101 praktycznych skryptów na stronę WWW. Wydanie II

Autor: Marcin Lis
ISBN: 83-246-0278-X
Format: B5, stron: 216

Wykorzystaj możliwości technologii skryptowych i uatrakcyjnij swoją stronę WWW

- Dodaj do strony mechanizm weryfikacji danych wpisanych przez użytkownika
- Kontroluj parametry okien przeglądarki
- Stwórz efekty specjalne w oparciu o warstwy
- Wprowadź elementy interaktywne

Dziś, gdy witrynę WWW można stworzyć praktycznie bez znajomości języka HTML, własne miejsce w sieci może mieć każdy. Cóż więc zrobić, by wyróżnić swoją stronę spośród setek tysięcy innych? Zwykła, statyczna strona to zdecydowanie za mało. Według regularnie przeprowadzanych badań największą popularnością wśród odwiedzających cieszą się witryny interaktywne z elementami animacji.
Nic prostszego – dokładamy do strony animacje! Tylko jak? Nie każdy ma czas na naukę nowych technologii, a w dodatku nie wiadomo, którą z nich wybrać. Napisanie skryptu w JavaScript lub dowolnym innym języku zwykle przerasta możliwości początkującego webmastera.

Książka „101 praktycznych skryptów na stronę WWW. Wydanie II" to zbiór gotowych do wykorzystania skryptów stworzonych za pomocą DHTML i JavaScript, dzięki którym uatrakcyjnisz każdą witrynę WWW. Wszystkie zaprezentowane tu skrypty są gotowe do uruchomienia – wystarczy wkleić kod źródłowy do kodu HTML. Każdy z nich może również być podstawą do tworzenia innych ciekawych efektów i materiałem do nauki praktycznych zastosowań języka JavaScript i dynamicznego HTML.

- Wyświetlanie okien dialogowych
- Zmiana kształtu kursora myszy
- Wyszukiwanie łańcuchów znaków na stronie
- Obsługa formularzy
- Otwieranie nowych okien przeglądarki o zadanych parametrach
- Pobieranie informacji o przeglądarce i rozdzielczości ekranu
- Wzbogacanie wyglądu hiperłączy
- Animowanie napisów
- Wyświetlanie banerów
- Autoryzacja użytkowników i korzystanie z plików cookies
- Animacje warstw

Przekonaj się, jak wiele można zmienić na stronie WWW za pomocą prostych skryptów

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 8. Animacje warstw 161

Rozdział 1.
Skrypty różne

Naszą wyprawę w krainę skryptów zaczniemy od stosunkowo prostych konstrukcji, które wykorzystują podstawowe zdarzenia zachodzące na stronie WWW. Omówimy zdarzenia takie jak kliknięcie myszą, najechanie kursorem na element witryny czy też załadowanie strony do przeglądarki. Ich oprogramowanie zazwyczaj nie jest skomplikowane — używamy tu typowych procedur obsługi.

Ogólny schemat postępowania wygląda w tym przypadku następująco:

```
<znacznik_html nazwa_zdarzenia="procedura_obsługi">
```

`procedura_obsługi` jest w tym przypadku kodem wykonywanym po zajściu zdarzenia na stronie. Jeśli nie jest on zbyt rozbudowany, możemy w całości umieścić go w znaczniku. Jeśli jednak planujemy bardziej zaawansowane działanie, lepiej jest napisać dodatkową funkcję i w znaczniku tylko ją wywoływać. Zatem szablon HTML wyglądałby następująco:

```
<html>
<head>
<script type="text/javascript">
function moja_funkcja()
{
  //treść funkcji obsługującej zdarzenie
}
</script>
</head>
<body>
<znacznik_html nazwa_zdarzenia="moja_funkcja();">
</znacznik_html>
</body>
</html>
```

Obydwa sposoby będą się pojawiać w prezentowanych przykładach, jednak częściej będziemy stosować ten drugi.

