```
003B4 "Gauss" WM_NCMOUSEMOVE (a004X) Verarbeitet wp=00000014 lp=006803B7 (951,104) in HT_0014h
003B4 "Gauss" WM_NCHITTEST (8404X) Gesendet wp=00000000 lp=006703B8 (952,103)
003B4 "Gauss" WM_SETCURSOR (2004X) Gesendet wp=000003B4 lp=02000014 MouseMove in HT_0014h hwnd 000003B4
003B4 "Gauss" WM_NCMOUSEMOVE (a004X) Verarbeitet wp=00000014 lp=006703B8 (952,103) in HT_0014h
003B4 "Gauss" WM_NCHITTEST (8404X) Gesendet wp=00000000 lp=006603B9 (953,102)
003B4 "Gauss" WM_SETCURSOR (2004X) Gesendet wp=000003B4 lp=02000014 MouseMove in HT_0014h hwnd 000003B4
003B4 "Gauss" WM_NCMOUSEMOVE (a004X) Verarbeitet wp=00000014 lp=006603B9 (953,102) in HT_0014h
003B4 "Gauss" WM_NCHITTEST (8404X) Gesendet wp=00000000 lp=006503B9 (953,101)
003B4 "Gauss" WM_SETCURSOR (2004X) Gesendet wp=000003B4 lp=02000014 MouseMove in HT_0014h hwnd 000003B4
003B4 "Gauss" WM_NCMOUSEMOVE (a004X) Verarbeitet wp=00000014 lp=006503B9 (953,101) in HT_0014h
003D8 {ComboLBox} LB_GETCURSEL (18804X) Gesendet wp=00000000 lp=00000000
003D8 {ComboLBox} LB_GETCURSEL (18804X) Gesendet wp=00000000 lp=00000000
003B4 "Gauss" WM_NCHITTEST (8404X) Gesendet wp=00000000 lp=006503B9 (953,101)
003B4 "Gauss" WM_SETCURSOR (2004X) Gesendet wp=000003B4 lp=02010014 LButtonDown in HT_0014h hwnd 000003B4
003B4 "Gauss" WM_NCLBUTTONDOWN (a104X) Verarbeitet wp=00000014 lp=006503B9 (953,101) in HT_0014h
003B4 "Gauss" WM_LBUTTONUP (20204X) Verarbeitet wp=00000000 lp=FFF20378 (888,65522)
003B4 "Gauss" WM_CAPTURECHANGED (21504X) Gesendet wp=00000000 lp=00000000 Aufzeichnen in 00000000
003B4 "Gauss" WM_SYSCOMMAND (11204X) Gesendet wp=0000F060 lp=006503B9 Close+0(Nowhere?) (953,101)
003B4 "Gauss" WM_CLOSE (1004X) Verarbeitet wp=00000000 lp=00000000
003C0 {ReBarWindow} WM_USER+0x10 (41004X) Gesendet wp=00000004 lp=00000000
003C0 {ReBarWindow} WM_USER+0x1C (41c04X) Gesendet wp=00000003 lp=00BDFABC
003B4 "Gauss" WM_GETHOTKEY (3304X) Gesendet wp=00000000 lp=00000000Leer
003B4 "Gauss" WM_WINDOWPOSCHANGING (4604X) Gesendet wp=00000000 lp=00BDFC7E (0,0)-(0,0) Z-Ordnung Oben
000DC {Shell_TrayW} WM_NCPAINT (8504X) Gesendet wp=00000B3C lp=00000000 wp=00000B3C!
000DC {Shell_TrayW} WM_ERASEBKGND (1404X) Gesendet wp=0000077E lp=00000000 hdc 0000077E
003B4 "Gauss" WM_WINDOWPOSCHANGED (4704X) Gesendet wp=00000000 lp=00BDFCBA (10000,10000)-(10906,10678) NoSize,
003B4 "Gauss" WM_MOVE (304X) Gesendet wp=00000000 lp=27272714 (10004,10023)
003CC {ToolbarWind} TB_BUTTONCOUNT (41804X) Gesendet wp=00000000 lp=00000000
```

Abb. 1: Diese Botschaften wurden von dem Programm WINSIGHT aufgezeichnet.

