

# Event-Handler, Browserkompatibilität und Standardaktionen

Ab JavaScript 1.2 (und damit in den Browsern Netscape 4.x und 6.x sowie Internet Explorer 4, 5, 5.5 und 6.0) lassen sich Event-Handler nicht nur in einzelnen Tags einsetzen, sondern auch auf ein gesamtes Dokument oder bestimmte Teilbereiche hin definieren. Neben den Inkompatibilitäten zwischen den verschiedenen Browsern stellt dabei insbesondere die Existenz von Standardaktionen des Browsers eine Rolle, die sich nur dann abschalten lassen, wenn browserspezifisch der korrekte Event-Handler mit dem korrekten Unterdrückungsaufwurf gekoppelt wird. In diesem Abschnitt geht es um eine Systematisierung der Event-Handler im Hinblick auf die Standardaktionen und daraus abgeleitet um die Lösung einiger häufig auftretender Probleme.

## Rechter Mausklick ohne Kontextmenü

Wer in eine Webseite mit der rechten Maustaste klickt, erhält im Browser ein Kontextmenü, das beim Klick auf ein Bild insbesondere die Möglichkeit bietet, dieses abzuspeichern. Immer wieder taucht dabei die Überlegung auf, mit einem JavaScript dieses Kontextmenü als Standardaktion des Browsers abzuschalten, um ‚Bilderklau‘ zu unterbinden. Sehen wir an dieser Stelle davon ab, daß der professionelle Surfer selbstverständlich Mittel und Wege kennt, trotzdem ein Bild abzuspeichern (die entsprechende Datei steht im Cache und wenn alles nichts hilft, läßt sich ein Bild immer noch als Screenshot ‚abkupfern‘)<sup>1</sup>, dann stellt sich die Abschaltung des Kontextmenüs für die verschiedenen Browser (Internet Explorer 4, 5 und 5.5 und Netscape 4.x und 6.x werden hier behandelt<sup>2</sup>) recht unterschiedlich dar:

Für Netscape 4.x sind verschiedene Lösungen publiziert, die konsequent das Kontextmenü unterdrücken. Für den Internet Explorer 4-6.0 findet sich in Newsgroups und Foren eine immer wiederkehrende Lösung, die mit einem Alert-Fenster arbeitet. Dies könnte den Anschein erwecken, daß ohne Alert-Fenster eine Unterdrückung des Kontextmenüs nicht möglich ist. Netscape 6.x benötigt eine spezifische Umsetzung von ECMA-Script. Mit einer systematischen Herangehensweise sind jedoch diese offenen Fragen problemlos zu lösen.

## Den rechten Mausklick abfangen

Der erste Schritt zu einer Lösung besteht in einer genauen Beobachtung, wie ein Browser auf einen rechten Mausklick reagiert. Nicht von ungefähr gibt es drei verschiedene, für unser Thema wichtige Event-Handler für die Maus:

- **onmousedown**: Beim Herunterdrücken der Maustaste
- **onmouseup**: Beim Loslassen der Maustaste.

