kann man feststellen, welche Taste gedrückt wurde. Diese Information kann man dann im Code verwenden.

So kann man z.B. einen Ball nach links, rechts, oben oder unten verschieben, wenn der Nutzer die Links-, Rechts-, Auf- oder Abwärtspfeiltaste drückt.

Man verwendet dazu das Ereignis „keydown“, das jedes Mal ausgelöst wird, wenn der Nutzer eine Taste drückt. Mit jQuery fügt man dem keydown-Ereignis einen Event-Handler hinzu. Auf diese Weise kann die Event-Handler-Funktion herausfinden, welche Taste gedrückt wurde, sobald ein keydown-Ereignis auftritt, und dann entsprechend antworten.

Erzeuge die „keydown.html“ Datei:

```
8   <body>
9   <canvas id="canvas" width="400" height="400"></canvas>
10  <script src="js/jquery-2.2.0.js"></script>
11  <script>
12  $("body").keydown(function (event) {
13      document.write(event.keyCode);
14  });
15  </script>
16  </body>
```

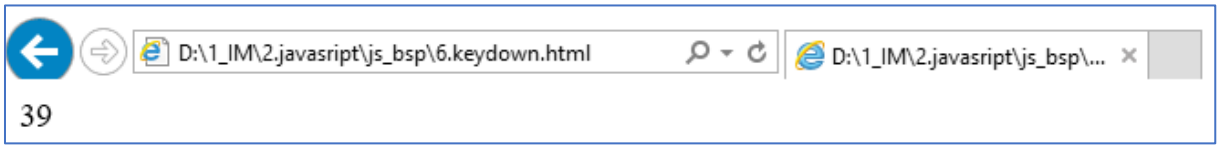
In Zeile 12 wird mit der jQuery-Funktion $ das body-Element ausgewählt und dann wird die Methode „keydown“ aufgerufen.

Das Argument der Methode „keydown“ ist eine Funktion, die jedes Mal aufgerufen wird, wenn eine Taste gedrückt wird. Die Informationen über das keydown-Ereignis werden durch das event-Objekt an die Funktion übergeben. Die Information, welche Taste gedrückt wurde, soll im event-Objekt als „event.keyCode“ gespeichert werden.

Innerhalb der Funktion gibt man mit „document.write“ die Eigenschaft keyCode des event-Objektes aus, nämlich eine Zahl.
Diese Zahl repräsentiert die gedrückte Taste. Jede Taste auf der Tastatur hat einen eigenen eindeutigen Tastencode.

- Leertaste: 32
- Pfeil nach rechts: 39

Ausgabe im Browser:



Eine weitere Tasteneingabe kann nur erfolgen, wenn vorher auf das „zurück“-Symbol geklickt wurde.

Sinn: Ein Ziel für diese Steuerung könnte die Animation eines Balls sein, den man mit den vier Pfeilen auf dem Bildschirm hin- und herschiebt.

3)Diverses

Das window-Object

Eigenschaft / Methode	Erläuterung	Beispiel
window.prompt() bzw. prompt()	erzeugt ein Eingabe-Fenster, der Wert kann in einer Variablen gespeichert werden.	eingabe=prompt("Bitte Namen eintragen","")
window.alert() bzw. alert()	Erzeugt ein Ausgabe-Fenster, in die runden Klammern, können Zeichfolgen bzw. Variablen eingetragen werden.	alert("Prima,"+eingabe+",dass Sie da sind!")
window.confirm() bzw. confirm()	Erzeugt ein Bestätigungsfenster, je nach Klick (Ok bzw. Abbrechen) wird der boolesche Wert true oder false zurückgegeben. Dieser kann in einer Variablen aufgefangen werden.	antwort=confirm("WollenSie wirklich beenden?")
window.open()	Öffnet ein neues Browserfenster. more...	window.open("seite1.htm")
window.close()	Schließt ein Browserfenster. (Beispiel mit Zeitfunktion)	win.setTimeout("window.close()",4000);

Sonderzeichen in Strings

Folgende Steuerzeichen können verwendet werden. Diese sind bei der Ausgabe nicht sichtbar, bewirken aber eine neue Zeile oder einen Tabulator, usw.