Skrypt 1. Skrypt, który po załadowaniu strony WWW wyświetla okno dialogowe

[E][F][K][N][O]

Skrypt ten wykorzystuje zdarzenie `onLoad`, które ma miejsce po załadowaniu strony przez przeglądarkę. Naszym zadaniem będzie wyświetlenie w takiej sytuacji typowego okna dialogowego z dowolnym tekstem. Wykorzystamy do tego funkcję `alert`, której jako parametr należy podać żądany tekst. Całą procedurę można bez problemu umieścić w znaczniku `<body>`. Jednak — nieco wbrew nazwie — nie należy wykorzystywać tego efektu do witania użytkownika, gdyż takie wyskakujące okno może go raczej zirytować niż zachęcić do częstego odwiedzania naszej witryny. Taki skrypt stosujemy tylko wtedy, jeśli zachodzi konieczność wyświetlenia szczególnie ważnej informacji, której odwiedzający witrynę nie powinni przeoczyć.

Warte uwagi jest to, że okno dialogowe (rysunek 1.1) pojawi się dopiero po pełnym załadowaniu strony. Jeśli zatem w treści umieściliśmy wywołania jakichś innych funkcji, zostaną one wykonane w pierwszej kolejności.

Rysunek 1.1. *Po załadowaniu strony ukazuje się okno dialogowe*

Skrypt 1

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
</head>
<body onLoad = "alert ('Uwaga! Serwis tylko dla osób pełnoletnich!')">
<h1 align="center">
```

```
Moja strona WWW
</h1>
</body>
</html>
```

Skrypt 2. Skrypt, który przy opuszczaniu strony WWW wyświetla okno pożegnalne [E][F][K][N][O]

Zdarzeniem analogicznym do `onLoad` jest `onUnload`, ma ono jednak miejsce przy opuszczaniu strony. Można je zatem wykorzystać do wyświetlenia okna dialogowego pojawiającego się, kiedy użytkownik będzie opuszczał naszą witrynę. Procedurę obsługi tego zdarzenia umieszczamy również w sekcji `<body>`. Oczywiście, podobnie jak w poprzednim przykładzie (skrypt 1.), nie należy tego efektu nadużywać, ale stosować w szczególnych przypadkach.

Skrypt 2

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
</head>
<body onUnload="alert ('Uwaga! Za pięć dni zamykamy serwis!')">
<h1 align="center">
Moja strona WWW
</h1>
</body>
</html>
```

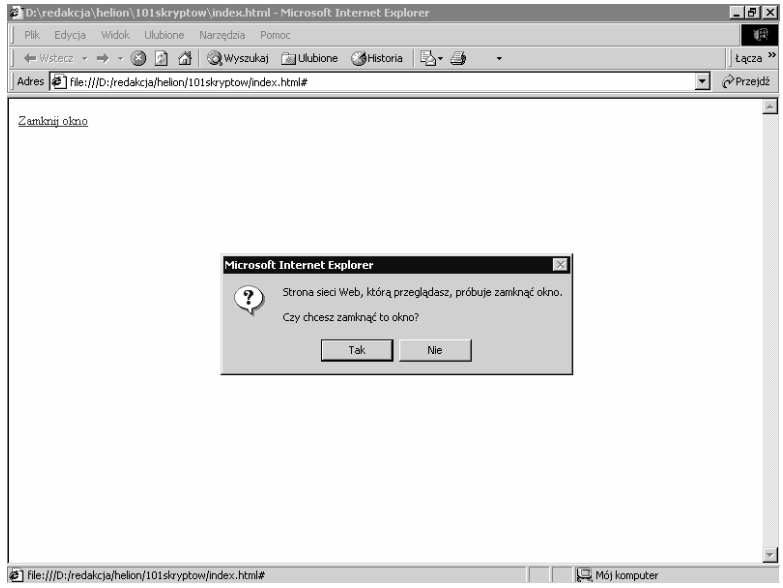
Skrypt 3. Skrypt, który po kliknięciu odnośnika umieszczonego w dokumencie zamyka okno przeglądarki [E][F][K][N][O]

Skrypt realizujący to zadanie ma bardzo prostą konstrukcję i chyba nie wymaga dokładnego tłumaczenia. Wykorzystamy tutaj metodę `close` obiektu `window`, która zamyka okno przeglądarki i zdarzenie `onClick` przypisane do znacznika `HTML <a>`. Dzięki temu kliknięcie odnośnika będzie powodowało zamknięcie okna przeglądarki.