---

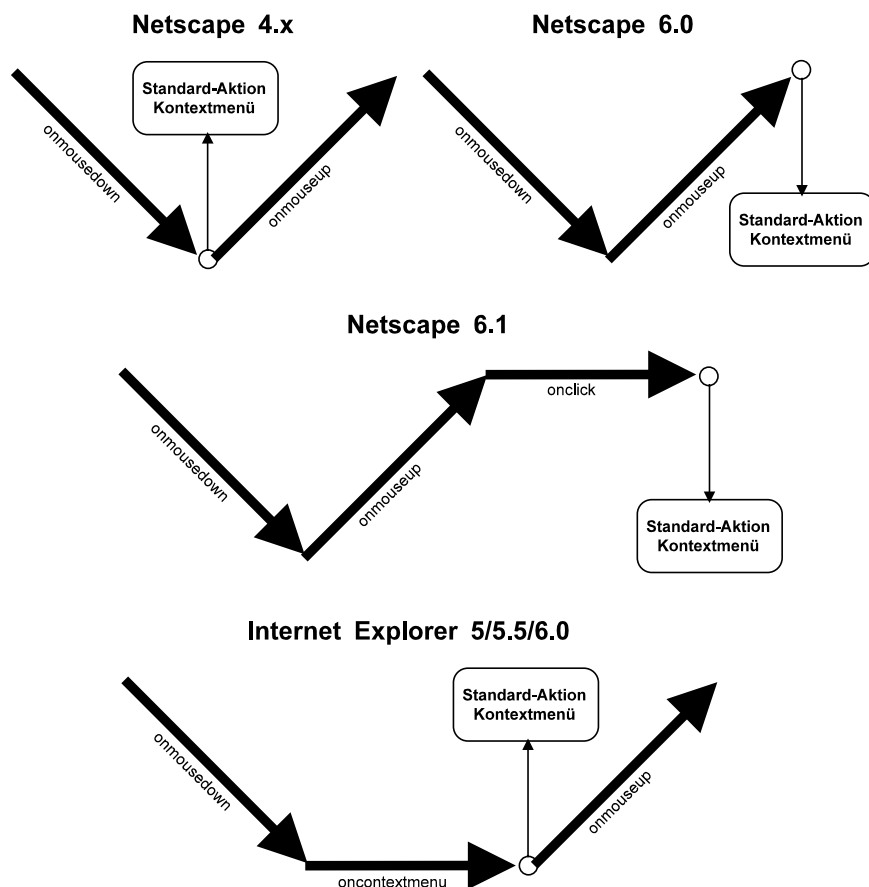
<sup>1</sup> Unter Windows XP/Internet Explorer 6.0 kommt hinzu, daß neben dem Kontextmenü bereits beim Überfahren eines Bildes auch eine kleine Schaltflächenleiste eingeblendet wird, die ein Abspeichern, Drucken oder Versenden des Bildes ermöglicht.

<sup>2</sup> Da Netscape 6.0 kaum Verbreitung gefunden hat, und mittlerweile fast völlig durch Netscape 6.1 ersetzt wurde, wird im Text zwar auf Netscape 6.0 eingegangen, die Listings sind jedoch nur für Netscape 6.1 geschrieben.

Der mittlerweile von T-Online vertriebene T-Online-Browser setzt intern auf den Internet Explorer 5.x/6.0 auf und kann im Hinblick auf JavaScript wie dieser behandelt werden.

- **onclick:** Nach dem Klicken der Maus (mousedown + mouseup)

Im Internet Explorer 5.x/6.0 und in Netscape 4.x ergibt eine experimentelle Nachprüfung, daß das Herunterdrücken der rechten Maustaste (ohne sie losgelassen zu haben!) bereits das Kontextmenü aufruft. Anders beim Internet Explorer 4.x und Netscape 6.x: Hier wird das Kontextmenü erst dann dargestellt, wenn die Maustaste losgelassen wird. Soll das Kontextmenü als Standardaktion also unterdrückt werden, muß bei Netscape 4.x beim Event-Handler *onmousedown* angesetzt werden, beim Internet Explorer 4.x und bei Netscape 6.x wäre zunächst zu erwarten, daß *onmouseup* für das Kontextmenü zuständig ist. Leider ist die Welt von JavaScript nicht ganz so einfach gestrickt, wie die nachfolgende Grafik verdeutlicht:



### Netscape 4.x: onmousedown und return false

Netscape 4.x erfordert für eine allgemeine Abfrage eines Events zunächst eine Anmeldung des Events. Da das Kontextmenü als Standardaktion bei *onmousedown* ausgelöst wird, gilt es also, das Herunterdrücken der Maustaste anzumelden und abzufangen:

```
document.captureEvents(Event.MOUSEDOWN)
document.onmousedown=keineanzeige
```

Die erste Zeile meldet das Herunterdrücken einer Maustaste als abzufangenden Event an, die zweite Zeile weist diesem Ereignis den Namen einer Funktion zu, die stets ausgeführt werden soll, wenn der Event eintritt.