Sonderzeichen	
\b	**backspace** Nach links ein Zeichen löschen.
\n	**new line** Neue Zeile. In Textdokumenten tritt danach meistens noch \r am Zeilenende auf.
\r	**carriage return** Wagenrücklauf
\f	**form feed** Eine Seite weiter.
\t	**tab** Einrückung
\"	**quote** Anführungszeichen. Wird benötigt wenn strings in strings eingebettet werden. Z.B.: Link oder Link
\'	**single quote** Einfaches Anführungszeichen
\\	**backslash** Um einen \ auszugeben muss z.B. document.write("c:\\pfad\\datei.txt") geschrieben werden. Also \ muss doppelt verwendet werden, da ein einfacher Backslash als Beginn eines Steuerzeichens interpretiert wird.

Beispiel: Kleinstrechner

<html><head>
<title>JavaScript</title>
</head>
<body bgcolor=#ffffff>
 kleiner Rechner:
 	
<script>
<!--
var text;
var a=1.6;
var b=5.678;
var p;
 	
p=a*b;
text=a+" * "+b+" ergibt: "+p;
 	
document.write(text);
 	
//-->
</script>
</body></html>
	Berechnung wird durchgeführt.
	Der Variablen text wird ein zusammengesetzter string aus Texten und Variable zugewiesen.
	
	Die Variable wird ausgegeben.

Fenster schließen:

erfolgt mit dem Befehl window.close(). Diese Funktion kann auch in einen Link eingebunden werden: Fenster schließen

<script>

var antwort = confirm("Soll das Fenster geschlossen werden?")

if (antwort == true)

window.close()

</script>

Fenster öffnen:

erfolgt mit dem Befehl Variable = window.open(„“, „Titel d. Fensters“, „width = Zahl, height = Zahl, scrollbars, menubar, resizable); Für Ausgaben in diesem Fenster muss nun zuerst die Variable angegeben werden und danach die Funktion: Variable.document.write(„Text“);

<script>

var fenster;

fenster = window.open("", "Fenster1", "width = 250, height = 100")

```
fenster.document.write("Ausgabe im neuen Fenster <br>")
</script>
```

Timer:

Mit der Funktion setTimeout(Aktion, Verzögerung in Millisekunden) können Aktionen in bestimmten Zeitabständen ablaufen. z.B. Diashow:

```
<script>
if (document.images)
{
	var bilder = new Array("03.jpg", "10.jpg", "11.jpg", "15.jpg");
	var naechstesBild = 0;
	var verzoegerung = 2000;  // Millisekunden
}
	function animation()
	{
		document.images[0].src = bilder[naechstesBild];
		naechstesBild ++;
		if (naechstesBild == bilder.length)
		{
			naechstesBild = 0;
		}
		window.setTimeout("animation()", verzoegerung);
	}
</script>
```

Bilder anzeigen:

Bilder können mit document.showimg.src = Variable.src angezeigt werden. Eine Image-Variable definiert man mit Variable = New Image().

```
<script>
NumberImg = new Array();
for (var i = 0; i<10; i++)
{
	NumberImg[i] = New Image();
	NumberImg[i] = "bild" + i + ".jpg";
}
function show_image(idx)
{
```

```
	document.showimg.src = NumberImg(idx).src;
}
function hide_image()
{
	document.showimg.src = NumberImg[0].src;
}
</script>
```

Datum/Zeit:

Mit new Date() wird das komplette Datum samt Uhrzeit ausgegeben, für die einzelnen Elemente gibt es folgende Funktionen:

```
<script>
var Jetzt = new Date();
var Tag = Jetzt.getDate();
var Monat = Jetzt.getMonth()+1;		// +1, weil Zählung der Monate beginnt mit 0
var Jahr = Jetzt.getYear();
var Stunden = Jetzt.getHours();
var Minuten = Jetzt.getMinutes();
var Sekunden = Jetzt.getSeconds();
</script>
```

Achtung: Firefox-Browser beginnt Zählung der Jahre bei 1900, daher Prüfung, ob Jahr<1000, wenn ja, dann +1900 !!!

Formularfeldprüfung manuell:

Die Funktion zur Prüfung wird im onSubmit-Ereignis aufgerufen. Hierbei wird geprüft, ob document.FormularID.Feld.Value einen Wert übergeben hat oder nicht. Der Cursor kann mit document.FormularID.Feld.focus() im betreffenden Feld platziert werden.

```
<script>
function pruefung()
{
	if (document.MeinFormular.telefon.value == "")
	{
		alert("Das Feld Telefon ist nicht ausgefüllt!");
		document.MeinFormular.telefon.focus();
		return false;
	}
}
</script>
<form onSubmit = "return pruefung()" action="" method="post" name="MeinFormular"
id="MeinFormular">
```