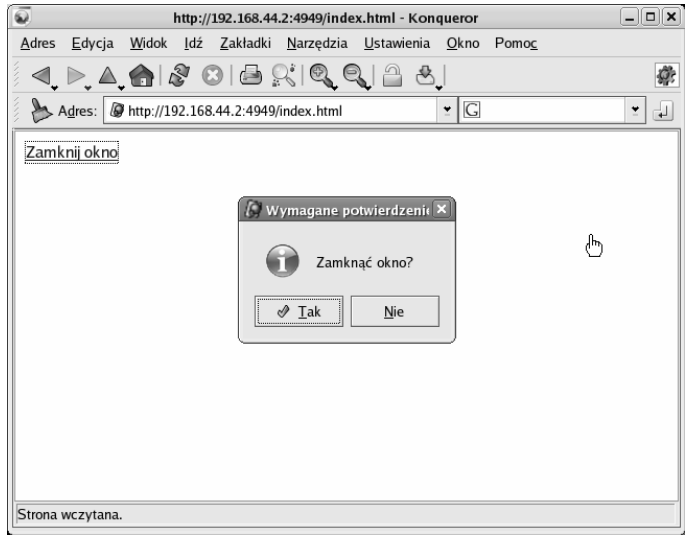
Skrypt 3

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
</head>
<body>
<a href="#" onClick="window.close()">
Zamknij okno
</a>
</body>
</html>
```

Zachowanie skryptu jest zależne od przeglądarki, w której został on uruchomiony. W przypadku Internet Explorera (rysunek 1.2) i Konquerora (rysunek 1.3) zostanie nam zadane pytanie, czy na pewno chcemy zamknąć okno, natomiast Opera zamyka je bez pytania. W przypadku przeglądarek opartych na module Gekko (FireFox, Nestcape Navigator) zamknąć można jedynie okno, które zostało wcześniej otwarte przez sam skrypt. Zatem w przypadku tych przeglądarek za pomocą przedstawionego kodu nie można zamknąć okna głównego.



Rysunek 1.2. *Próba zamknięcia okna powoduje reakcję Internet Explorera*



Rysunek 1.3. *Reakcja Konquerora na próbę zamknięcia głównego okna przeglądarki*

Skrypt 4. Skrypt, który po kliknięciu przycisku pyta użytkownika, czy na pewno chce opuścić bieżącą stronę

[E][F][K][N][O]

W tym skrypcie wykorzystamy zdarzenie `onClick` przypisane do przycisku, który jest standardowym elementem tworzonym za pomocą znacznika `<input>`. Wykorzystamy również metodę `confirm`. Jako jej parametr należy podać tekst, który zostanie wyświetlony w oknie dialogowym. W naszym przypadku jest to pytanie o to, czy chcemy przejść na nową stronę. Metoda `confirm` zwraca wartość `true`, jeżeli użytkownik kliknie przycisk *OK*, lub `false`, jeżeli kliknie *Anuluj*. Przypisanie:

```
window.location.href="http://helion.pl";
```

zmienia lokalizację bieżącej strony na adres *http://helion.pl*.

Skrypt 4

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis                http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function obsluga_zdarzenia()
```