Die zugewiesene Funktion enthält dann die Ermittlung, welche Maustaste geklickt wurde und was geschehen soll, wenn es sich um die rechte Maustaste handelt. Die Abfrage der Maustaste ist bei den verschiedenen Browsern unterschiedlich. Hier eine kurze Übersicht:

Abfrage der Maustasten		
Browser	Abfrage	Rückgabe
NS 4.x	e.which	1 => links 2 => mitte 3 => rechts
NS 6.0	e.button	1 => links 2 => mitte 3 => rechts
NS 6.1	e.button	0 => links 1 => mitte 2 => rechts
IE 4/5.x/6	event.button	1 => links 2 => rechts 4 => mitte

Für Netscape 4.x ergibt sich daraus die Möglichkeit, das Kontextmenü zu unterdrücken, durch folgende Funktion:

```
function keineanzeige(e)
{
if(e.which==3)
return false
}

```

Wenn *e.which* den Wert 3 hat, wurde die rechte Maustaste geklickt. Das nachfolgende *return false* unterdrückt die zugehörige Standardaktion (hier das Aufpoppen des Kontextmenüs).

Netscape 6.x: mouseup und die Methode preventDefault()

Netscape **6.0** hält sich insofern an den ECMA-Standard für JavaScript, als nach ECMA sowohl die Zuweisung einer Funktion zu einem Event wie auch die Unterdrückung der Standardaktion anders aussieht als in JavaScript (Netscape 4.x) und JScript (Internet Explorer). Hinzu kommt, daß die Abfrage der Maustaste nicht mehr wie in Netscape 4.x über *e.which*, sondern über *e.button* läuft und die Tatsache, daß das Kontextmenü erst beim Loslassen der Maustaste (*onmouseup*) aufpoppt. Die Zuweisung einer Funktion für das Loslassen der Maustaste und die entsprechende Funktion sehen deshalb so aus:

```
document.addEventListener("mouseup",keineanzeige,false)
function keineanzeige(e)
{
if(e.button==3)
e.preventDefault()
}

```

Netscape **6.1** wartet mit zwei Änderungen verglichen mit Netscape 6.0 auf, die ein kompatibles Programmieren reichlich erschweren:

- Die Maustasten werden nicht mehr (links, mitte, rechts) mit 1, 2 bzw. 3 zurückgemeldet, sondern mit 0, 1 bzw. 2.
- Nicht mehr *onmouseup* sondern *onclick* ist nunmehr der auslösende Event-Handler.

Das entsprechende Listing sieht deshalb so aus:

```
document.addEventListener("click",keineanzeige,false)
function keineanzeige(e)
{
if(e.button==2)
e.preventDefault()
}

```

## Internet Explorer: Zweierlei Einbindungen

Beim Internet Explorer scheitert der Versuch, nach dem Abfangen des Events *onmousedown* diesen mit *return false* zu unterbinden. Das Kontextmenü poppt dennoch auf. In der Literatur hat sich deshalb als Lösung etabliert, hier mit einem Alert-Fenster zu arbeiten, um das Kontextmenü zu unterbinden:

```
document.onmousedown=keineanzeige
function keineanzeige()
{
  if(event.button==2)
  {
    alert("kein Kontextmenü")
    return false
  }
}
```

Obwohl häufig zu finden, ist hierbei das abschließende *return false* völlig überflüssig. Das Kontextmenü wird einzig deshalb unterbunden, weil das Alert-Fenster sein Aufpoppen verhindert.

