

8.1.1 open() und close()

Ein Fenster kann mit der Methode »close()« geschlossen und mit der Methode »open()« geöffnet werden. Gibt man beim Aufruf von »open()« die URL als Argument an, lädt der Browser die entsprechende Seite in das neue Fenster.

listing08_01.html

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    function fenster_oeffnen() {
      neues_fenster = window.open("http://www.omnigena.com");
    }
  //-->
</script>
</head>
<body>
<form>
  <input type="button" value="Oeffnen" onclick="fenster_oeffnen()">
</form>
</body>
</html>
```

Auf Knopfdruck wird die Funktion »fenster_oeffnen()« aufgerufen, die mittels »window.open()« ein neues Fenster erzeugt.

Wir haben die Objektvariable »neues_fenster« definiert. Diese Variable stellt eine Referenz auf das neu erzeugte Fenster-Objekt dar. Wir können so alle Methoden, die wir für das window-Objekt nutzen, ebenso auf das Objekt »neues_fenster« anwenden. In der Funktion »fenster_schliessen()« wenden wird die Methode »close()« auf das Objekt an.

listing08_02.html

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    function fenster_oeffnen() {
      neues_fenster = window.open("http://www.omnigena.com");
    }
    function fenster_schliessen() {
      neues_fenster.close();
    }
  // -->
</script>
</head>

<body>
<form>
  <input type="button" value="Oeffnen" onclick="fenster_oeffnen()">
```

```

    <input type="button" value="Schliessen" onclick="fenster_schliessen()">
  </form>
</body>
</html>

```

Nun erweitern wir das Skript, sodass wir die Fenstergröße manipulieren können:

listing08_03.html

```

<html>
<head>
<script type="text/JavaScript">
  <!--
    function fenster_oeffnen() {
      neues_fenster = window.open("http://www.omnigena.com",
        "anzeigeFenster", "height = 384, width = 512");
    }
  // -->
</script>
</head>
<body>
<form>
  <input type="button" value="Oeffnen" onclick="fenster_oeffnen()">
</form>
</body>
</html>

```

Wir öffnen hierbei das Fenster nicht nur mit dem URL-Parameter, sondern übergeben noch zwei weitere Parameter, den Framenamen und eine Liste mit Parametern, die das Aussehen des Fensters bestimmen.

```
open([URL][Framename][Parameterliste]);
```

Bezogen auf das Beispiel:

```

open( "http://www.omnigena.com",
      "anzeigeFenster",
      "height = 384, width = 512");

```

Die URL ist hier "http://www.omnigena.com", der Framename "anzeigeFenster" und die Fenstereigenschaften "height = 384, width = 512". Bitte beachten Sie, dass alle Parameterangaben durch Kommata getrennt, in doppelte Hochkommata eingeschlossen werden.

Der Framename wird benötigt, um im HTML-Code das Ziel »target« zu bestimmen, indem eine Ausgabe stattfinden soll. Die häufigste Anwendung findet das »target«-Attribut im <href>-Tag, welcher einen Hyperlink definiert. Durch das »target«-Attribut wird festgelegt, in welcher Frame die referenzier-

te HTML-Seite geöffnet werden soll.

Folgende Skizze zeigt die manipulierbaren Elemente eines Fensters, die durch die »open()«-Methode des window-Objektes zur Verfügung stehen. Problematisch sind hierbei die browserspezifischen Eigenarten der Parameter.