Warten auf Ereignisse

Wenn wir in einem Windows-Programm eine Schaltfläche anklicken, wenn wir die Maus bewegen oder eine Taste drücken, bekommt das Programm von diesem **Ereignis** (englisch: event) durch das Betriebssystem Windows eine entsprechende **Botschaft** (englisch: message) gesendet. In Abb. 1 sehen wir eine Reihe solcher Botschaften. Einige von Ihnen betreffen das HTML-Dokument `Gauss.htm`: Ein Knopf wird gedrückt (WM_NCLBUTTONDOWN) und wieder losgelassen (WM_LBUTTONUP). Unser HTML-Dokument kümmert sich nicht selbst darum, was gerade mit der Maus oder der Tastatur geschieht; diese Arbeit übernimmt das Betriebssystem Windows. Wenn Windows nämlich ein Ereignis registriert, sendet es eine entsprechende Botschaft an das HTML-Dokument. Dieses kann dann entsprechend reagieren. Z. B. kann durch die Botschaft „Schaltfläche gedrückt“ eine JavaScript-Funktion aufgerufen werden.

Wie diese Arbeitsteilung zwischen Betriebssystem und Anwendungsprogramm im Detail funktioniert, auf welche Ereignisse – neben dem Betätigen einer Schaltfläche – JavaScript reagieren kann und wozu diese Ereignisse sich nutzen lassen können, darüber handelt dieses Kapitel.

Von den Ereignissen zur Ereignisbehandlung

Ein einfaches Beispiel soll helfen, die Geschehnisse bei der Verarbeitung eines Ereignisses zu erklären: In dem Euro-Umrechner-Dokument von Abb. 2 wird der eingegebene DM-Betrag in den entsprechenden Euro-Betrag umgerechnet, wenn die Schaltfläche betätigt wird. Wir gehen davon aus, dass der DM-Wert schon eingetragen worden ist.

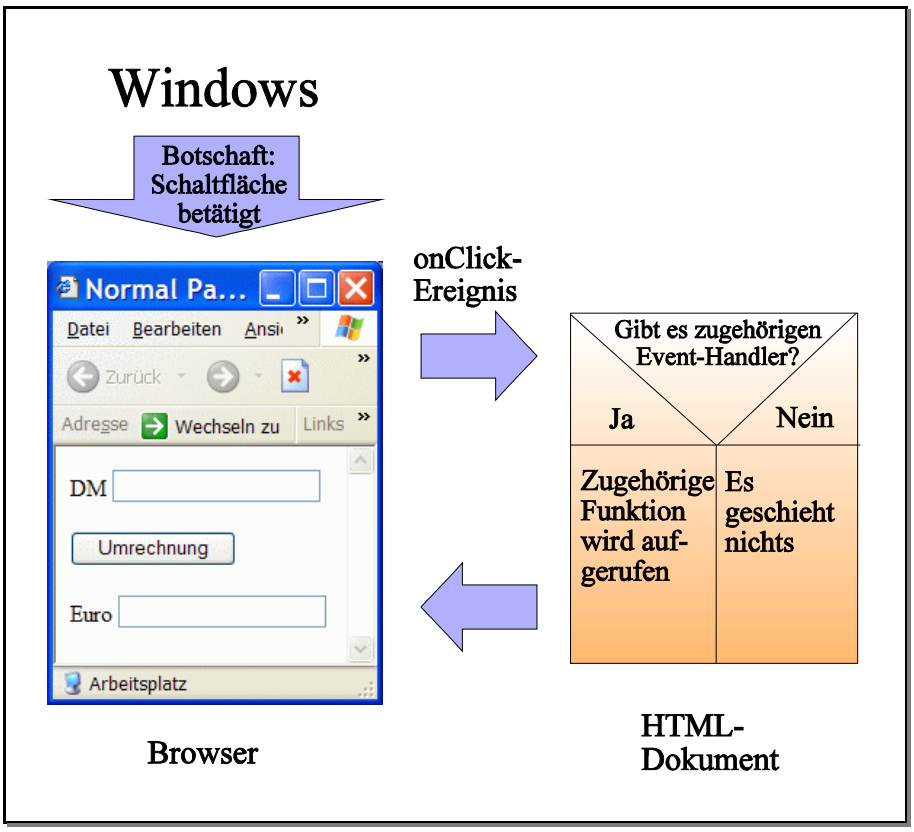

Abb. 2: Wenn eine Schaltfläche angeklickt wird...