Wenn diese Lösung nicht genügt, muß zwischen Internet Explorer 4.x und 5.x/6.0 unterschieden werden: Der Internet Explorer 4.x hat das Kontextmenü nicht als Standardaktion des Browsers eingebunden, sondern als Standardaktion des Betriebssystems, die sich über die Browser-Event-Handler nicht abschalten läßt. (Einzigste Ausnahme bildet die mit dem Internet Explorer 4.x eingeführte Scriptlet-Technik.) Die beschriebene Lösung mittels eines aufpoppenden Alert-Fensters funktioniert also einfach deshalb, weil im Betriebssystem festgelegt ist, daß ein Alert-Fenster das Kontextmenü unterdrückt. Eine – wenn auch nicht wirklich befriedigende – Alternative besteht darin, kurzfristig ein neues Fenster zu öffnen und mit kurzer Verzögerung wieder zu schließen. Auch dies unterdrückt das Kontextmenü, ohne daß der Benutzer erst ein Alert-Fenster mit OK schließen muß.

## Internet Explorer 5.x und 6.0: oncontextmenu

Im Internet Explorer 5.x und 6.0 läßt sich das Kontextmenü unterdrücken. Offenkundig ist aber der Event-Handler *onmousedown* nicht für das Aufpoppen des Kontextmenüs zuständig und kann es folglich auch nicht mit *return false* abschalten. Die Frage, welcher Event-Handler für das Kontextmenü zuständig ist, führt zu der Überlegung, daß der Internet Explorer weitere Event-Handler kennt, die in JavaScript nicht bekannt sind, sondern einzig im Internet Explorer (Jscript) zur Verfügung stehen.

Zur Erkundung der weiteren Event-Handler des Internet Explorers hilft die Abfrage über *for(i in x)* weiter. *x* steht hierbei zunächst für ein Array, dessen einzelne Elemente mit der Zählervariablen *i* nacheinander durchlaufen werden. Da auch Objekte mit ihren Eigenschaften in JavaScript intern als Arrays abgespeichert werden, ist es damit möglich, für ein Objekt nacheinander dessen Eigenschaften abzufragen. Listing 1 liefert im Internet Explorer eine Darstellung aller Eigenschaften und auch Event-Handler, die für den <body>-Tag zur Verfügung stehen.



## Eine kompatible Lösung

Um für Netscape 4.x, Netscape 6.1 und Internet Explorer 5.x/6.0 jeweils den korrekten Code für die Unterdrückung des Kontextmenüs zur Verfügung zu stellen, muß abschließend vom JavaScript ermittelt werden, um welchen Browser es sich handelt.

Hierbei hat sich eine Lösung etabliert, die mit nachfolgendem Schema arbeitet:

```
if(document.layers) {Netscape 4.x-Code}
if(document.all)
{Internet Explorer-Code}
else
if(document.getElementById) {Netscape 6.1-Code}
```

Beispiel 2 zeigt das komplette JavaScript zur Unterdrückung des Kontextmenüs in allen benannten Browsern.



### Beispiel 2

#### Unterdrückung des Kontextmenüs – kompatibel für Netscape 4.x und 6.1 sowie Internet Explorer 5.x/6

```
18. <script language="JavaScript1.2">
19. <!--
20. // Kontextmenü beim rechten Mausklick verhindern
21. // Für IE 5,5.5,6.0 und NS 4.x sowie NS 6.1
22. // Copyright: rws - rainer werle software, 2001
23. // rws@werle.com
24.
25. if (document.layers)
26. {
27. document.captureEvents(Event.MOUSEDOWN)
28. document.onmousedown=keineanzeige;
29. }
30. if (document.all)
31. document.oncontextmenu=keineanzeige;
32. else
33. if (document.getElementById)
34. document.addEventListener("click",keineanzeige,false);
35.
36. function keineanzeige(e)
37. {
38. if (document.layers && e.which==3)
39. return false;
40. if (document.all)
41. return false;
42. else
43. if (document.getElementById && e.button==2)
44. e.preventDefault();
45. }
46. //-->
47. </script>
```

## Tastatureingaben als Events

Noch etwas komplexer stellt sich die Situation bei der Abfrage von Tastatureingaben in JavaScript dar. Es gibt hierfür drei Event-Handler, die in jedem der Browser unterschiedlich implementiert sind.