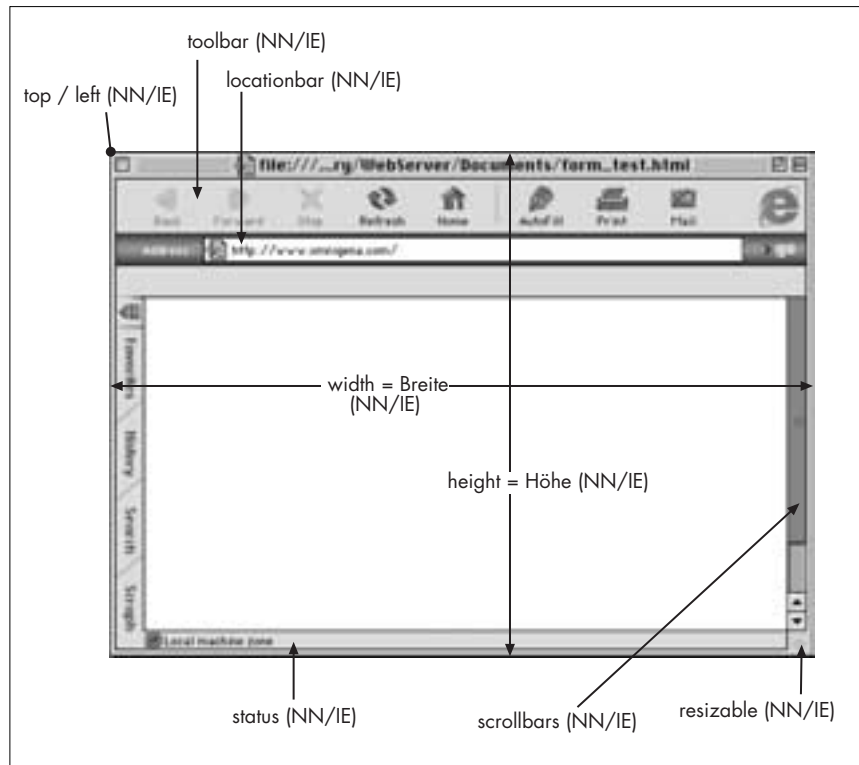


Abb. 8.3 Eigenschaften des window-Objektes

Zu beachten ist, dass im JavaScript v1.2 Standard »width« und »height« durch »innerWidth« und »innerHeight« abgelöst wurden. Trotzdem unterstützen bis dato alle Browser die alten Parameter »width« und »height«. Tatsächlich kann man nur durch deren Verwendung eine Lösung realisieren, die auf allen gängigen Browsern lauffähig ist.

| Parameter | Wirkung | Wert |
|-------------|---|---------------|
| toolbar | Werkzeugleiste IE/NN | yes/no |
| locationbar | Adresszeile mit URL NN | yes/no |
| location | Adresszeile mit URL NN | yes/no |
| personalbar | Anwender Werkzeugleiste NN | yes/no |
| menubar | Menueleiste NN | yes/no |
| scrollbars | Rollbalken IE/NN | yes/no |
| status | Statuszeile IE/NN | yes/no |
| resizable | Größe veränderbar IE/NN | yes/no |
| width | Breite des Fensters in Pixel IE/NN | Wert in Pixel |
| height | Höhe des Fensters in Pixel IE/NN | Wert in Pixel |
| innerHeight | Höhe des Anzeigebereiches NN | Wert in Pixel |
| innerWidth | Breite des Anzeigebereiches NN | Wert in Pixel |
| left | Position der oberen linken Ecke horiz. IE/NN | Wert in Pixel |
| top | Position der oberen linken Ecke vertik. IE/NN | Wert in Pixel |
| hotkeys | Tastaturbefehle zum Steuern des Browsers NN | yes/no |
| dependent | Fenster wird geschlossen, wenn Elternfenster schliesst NN | yes/nn |

Tab. 8.1 Eigenschaften des window-Objektes

8.1.2 Drucken des Fensterinhaltes

Häufig möchte man den Inhalt eines Fensters ausdrucken. Beispielsweise bei der Anzeige von Bestellformularen. Hierfür steht die »print()«-Methode zur Verfügung. Die »print()«-Methode erwartet keine Parameter.

```
window.print()
```