Das Betriebssystem Windows prüft permanent nach, ob der Anwender irgendeine Eingabe tätigt, sei es mit der Maus, der Tastatur oder einem anderen Gerät. In dem Augenblick, in dem die Schaltfläche „Umrechnung“ angeklickt wird, informiert Windows den Browser über dieses Ereignis mit einer Botschaft (Abb. 2). Der Browser überprüft nun, ob und wie er darauf reagieren muss. Dazu kontrolliert er nach, ob für diese Schaltfläche eine **Ereignisbehandlung** vorgesehen ist. Statt des deutschen Wortes „Ereignisbehandlung“ benutzt man häufiger den englischen Ausdruck „**Event-Handler**“. Der Event-Handler stellt die Verbindung zwischen Ereignis und der **Ereignisbehandlungsfunktion** her; das ist diejenige JavaScript-Funktion, die durch das Ereignis ausgelöst werden soll. In unserem Fall wird diese Verbindung durch das onClick-Attribut des input-Tags unsere Schaltfläche hergestellt:

```
<input    type = "button"
          value = "Umrechnung"
          onClick = "rechnen()" >
```

Hier wird durch das Anklicken der Umrechnung-Schaltfläche die JavaScript-Funktion `rechnen()` aufgerufen. Wenn in dem input-Tag das onClick-Attribut fehlt, gibt es keinen Event-Handler für das Ereignis „Schalfläche betätigt“; in diesem Fall zeigt der Browser keine Reaktion auf die Windows-Botschaft.

So reagiert unsere Schaltfläche z. B. noch nicht, wenn wir die Maus über sie positionieren. In dem input-Tag wird dazu kein Event-Handler genannt. Wenn man in diesem Fall also eine Reaktion wünscht, muss man eine entsprechende Verbindung zu einer JavaScript-Funktion

herstellen. Der Event-Handler für das `MouseOver`-Ereignis wird durch das `onMouseOver`-Attribut gebildet:

```
<input     type = "button"
           value = "Umrechnung"
           onClick = "rechnen()"
           OnMouseOver = "anzeigen()" >
```

Über die Funktion `anzeigen()` könnte dann in der Statuszeile angezeigt werden, welche Bedeutung die Schaltfläche hat:

```
function anzeigen()
{
      status = "rechnet DM-Betrag in Euro-Betrag um";
}
```

Nicht nur für Schaltflächen, sondern auch für andere Objekte eines HTML-Dokuments gilt:

> Ereignisse sind immer an bestimmte Objekte gebunden. Die Verknüpfung zwischen Ereignis und der zugehörigen Ereignisbehandlungsfunktion erfolgt über einen Event-Handler; dieser wird durch ein Attribut im entsprechenden Objekt-Tag realisiert. Das Attribut hat die Form
>
> `OnEreignis = "Ereignisbehandlungsfunktion()"`
>
> Ein Objekt kann auch mehrere Event-Handler besitzen und so auf verschiedene Ereignisse unterschiedlich reagieren.

Wichtige Ereignisse

Ereignisse werden im HTML-Dokument durch Event-Handler bzw. Objekt-Attribute wie `onClick` oder `onMouseOver` erfasst. Es gibt eine große Anzahl von Ereignissen und entsprechend viele Event-Handler. Wir wollen hier nur die wichtigsten in einer Tabelle vorstellen. Nicht alle Ereignisse machen bei allen Objekten Sinn; deswegen wird in der Tag-Spalte angegeben, mit welchen Objekten die jeweiligen Ereignisse verknüpft werden können.

Event-Handler	**Tag**	**wird aktiviert, wenn...**
onLoad	<BODY>	Browser ein Dokument geladen hat
onUnload	<BODY>	Benutzer ein Dokument schließt oder verlässt
onAbort	<IMG>	der Benutzer das Laden einer Grafik abbricht
onClick	<A href=...> <INPUT type="button" ...>	der Benutzer mit der Maus auf einen Link bzw. eine Schaltfläche klickt

Event-Handler	Tag	wird aktiviert, wenn...
onError	<IMG>	die Grafik nicht geladen werden konnte
onMouseOut	<IMG> <A href=...> <INPUT type="button" ...>	der Mauszeiger das Objekt verlässt
onMouseOver	<IMG> <A href=...> <INPUT type="button" ...>	der Mauszeiger auf das Objekt positioniert wird
onBlur	<INPUT type = "text" ...>	der Benutzer das Textfeld verlässt
onChange	<INPUT type = "text" ...>	der Benutzer das Textfeld verlässt und eine Änderung in dem Feld vorgenommen wurde
onFocus	<INPUT type = "text" ...>	der Benutzer das Feld aktiviert, z.B. durch Anklicken oder Tabulatorsprung

Aufgaben

1. Wenn die Grafik nicht geladen werden konnte (etwa weil sie nicht auf dem Server vorhanden ist), soll eine Meldung über den Inhalt des Bildes informieren.

2. Unmittelbar nach dem Laden des Dokuments im Browser soll ein Meldung zur Begrüßung erfolgen. Wie muss das entsprechende HTML-Dokument aussehen?