- **onkeydown**: Beim Herunterdrücken der Taste
- **onkeypress**: Wenn die Taste heruntergedrückt ist
- **onkeyup**: beim Loslassen der Taste

Hinzu kommt, daß diese Event-Handler für die einzelnen Tasten der Tastatur deren ASCII-Wert als Dezimalzahl zurückliefern, für eine Ausgabe der zugehörigen Buchstaben jedoch eine hexadezimale Angabe notwendig ist.

### Problemfall Back-Taste

Soll dem Benutzer die Möglichkeit geboten werden, in eine Webseite eigenen Text einzugeben, so wäre es natürlich wünschenswert, daß Tippfehler mit der Back-Taste korrigiert werden können. Bei Netscape 6.x und dem Internet Explorer steht dem jedoch die Standard-Aktion der Back-Taste im Weg: Die Back-Taste funktioniert wie die Zurück-Schaltfläche des Browsers und blättert eine Seite in der Browser-History zurück. Diese Standard-Aktion muß also ausgeschaltet werden, sollen Benutzereingaben einen Sinn machen.

Doch damit nicht genug, ergeben sich erstaunliche Browser-Inkompatibilitäten, wobei Netscape 6.x sich besonders beharrlich einer vernünftigen Handhabung entzieht. Die Tabelle über die Rückgabewerte zeigt in ersten Ansätzen die Problematik auf:

Rückgabewerte der Tastatur-Event-Handler in Netscape und Internet Explorer										
	Taste	a	A	ü	Ü	Ö	Back*	Return	Shift	AltGr
	ASCII-Wert	97	65	252	220	214				
	HTML-Code			&#252;	&#220;	&#214;				
Netscape 4.x	onkeydown	97	65	252	220	214	8	13	0	0 0**
	onkeypress	<u>97</u>	<u>65</u>	<u>252</u>	<u>220</u>	<u>214</u>	<u>8</u>	<u>13</u>	<u>0</u>	0 0**
	onkeyup	97	65	252	220	214	8	13	0	0
Netscape 6.x	onkeydown	<u>65</u>	<u>65</u>	<u>59</u>	<u>59</u>	<u>192</u>	<u>8</u>	<u>13</u>	<u>16</u>	17 18**
	onkeypress	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>8</u>	<u>13</u>	<u>0</u>	-
	onkeyup	65	65	59	59	192	8	13	16	17 18**
IE 4/5/5.5	onkeydown	<u>65</u>	<u>65</u>	<u>186</u>	<u>186</u>	<u>192</u>	<u>8</u>	<u>13</u>	<u>16</u>	17 18**
	onkeypress	<u>97</u>	<u>65</u>	<u>252</u>	<u>220</u>	<u>214</u>	<u>8</u>	<u>13</u>	-	-
	onkeyup	65	65	186	186	192	8	13	16	17 18**

\* Rückgabewert teilweise erst nach Ausschalten der Standardaktion

\*\* Der Event-Handler feuert zweimal, gibt also nacheinander zwei Werte zurück!

\_\_ Unterstrichen sind Rückgabewerte, die repetitiv arbeiten, das heißt, daß der Event-Handler solange die Taste gedrückt gehalten wird, immer wieder feuert.