Rysunek 1.4. *Po kliknięciu skrypt prosi o potwierdzenie chęci przejścia do nowej witryny*


```
{
  if (confirm ('Czy na pewno chcesz przejść na stronę helion.pl?'))
    window.location.href = "http://helion.pl";
}
// Koniec kodu JavaScript -->
</script>
</head>
<body>
<div align="center">
<input
 type="button"
 value="helion.pl"
 onClick="obsluga_zdarzenia();"
>
</div>
</body>
</html>
```

Skrypt 5. Skrypt, który zmienia napis na pasku stanu, po najechaniu kursorem lub kliknięciu odnośnika [E][F][K][N][O]

Po uruchomieniu skryptu na pasku stanu wyświetlany będzie tekst *Przykładowa strona z obsługą skryptów*. Naprowadzenie kursora na odnośnik spowoduje wyświetlenie napisu *Strona Wydawnictwa Helion* (rysunek 1.5), natomiast przesunięcie go poza odnośnik przywróci na pasku stanu napis pierwotny.

Rysunek 1.5. *Po najechaniu na odnośnik zmienia się napis na pasku stanu*

Takie zachowanie osiągamy dzięki oprogramowaniu procedur `onMouseOver` i `onMouseOut`. W obu przypadkach jest wywoływana funkcja o nazwie `setText`, która zmieni tekst paska stanu na ciąg znaków przekazany jej w postaci argumentu. W przypadku kliknięcia odnośnika będzie wywoływana procedura zdarzenia `onClick`, czyli również funkcja `setText`, i na krótką chwilę pojawi się napis *Otwieram nową stronę*.

Zmianę napisu na pasku stanu osiągamy przez przypisanie odpowiedniego ciągu znaków właściwości `status` obiektu `window`. Dokonujemy tego za pomocą instrukcji:

```
window.status="napis";
```

Napis domyślny zmieniamy dzięki przypisaniu:

```
window.defaultStatus="napis";
```

W przypadku przeglądarek FireFox i Netscape Navigator do prawidłowego działania skryptu niezbędne jest włączenie w opcjach tych programów możliwości podmiany napisów na pasku stanu.

Skrypt 5

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function showText(text)
{
  window.status=text;
}
window.defaultStatus="Przykładowa strona z obsługą skryptów";
```

```
// Koniec kodu JavaScript -->
</script>
</head>
<body>
<h2 align="center">
<a href="http://helion.pl"
   onClick="showText('Otwieram nową stronę');return true;"
   onMouseOver="showText('Strona Wydawnictwa Helion');return true;"
   onMouseOut="showText('Przykładowa strona z obsługą skryptów');return true;"
>
Strona Wydawnictwa Helion
</a>
</h2>
</body>
</html>
```

Skrypt 6. Skrypt, który po najechaniu na odnośnik zmienia wygląd kursora myszy na „celownik” [E][F][K][N][O]

Modyfikację wyglądu kursora myszy możemy osiągnąć, zmieniając styl przypisany do danego obiektu. Tym razem zamiast wywoływać procedurę obsługi zdarzenia w postaci:

```
<znacznik_html onMouseOver="procedura_obslugi">
```

zmodyfikujemy bezpośrednio styl CSS przypisany do znacznika, co schematycznie można przedstawić następująco:

```
<znacznik_html style="definicja_stylu">
```

Skoro chcemy zmienić wygląd kursora na celownik, definicja stylu będzie wyglądała tak:

```
style="cursor:crosshair"
```

Skrypt 6

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
</head>
<body>
<h2 align="center">
<a href="http://helion.pl"
   style="cursor:crosshair">
```

```
Wydawnictwo Helion
</a>
</h2>
</body>
</html>
```