3. Beim Verlassen einer Webseite soll in einer Meldung angezeigt werden, wie lange diese angeschaut worden ist.

Animierte Schaltflächen

Animierte Schaltflächen hast du sicherlich schon einmal gesehen: sie werden „lebendig“, wenn der Mauszeiger in ihre Nähe kommt. In Abb. 3 ist eine mögliche Animation zu sehen. Sobald der Mauszeiger auf die Schaltfläche gelangt, wechselt die Schrift ihre Farbe von weiß nach schwarz. Sie wird wieder weiß, sobald der Mauszeiger die Schaltfläche verlässt. Beim Anklicken erhält die Schrift eine rote Farbe.

Abb. 3: Animation einer Schaltfläche

Wie kann man diese Animation mit JavaScript realisieren? Der Trick besteht darin, als Schaltfläche ein Bildobjekt zu benutzen, welches bei den verschiedenen Ereignissen seinen Inhalt, d. h. seine Quelle ändert. Dazu wollen wir die JavaScript-Funktionen `weiss2schwarz()`, `schwarz2weiss()` und `gedrueckt()` benutzen. Hier sollen dem Bildobjekt `knopf` die Quellen `klick2.gif`, `klick1.gif` bzw. `klick3.gif` zugeordnet werden.

Dabei wird die Funktion `weiss2schwarz()` vom onMouseOver-Ereignis ausgelöst, die Funktion `schwarz2weiss()` vom onMouseOut-Ereignis und die Funktion `gedrueckt()` vom onClick-Ereignis. Leider kennt das Bildobjekt keinen Event-Handler für das onClick-Ereignis, dies zeigt ein Blick in unsere Tabelle. Wir können aber unser Bildobjekt zum Bestandteil eines Ankerobjektes machen:

```
<a ...> <img ...></a>
```

Dieses Ankerobjekt kann nun ein onClick-Ereignis verarbeiten. Wenn wir jetzt dieses mit der Funktion `gedrueckt()` verknüpfen, dann wird diese Funktion ausgeübt, sobald das Bild angeklickt wird. Das vollständige HTML-Dokument sieht dann so aus:

```
<html>
<head>
<title>Animierte Schaltfläche</title>
</head>

<body>

<script language = "JavaScript">

function weiss2schwarz()
{
     knopf.src="klick2.gif";
}

function schwarz2weiss()
{
     knopf.src = "klick1.gif";
}

function gedrueckt()
{
     knopf.src = "klick3.gif";
}

</script>

<p><font size="6" Animierte Schaltfläche</font></p>

<p>
<a href = "#" onClick = "gedrueckt()">
<img      name = "knopf" src="klick1.gif"
          border = 0
```

```
          onMouseOver = "weiss2schwarz()"
          onMouseOut = "schwarz2weiss()">
</a>
</p>
</body>
</html>
```

Solange das HTML-Dokument und die drei Bilddateien auf der Festplatte vorliegen, läuft das Programm problemlos. Schwierigkeiten zeigen sich jedoch, wenn unser Dokument von einem Server über das Internet geladen wird. Zumindest am Anfang macht unsere Animation schlapp: die Bilder wechseln nur mit Verzögerung. Das liegt daran, dass erst nach einem entsprechenden Ereignis das jeweils benötigte Bildmaterial über das Internet angefordert wird. Und das braucht eben manchmal seine Zeit.

Damit solche störenden Verzögerungen nicht auftreten, bedient man sich wieder eines Tricks: Direkt beim Laden des Dokuments werden sofort alle drei Bilder in drei nicht-sichtbare Bildobjekte geladen. Sie stehen ab da fortwährend zur Verfügung und müssen nicht mehr vom Server geholt werden, wenn sie für den Austausch benutzt werden. Hier sind die erforderlichen Ergänzungen und Änderungen angegeben:

```
...

<body onload = "init()">

<script language = "JavaScript">

var bild1, bild2, bild3; // global

function init()
{
     //nicht sichtbare Bildobjekte werden erzeugt...
     bild1 = new Image();
     bild2 = new Image();
     bild3 = new Image();
     //... und die Bildquellen zugeordnet
     bild1.src = "klick1.gif";
     bild2.src = "klick2.gif";
     bild3.src = "klick3.gif";
}


function weiss2schwarz()
{
     knopf.src = bild2.src;
}

function schwarz2weiss()
{
     knopf.src = bild1.src;
}
```

```
function gedrueckt()
{
     knopf.src = bild3.src;
}

</script>

...
```

Aufgaben

1. Erkläre die Funktionsweise des oben stehenden Programms.

2. Ergänze die Startseite des Pizza-Projektes aus dem Kapitel „Wir gestalten Webseiten“ so, dass jeder Link, über dem der Mauszeiger steht, in der Statuszeile des Browsers erläutert wird.