### Rückgabewerte: Netscape 4.x

Sehr gutmütig gebärdet sich Netscape 4.x, denn alle drei Event-Handler (*onkeydown*, *onkeypress*, *onkeyup*) geben als Rückgabewert den ASCII-Wert der gedrückten Taste in Dezimaldarstellung zurück. Einige Punkte sind dennoch erwähnenswert:

1. *onkeypress* arbeitet repetitiv: Solange die Taste gedrückt ist, feuert der Event-Handler immer wieder. Das ist im Normalfall erwünscht (auch Textverarbeitungsprogramme arbeiten in dieser Weise), sollte aber berücksichtigt werden, wenn eine repetitive Eingabe nicht sinnvoll ist.
2. Das Drücken der Shift-Taste liefert die Rückgabe 0, wobei diese Rückgabe bei *onkeypress* solange anhält, bis eine andere Taste zusätzlich gedrückt wird. Es ist für Texteingaben in Webseiten deshalb sinnvoll, eine Abfrage auf 0 einzufügen, die keine Ausgabe liefert.
3. Gleiches gilt für das Drücken der AltGr-Taste, die zwar nichtrepetitiv arbeitet, dafür aber (wie auch in den anderen Browsern) doppelt feuert: Ein Klick auf AltGr liefert in Netscape 4.x zweimal nacheinander den Wert 0 zurück.

### Rückgabewerte: Internet Explorer

Der Internet Explorer liefert für *onkeydown* und *onkeyup* die Tastennummer zurück, für *onkeypress* jedoch den dezimalen ASCII-Wert der gedrückten Taste. Es ist deshalb sinnvoll, die normale Tastatureingabe über *onkeypress* abzufragen. Dabei liefern auch Tastenkombinationen wie *AltGr + q* (entspricht @) korrekte Rückgabewerte. Weiterhin gilt es zu beachten:

1. *onkeydown* und *onkeypress* arbeiten repetitiv.
2. Auch hier feuert die AltGr-Taste zweimal und liefert nacheinander die Werte 17 und 18 zurück.
3. Die Shift-Taste liefert bei *onkeypress* keinen Rückgabewert: Der Event-Handler feuert nicht.

### Rückgabewerte: Netscape 6.x

Während der Internet Explorer bei *onkeypress* die korrekten ASCII-Werte für jede Tastaturtaste liefert, meldet Netscape 6.x hier jeweils den Wert 0 zurück. Bei Netscape 6.x muß deshalb die Abfrage der gedrückten Taste über *onkeydown* erfolgen.

Die Angaben in der Tabelle über die Rückgabewerte zeigt auch, daß dabei mehrere Probleme auftreten:

1. Für normale Buchstaben (keine Umlaute, kein ‚ß‘) wird der korrekte ASCII-Wert des Großbuchstabens zurückgegeben. Eine gleichzeitige Abfrage, ob die Shift-Taste gedrückt ist, ermöglicht es jedoch, mit *toLowerCase()* auch den richtigen Kleinbuchstaben zu erzeugen.
2. Schwieriger wird es bei Umlauten, Satzzeichen etc. Während Eingaben in ein Textfeld einer Webseite von Netscape 6.x korrekt interpretiert werden, erkennen die Event-Handler nicht die andere (z.B. deutsche) Tastaturbelegung.
3. Das Drücken der AltGr-Taste liefert zwar den Doppel-Event-Handler mit den Rückgabewerten 17 und 18, führt aber ebenfalls nicht zu einer geänderten Rückgabe der zugleich gedrückten Tasten (etwa für @).

Für die letzten beiden Punkte ergibt sich als Behelfslösung die Möglichkeit, über vier Arrays eine Umrechnung vorzunehmen. Hier für die üblichen Tasten die ASCII-Werte:

ASCII-Umwandlung für Netscape 6.x	
Eingabe	^ 1 2 3 4 5 6 7 8 9 0 ß ' q e ü + ö ä # < m , . -
Ausgabe	220 49 50 51 52 53 54 55 56 57 48 219 221 81 69 59 61 192 222 191 226 77 188 190 109
+ Shift	176 33 34 167 36 37 38 47 40 41 61 63 96 81 69 220 42 214 196 39 62 77 59 58 95
+ AltGr	94 49 178 179 52 53 54 123 91 93 125 92 180 64 128 252 126 246 228 35 124 181 44 46 45

```
eingabe=new
```

```
Array(220,49,50,51,52,53,54,55,56,57,48,219,221,81,69,59,61,192,222,191,226,77,188,190,109)
```

```

ausgabe=new
Array(94,49,50,51,52,53,54,55,56,57,48,223,180,113,101,252,43,246,228,35,60,109,44,46,45)
shift=new
Array(176,33,34,167,36,37,38,47,40,41,61,63,96,81,69,220,42,214,196,39,62,77,59,58,95)
altgr=new
Array(94,49,178,179,52,53,54,123,91,93,125,92,180,64,128,252,126,246,228,35,124,181,44,46,45)

```

Schließlich gilt es für Texteingaben bei Netscape 6.x noch einen Punkt zu berücksichtigen: Nach dem Laden der Seite ist der Focus auf dem Adress/URL-Eingabefeld des Browsers. Damit der Benutzer sofort eine Eingabe tätigen kann, muß im <body>-Tag mit

```
<body onLoad="window.focus()">
```

der Focus auf das Fenster gesetzt werden.

### Standard-Aktion unterdrücken

Abschließend gilt es noch für die Back-Taste die Standard-Aktion (Zurückblättern) für den Internet Explorer und für Netscape 6.x zu unterdrücken. Auch hier unterscheiden sich die beiden Browser wieder grundlegend: Während der Internet Explorer die Standard-Aktion bei Tastatureingaben bereits bei *onkeydown* startet, findet sie in Netscape erst bei *onkeypress* statt, das in diesem Fall den korrekten Rückgabewert 8 liefert. Dementsprechend muß das Ausschalten der Standard-Aktion mit *return false* bzw. *preventDefault()* bei den jeweiligen Event-Handlern ansetzen.



Tastatureingabe in eine Webseite (hier beim Netscape 6.0)



### Beispiel 3

Tastatureingabe in eine Webseite – kompatibel für Netscape 4.x/6.x und Internet Explorer 4.x/5.x/6.x

```

48. <html>
49. <head>
50. <meta http-equiv="Content-Type" content="text/html; charset=iso-
    8859-1">
51. <meta http-equiv="Content-Language" content="de">
52. <title>Tastatureingabe und Back-Taste abschalten</title>

```

```

53. <script language="JavaScript1.2">
54. <!--
55. // Abschalten der Standard-Aktion für die Back-Taste
56. // Und Textausgabe im Browserfenster
57. // Für IE 4,5,5.5,6.0 und NS 4.x sowie NS 6.x
58. // Copyright: rws - rainer werle software, 2001
59. // rws@werle.com
60.
61. if (document.layers)
62. {
63. document.captureEvents(Event.KEYPRESS)
64. document.onkeypress=darstellen;
65. }
66. if (document.all)
67. {
68. document.onkeydown=sondertaste;
69. document.onkeypress=darstellen;
70. }
71. else
72. if (document.getElementById)
73. {
74. document.addEventListener("keydown",darstellen,false);
75. document.addEventListener("keypress",sondertaste,false);
76. }
77.
78. function hexwand(x)
79. {
80. x="%"+parseInt(x).toString(16)
81. return unescape(x)
82. }
83.
84. var taste="";
85. function sondertaste(e)
86. {
87. if(document.all)
88. {
89. if(parseInt(event.keyCode)==8)
90. {
91. if(taste!="")
92. {
93. if(taste.length>3 && taste.substring(taste.length-
94. 4,taste.length)=="<br>")
95. taste=taste.substring(0,taste.length-4)
96. else
97. taste=taste.substring(0,taste.length-1)
98. document.all.textfeld.innerHTML="<p>" + taste.fontSize(5) + "</p>"
99. return false;
100. }
101. }
102. else
103. {
104. if (document.getElementById)
105. if(parseInt(e.keyCode)==8) e.preventDefault();
106. }
107. }
108.
109. var altkeytest=false;
110. function darstellen(e)
111. {
112. //Netscape 4.x*****