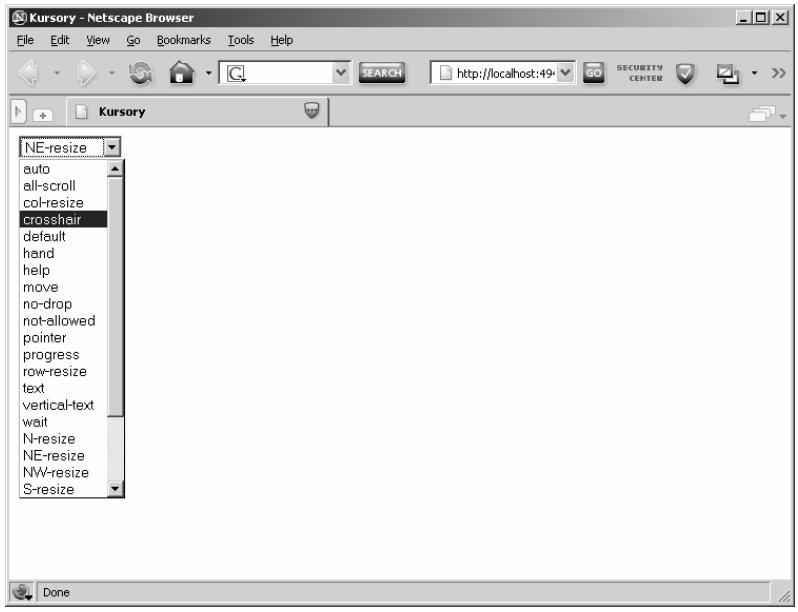
Skrypt 7. Skrypt umożliwiający wybranie kształtu kursora myszy z listy [E][F][K][N][O]

Działanie skryptu będzie zależało od przeglądarki, w której został uruchomiony, gdyż programy te rozpoznają różne zestawy kursorów. Wygląd kursora możemy zmienić dzięki przypisaniu:

```
document.body.style.cursor = nazwa_kursora;
```

Dopuszczalne nazwy kursorów wraz z ich opisem oraz oznaczeniem dostępności w poszczególnych przeglądarkach zostały przedstawione w tabeli 1.1.

Kod przedstawiony w skrypcie 7. wyświetla na ekranie listę dostępnych kursorów (rysunek 1.6). Po wybraniu danej pozycji odpowiedni kursor jest przypisywany do dokumentu.



Rysunek 1.6. *Wybór kształtu kursora z listy*

Tabela 1.1. *Nazwy kursorów wraz z opisami*

Nazwa	Opis	Znaczenie	E	F	K	N	O
all-scroll	Strzałki skierowane w czterech kierunkach.	Strona może być przewijana we wszystkich kierunkach.	Tak	Nie	Nie	Nie	Nie
auto	Kursor domyślny.	Przeglądarka sama dobiera wygląd kursora zależnie od obiektu, nad którym się znajduje.	Tak	Tak	Tak	Tak	Tak
col-resize	Dwie strzałki wskazujące w prawo i w lewo z pionową linią rozdzielającą.	Obiekt (kolumna) może być skalowany w poziomie.	Tak	Nie	Nie	Nie	Nie
crosshair	Dwie linie krzyżujące się pod kątem prostym.	Wskaźnik, celownik.	Tak	Tak	Tak	Tak	Tak
default	Kursor domyślny, zazwyczaj strzałka.	Domyślny kursor przeglądarki.	Tak	Tak	Tak	Tak	Tak
hand	Ręka z palcem wskazującym.	Kursor pojawiający się domyślnie po najechaniu na odnośnik.	Tak	Nie	Tak	Nie	Tak
help	Strzałka ze znakiem zapytania.	Wskazuje, że pomoc jest dostępna.	Tak	Tak	Tak	Tak	Tak
move	Strzałki skrzyżowane pod kątem prostym.	Obiekt jest przemieszczany przez użytkownika.	Tak	Tak	Tak	Tak	Tak
no-drop	Dłoń z niewielkim przekreślonym kółkiem.	W technice „przeciągnij i upuść” — obiekt nie może być zwolniony w tym miejscu.	Tak	Nie	Nie	Nie	Nie
not-allowed	Przekreślone kółko.	Żądana czynność nie może być wykonana.	Tak	Nie	Nie	Nie	Nie
pointer	Ręka z palcem wskazującym.	Kursor pojawiający się domyślnie po najechaniu na odnośnik.	Tak	Tak	Tak	Tak	Tak
progress	Strzałka z klepsydrą.	W tle wykonywana jest operacja.	Tak	Tak	Tak	Tak	Tak
row-resize	Dwie strzałki wskazujące w górę i w dół z poziomą linią rozdzielającą.	Obiekt/wiersz może być skalowany w pionie.	Tak	Nie	Nie	Nie	Nie
text	Najczęściej kursor w kształcie wielkiej litery I.	Kursor znajduje się nad tekstem.	Tak	Tak	Tak	Tak	Tak