8.1.3 Bewegen eines Fensters

Soll ein neu geöffnetes Fenster an eine vom Anwender definierte Position auf dem Bildschirm gerückt werden, so steht uns hierfür die Methode »moveTo()« zur Verfügung.

```
<html>
<head>
<script type="text/JavaScript">
  <!--
  function fenster_oeffnen(){
    var position_x = 0;
    var position_y = 0;
```

listing08_04.html

```

    neues_fenster = window.open("", "", "width=200 , height=200");

    // *****
    // ***** bewegt Fenster an die neue Position (0/0) *****
    // ***** entspricht linker oberer Bildschirmcke *****
    // *****
    neues_fenster.moveTo( position_x, position_y );
}
// -->
</script>
</head>
<body onload = "fenster_oeffnen()"> <!-- ruft die JavaScript Funktion auf sobald das
                                Fenster geladen ist -->

</body>
</html>

```

Die »moveTo()«-Methode erwartet zwei Parameter, die x- und die y-Koordinate. Der Nullpunkt (0/0) befindet sich in der linken, oberen Bildschirmcke.

Ein anderer Effekt ist der sogenannte *Erdbebeneffekt*, der das Fenster einige Sekunden lang »beben« lässt. Schauen wir uns das Skript an:

listing08_05.html

```

<html>
<head>
<script type="text/JavaScript">
<!--
function erdbeben(){
    var position_x = 0, position_y = 0;
    for ( var ii = 0 ; ii < 200 ; ii++ ){
        position_x = parseInt(Math.random() * 20 + 1); // erzeugt eine Zufallszahl
        position_y = parseInt(Math.random() * 20 + 1); // erzeugt eine Zufallszahl

        window.moveTo( position_x, position_y ); // bewegt Fenster an die neue
                                                // Position
    }
}
//-->
</script>
</head>
<body onload = "erdbeben()"> <!-- ruft die JavaScript Funktion erdbeben() auf -->
</body>
</html>

```

In diesem Skript rufen wir nach dem Laden des Browserfensters die Funktion »erdbeben()« auf. Wir erreichen dieses mit Hilfe des Event-Handlers »onLoad« (siehe Kap. 11.5). Danach programmieren wir eine Schleife, die 200-mal wiederholt wird. In dieser Schleife wird bei jeder Iteration eine Zufallszahl für die x-Koordinate und eine für die y-Koordinate zwischen 1 und 20 erzeugt. Diese Zufallszahlen werden dann für die x- und y-Koordinate in die »moveTo()«-Methode eingesetzt, die das Fenster dann an diese Position bewegt. Fol-

gende Skizze veranschaulicht noch einmal die Arbeitsweise des Skripts:

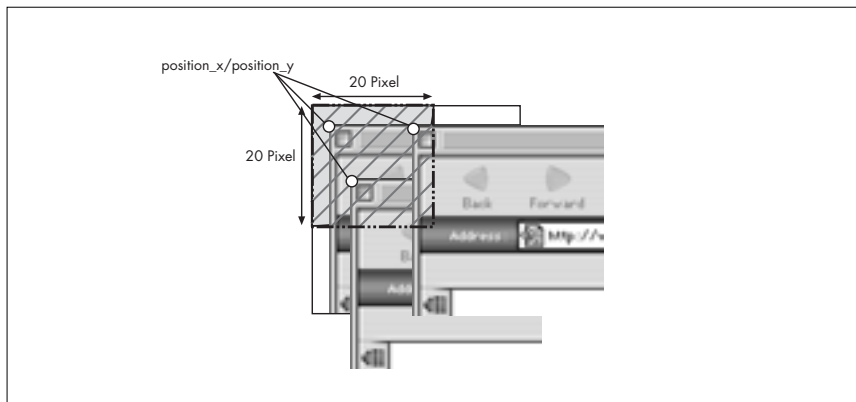


Abb. 8.4 Erdbebeneffekt

Im Radius von 20 Pixeln auf der x-Achse und 20 Pixeln auf der y-Achse kann die obere linke Ecke des Browserfensters positioniert werden. Die Positionierung erfolgt zufällig, wodurch der Wackeleffekt zustande kommt.