```

```

113.  if (document.layers)
114.  {
115.  if(parseInt(e.which)==8)
116.  {
117.  if(taste!="")
118.  {
119.  if(taste.length>3 && taste.substring(taste.length-
120.  4,taste.length)=="<br>")
121.  taste=taste.substring(0,taste.length-4)
122.  else
123.  taste=taste.substring(0,taste.length-1)
124.  }
125.  }
126.  else
127.  {
128.  if(parseInt(e.which)==13)
129.  taste=taste + "<br>"
130.  else
131.  taste=taste + hexwand(e.which)
132.  }
133.  document.textfeld.document.open()
134.  document.textfeld.document.write("<p>" + taste.fontSize(5) + "</p>")
135.  document.textfeld.document.close()
136.  }
137.  //Internet Explorer 4/5/5.5/6.0*****
138.  if (document.all)
139.  {
140.  if(parseInt(event.keyCode)==13)
141.  taste=taste + "<br>"
142.  else
143.  taste=taste + hexwand(event.keyCode)
144.  document.all.textfeld.innerHTML="<p>" + taste.fontSize(5) + "</p>"
145.  }
146.  else
147.  //Netscape 6.x*****
148.  if (document.getElementById)
149.  {
150.  if(parseInt(e.keyCode)==17) altkeytest=false;
151.  if(parseInt(e.keyCode)==18) altkeytest=true;
152.  if(parseInt(e.keyCode)==8)
153.  {
154.  if(taste!="")
155.  {
156.  if(taste.length>3 && taste.substring(taste.length-
157.  4,taste.length)=="<br>")
158.  taste=taste.substring(0,taste.length-4)
159.  else
160.  taste=taste.substring(0,taste.length-1)
161.  }
162.  }
163.  else
164.  {
165.  if(parseInt(e.keyCode)==13)
166.  {
167.  taste=taste + "<br>"
168.  }
169.  }
170.  if(e.keyCode>31)
171.  {

```

```
172. for(i=0;i<=eingabe.length-1;i++)
173. {
174.   if(e.keyCode==eingabe[i])
175.   {
176.     var help=ausgabe[i]
177.     if(e.shiftKey) help=shift[i]
178.     if(altkeytest) {help=altgr[i]; altkeytest=false}
179.     break
180.   }
181.   else
182.     help=e.keyCode
183.   }
184.   if(e.shiftKey==false)
185.     taste=taste + (hexwand(help).toLowerCase())
186.   else
187.     taste=taste + hexwand(help)
188.   }
189. }
190. }
191. document.getElementById("textfeld").innerHTML="<p>" +
  taste.fontSize(5) + "</p>"
192. }
193. }
194. eingabe=new
195. Array(220,49,50,51,52,53,54,55,56,57,48,219,221,81,69,59,61,192,222,
  191,226,77,188,190,109)
196. ausgabe=new
197.
  Array(94,49,50,51,52,53,54,55,56,57,48,223,180,113,101,252,43,246,22
  8,35,60,109,44,46,45)
198. shift=new
  Array(176,33,34,167,36,37,38,47,40,41,61,63,96,81,69,220,42,214,196,
  39,62,77,59,58,95)
199. altgr=new
200.
  Array(94,49,178,179,52,53,54,123,91,93,125,92,180,64,128,252,126,246
  ,228,35,124,181,44,46,45)
201. //-->
202. </script>
203. </head>
204. <body onLoad="window.focus()">
205. <p align="center"><b><font size="6">Hier erscheint die
  Tastatureingabe:</font></b></p>
206. <div id="textfeld" style="position:absolute; top:100; left:200">
207. <p>&nbsp;</p>
208. </div>
209. </body>
210. </html>
```