Tabela 1.1. *Nazwy kursorów wraz z opisami (ciąg dalszy)*

Nazwa	Opis	Znaczenie	E	F	K	N	O
url(URL)	Kursor zdefiniowany przez użytkownika.	Kursor zostanie pobrany z lokalizacji podanej przez URL. Obsługiwane są formaty *.CUR* i *.ANI*.	Tak	Nie[1]	Nie	Nie	
vertical-text	Kursor w kształcie wielkiej litery I.	Możliwa jest edycja tekstu.	Tak	Nie	Nie	Nie	Nie
wait	Klepsydra lub zegarek.	Program jest zajęty, użytkownik powinien czekać.	Tak	Tak	Tak	Tak	Tak
N-resize	Strzałka wskazująca w górę (na północ).	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
NE-resize	Strzałka wskazująca kierunek północno-wschodni.	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
NW-resize	Strzałka wskazująca kierunek północno-zachodni.	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
S-resize	Strzałka wskazująca w dół (na południe).	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
SE-resize	Strzałka wskazująca kierunek południowo-wschodni.	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
SW-resize	Strzałka wskazująca kierunek południowo-zachodni.	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
E-resize	Strzałka wskazująca w prawo (na wschód).	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak
W-resize	Strzałka wskazująca w lewo (na zachód).	Obiekt może być skalowany w podanym kierunku.	Tak	Tak	Tak	Tak	Tak

Lista jest tworzona za pomocą standardowej konstrukcji formularza HTML:

```
<form>
<select name="nazwa_listy"
        size="wielkość"
        onChange="procedura_obsługi();">
  <option value="nazwa_opcji">tekst opcji
>
</form>
```

[1] Według zapowiedzi obsługa zewnętrznych kursorów zostanie wprowadzona w wersji 1.5.

Zdarzeniu `onChange` przypisujemy procedurę obsługi, którą w tym przypadku jest funkcja `changeCursor`. Wewnątrz funkcji jest odczytywana wartość (`value`) wybranego elementu listy. Indeks tego elementu znajduje się we właściwości `selectedIndex` obiektu listy. W przypadku opcji *url* w katalogu, w którym umieszczony jest plik z kodem skryptu, musi znajdować się również plik o nazwie *kursor.cur* zawierający definicję kursora.

Skrypt 7

```
<html>
<head>
<title>Kursory</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis                http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function changeCursor()
{
  var lista = document.getElementById('lista');
  var name = lista[lista.selectedIndex].value;
  document.body.style.cursor=name;
}
//Koniec kodu JavaScript -->
</script>
</head>
<select id="lista"
        size="1"
        onChange="changeCursor();"
>
  <option value="auto">auto
  <option value="all-scroll">all-scroll
  <option value="col-resize">col-resize
  <option value="crosshair">crosshair
  <option value="default">default
  <option value="hand">hand
  <option value="help">help
  <option value="move">move
  <option value="no-drop">no-drop
  <option value="not-allowed">not-allowed
  <option value="pointer">pointer
  <option value="progress">progress
  <option value="row-resize">row-resize
  <option value="text">text
  <option value="vertical-text">vertical-text
  <option value="wait">wait
  <option value="N-resize">N-resize
  <option value="NE-resize">NE-resize
  <option value="NW-resize">NW-resize
  <option value="S-resize">S-resize
  <option value="SE-resize">SE-resize
  <option value="SW-resize">SW-resize
  <option value="E-resize">E-resize
  <option value="W-resize">W-resize
  <option value="url('kursor.cur')">url
```

```
</select>
</body>
</html>
```