8.1.4 Vollbildfenster öffnen

Soll beispielsweise ein Fenster über den gesamten zur Verfügung stehenden Bildschirm ausgedehnt werden, muss zunächst die Bildschirmauflösung abgefragt werden. Es ist möglich, mit dem screen-Objekt die Bildschirmauflösung des Anwenders abzufragen.

```
<html>
<head>
<script type = "text/JavaScript">
<!--
    function fenster_max(){

        var bildschirm_hoehe = 0;
        var bildschirm_breite = 0;

        bildschirm_hoehe = screen.height;
        bildschirm_breite = screen.width;

        self.moveTo( 0, 0 );
        self.resizeTo( bildschirm_breite, bildschirm_hoehe );

    }
// -->
</script>
```

listing08_06.html

```
</head>
<body bgcolor="#d0ddd0" onload = "fenster_max()" >
</body>
</html>
```

In dem o.g. Skript wird nach dem Laden des Browserfensters die Funktion »fenster_max()« aufgerufen. Diese Funktion fragt zuerst einmal die Bildschirmauflösung des Anwenders mit Hilfe des screen-Objektes ab. Die Bildschirmhöhe wird mit

screen.height

und die Bildschirmbreite mit

screen.width

ermittelt. Darauf wird das Fenster mit Hilfe der Methode »moveTo(0/0)« an die Position (0/0) bewegt. Als Letztes wird die Größe des Fensters mit der »resizeTo()«-Methode auf die zuvor ermittelte Bildschirmgröße gesetzt. Erwähnt sei an dieser Stelle noch das self-Objekt, welches auch im Skript verwendet wird. Dieses bezeichnet immer das aktuell geöffnete aktive Fenster.

8.1.5 alert()

Die Methode »alert()« erzeugt ein Hinweisfenster mit einem Text und einem OK-Knopf. Es wird erzeugt durch den Aufruf:

```
alert("Die Variable ist falsch!");
```

Das Aussehen dieser Mitteilungsbox ist systemabhängig, sieht also unter Windows anders aus als unter MAC OS. Jedoch enthält die Box immer einen Button. Um einen Zeilenumbruch innerhalb des auszugebenden Textes zu erzeugen, verwendet man das Zeichen »\n«.

listing08_07.html

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    function al_box_oeffnen() {
      alert("Das ist eine Alert-Box! \n und dass ist gut so!");
    }
  //-->
</script>
</head>
```

```

<body>
<form>
  <input type="button" value="Alert-Box" onclick="al_box_oeffnen()">
</form>
</body>
</html>

```



Abb. 8.5 Aussehen einer Alert-Box

8.1.6 confirm()

Die Methode »confirm()« öffnet ein Dialogfenster mit einem Text und zwei Schaltern: *OK* und *Cancel* bzw. bei deutschsprachigen Browsern *OK* und *Abbrechen*.

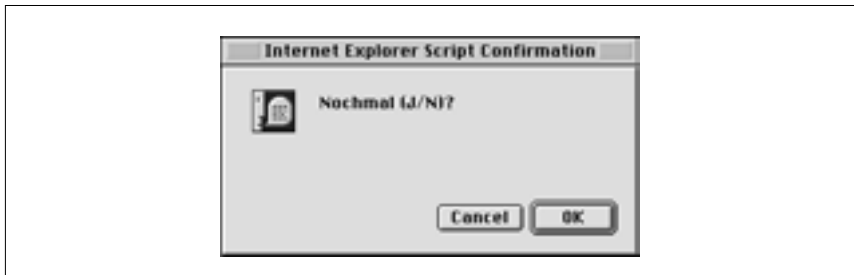


Abb. 8.6 Aussehen einer Confirm-Box

Sie wird erzeugt durch den Aufruf:

```
a = confirm("Beispiel einer Confirmbox");
```

Wird der OK-Knopf gedrückt, liefert »confirm()« den Wert »true«. Wird der Cancel-Knopf bzw. Abbrechen-Button gedrückt, liefert »confirm()« den Wert »false«. So kann man je nach Wert, den die Variable »a« hat, mit Hilfe einer

if-Verzweigung zu der entsprechenden Programmstelle springen.

listing08_08.html

```
<html>
<head>
<script type="text/JavaScript">
  <!--

      function conf_box_oeffnen() {
        a=confirm("Das ist eine Confirm-Box!");
        if ( a == true ){
          document.write("Sie haben den OK-Button gedr&uuml;ckt!");
          // Hier koennte auch der Programmcode stehen, der ausgefuehrt
          // wird, wenn der OK-Button gedrueckt wurde
        }
      }

  // -->
</script>
</head>
<body>
<form>
  <input type="button" value="Confirm-Box" onclick="conf_box_oeffnen()">
</form>
</body>
</html>
```

8.1.7 prompt()

Die Methode »prompt()« öffnet ein Eingabefenster mit einem einzeiligen Eingabefeld und zwei Schaltern: *OK* und *Cancel* bzw. bei deutschsprachigen Browsern *Abbrechen*. Der Programmierer kann beim Aufruf von »prompt()« zwei Argumente angeben.

1. Den Text, der den Anwender zur Eingabe auffordern soll.
2. Einen Standardtext für das Eingabefeld.

Die Methode »prompt()« liefert »false« zurück, wenn der Abbrechen-Knopf gedrückt wird, sonst wird die Zeichenkette zurückgeliefert.

```
eingabe = prompt ("Bitte geben Sie Ihren Namen ein!","");
```

Geben Sie das zweite Argument nicht an, erscheint im Eingabefeld »<undefinied>«.

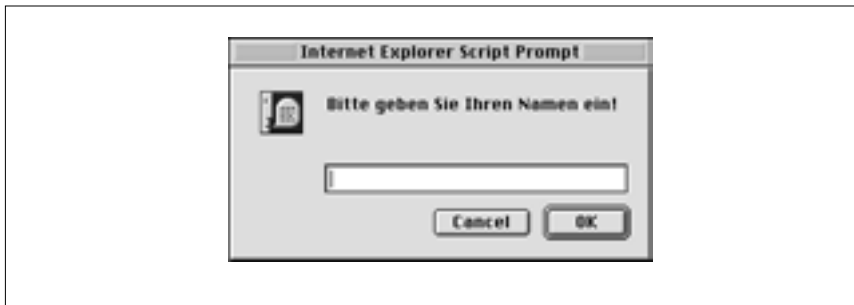


Abb. 8.7 Aussehen einer Prompt-Box

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    function pro_box_oeffnen() {
      a = prompt("Wo wohnen Sie?", "");
    }
    // -->
</script>
</head>
<body>
<form>
  <input type="button" value="Prompt-Box" onclick="pro_box_oeffnen()">
</form>
</body>
</html>
```

listing08_09.html

9. Browserversion prüfen

Mit steigender Komplexität der Programme wächst auch die Gefahr, dass sie nicht mehr mit allen Webbrowsern problemlos funktionieren. Damit Sie Skripte browserangepasst schreiben können, möchte ich Ihnen jetzt zeigen, wie man die Browser unterscheiden kann.

Folgende Möglichkeiten haben sich bewährt:

1. Auslesen des navigator-Objekts;
2. JavaScript-Version im <script>-Tag ermitteln;
3. Existenz eines versionsabhängigen Objekts ermitteln, bevor es verwendet wird.

9.1 Auslesen des navigator-Objekts

Mit der Funktion »indexOf()« kann man überprüfen, ob eine bestimmte Zeichenkette in einem String enthalten ist (siehe Kap. 8.9.2). Schauen wir uns also folgendes Skript an:

```
listing09_01.html    <html>
                    <head> </head>
                    <body>
                    <script type="text/JavaScript">
                    <!--
                    if (navigator.userAgent.indexOf("MSIE") != -1){
                    // MSIE steht für Internet Explorer
                    document.write("Internet Explorer!");
                    }
                    else {
                    document.write("kein Internet Explorer!"); // kein MSIE
                    }
                    // -->
                    </script>
                    </body>
                    </html>
```

Liefert also »indexOf()« den Wert »-1«, wurde »MSIE« in dem String nicht ge-

funden. Daraus folgt: Der Anwender benutzt einen anderen Browser als den Microsoft Internet Explorer.

Problematisch ist dies, wenn der Anwender einen Webbrowser verwendet, der vorgibt, ein anderer zu sein, ohne dessen Funktionalität zu besitzen. Beispielsweise wäre dieses beim Opera-Browser der Fall, der sich als Internet Explorer ausgeben kann. Hier noch einmal analog das Beispiel für den Netscape Navigator:

```
<html>
<head>
</head>
<body>
<script type="text/JavaScript">
  <!--
    if (navigator.userAgent.indexOf("MSIE") == -1) {
      // Netscape Browser
      document.write("Netscape Navigator.");
    }
    else {
      // kein Netscape Browser
      document.write("kein Netscape Navigator.");
    }
  // -->
</script>
</body>
</html>
```

listing09_02.html

9.2 JavaScript-Version im <script>-Tag ermitteln

Wie bereits gesehen, können im <script>-Tag verschiedene JavaScript-Versionen angegeben werden. So wird das folgende Beispiel nur von einem JavaScript v1.2-fähigen Browser ausgeführt. Jedoch nicht von einem Browser, der JavaScript v.1.2 nicht versteht.

```
<html>
<head> </head>
<body>
<script type="text/JavaScript" language="JavaScript1.2">
  <!--
    document.write("JavaScript 1.2 Browser!");
  // -->
</script>
</body>
</html>
```

listing09_03.html

Wenn Sie allerdings eine Funktion zwischen solchen <script>-Tags definieren,

kann es zu Problemen mit älteren Browsern kommen.

listing09_04.html

```
<html>
<head>
<script language="JavaScript1.2">
  <!--
    function test() {
      alert("JavaScript 1.2 Browser!");
    }
  // -->
</script>
</head>
<body>
<form>
  <input type="button" value="test" onclick="test()">
</form>
</body>
</html>
```

Das Beispiel funktioniert ohne Probleme in einem JavaScript v1.2-fähigen Browser (Netscape 4.0 oder Internet Explorer 4.0). In älteren Browsern wird jedoch eine Fehlermeldung ausgegeben, wenn Sie auf den Knopf klicken, da die Funktion »test()« nicht definiert ist. Sie können das Problem wie folgt lösen:

listing09_05.html

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    var browser_flag = false;
    function test() {
      if (browser_flag == true){
        alert("JavaScript 1.2 Browser!");
      }
    }
  // -->
</script>
<script language="JavaScript1.2">
  browser_flag = true;
</script>
</head>
<body>
<form>
  <input type="button" value="test" onclick="test()">
</form>
</body>
</html>
```

Problematisch ist dieses, wenn unterschiedliche JavaScript-Versionen nicht abwärtskompatibel sind.

9.3. Objektexistenz überprüfen

Wenn ein bestimmtes Objekt verwendet werden soll, dass nur in neueren Browserversionen existiert, kann man vor dessen Verwendung die Existenz überprüfen, um Fehlermeldungen zu vermeiden.

```
<html>
<head>
<script type="text/JavaScript">
  <!--
    if ( document.select != null) {
      alert( "die select-Eigenschaft ist vorhanden");
    }
  // -->
</script>
</head>
<body>
</body>
</html>
```

listing09_06.html

Wie zu sehen ist, ist keine der Überprüfungsverfahren perfekt. Je größer die Anzahl der Benutzer ist, um so wahrscheinlicher wird es sein, dass es zu Problemen mit verschiedenen Browserversionen kommt.

Übungen und Aufgaben

- 9.1 Es soll ein Skript geschrieben werden, dass überprüft, welches Betriebssystem der Anwender benutzt, und das Ergebnis auf dem Browserfenster ausgibt. Verwenden Sie hierzu die Eigenschaften des navigator-Objektes!
- 9.2 Es soll ein Skript entwickelt werden, dass die Browsersprache ermittelt. Es soll zwischen deutschsprachigen und einem englischsprachigen Browsern unterschieden werden. Je nach Browsertyp öffnet sich ein Fenster mit einer deutschen oder einer englischen Begrüßung!