Skrypt 8. Skrypt blokujący możliwość naciśnięcia prawego przycisku myszy [E][F][K][N]

Efekt blokady prawego przycisku myszy jest dosyć często spotykany w Internecie. Uniemożliwia on między innymi zapisanie grafiki znajdującej się na stronie, gdyż funkcja ta wymaga dostępu do podręcznego menu, które wywołujemy właśnie prawym przyciskiem. Oczywiście blokada taka będzie skuteczna jedynie dla niedoświadczonych użytkowników sieci. Wystarczy przecież zajrzeć do źródeł strony, aby przekonać się, gdzie znajdują się obrazki. Przeglądarki umożliwiają również zablokowanie działania skryptów, a grafika znajdzie się też w ich cache'u.

Można, co prawda, zastosować znacznik `meta` w postaci:

```
<meta http-equiv="pragma" content="no-cache">
```

jednak przeglądarki nie zawsze go respektują i taka strona często zostaje zapisana na dysku.

Blokada przycisku może być jednak ciekawym rozwiązaniem. Aby ją zrealizować, należy zdarzeniu `onmousedown` obiektu `document` przypisać naszą własną procedurę obsługi w postaci:

```
document.onmousedown=procedura_obslugi;
```

W samej procedurze musimy sprawdzić, który klawisz myszy został wciśnięty. W przypadku przeglądarki Internet Explorer badamy wartość parametru `button` obiektu `event`, która dla prawego przycisku jest równa `2`. Jeśli zatem `event` ma wartość `2`, korzystamy z funkcji `alert`, aby wyświetlić okno dialogowe z informacją o blokadzie (rysunek 1.7), i zwracamy wartość `false`; w przeciwnym wypadku zwracamy wartość `true`.

W przypadku pozostałych przeglądarek (Firefox, Konqueror, Netscape) badamy właściwość `which` obiektu `evt` przekazanego funkcji, która dla prawego klawisza myszy ma wartość `3`. Jeśli więc `which` ma wartość `3`, wyświetlamy okno dialogowe z informacją. Przeglądarka Konqueror wymaga dodatkowo, aby zwrócona została wartość `false` (co też czynimy). Przeglądarki FireFox i Netscape wymagają dodatkowego zablokowania wyświetlania menu kontekstowego, co robimy, przypisując właściwości `oncontextmenu` obiektu `document` procedurę obsługi w postaci funkcji `menuBlock`, której jedynym zadaniem jest zwrócenie wartości `false`[2].

[2] Warto jednak zauważyć, że przeglądarki te pozwalają (poprzez wybranie odpowiednich opcji z menu) na uniemożliwienie skryptom podmiany procedury obsługi dla zdarzenia `oncontextmenu` i blokowania menu kontekstowego.

Rysunek 1.7. *Blokowanie możliwości wykorzystania prawego przycisku myszy*

Skrypt 8

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Blokada prawego przycisku myszy</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function mouseClick(evt)
{
  if(evt){
    if(evt.which == 3){
      alert('Blokada prawego przycisku myszy!');
      return false;
    }
  }
  else{
    if (event.button == 2){
      alert('Blokada prawego przycisku myszy!');
      return false;
    }
    else{
      return true;
    }
  }
}
function menuBlock()
{
  return false;
}
document.onmousedown=mouseClick;
document.oncontextmenu=menuBlock;
// Koniec kodu JavaScript -->
</script>
```

```
</head>
<body>
<h2 align="center">Tutaj treść strony</h2>
</body>
</html>
```

Skrypt 9. Blokada lewego przycisku myszy [E][F][K][N][O]

Skoro wiemy, jak zablokować prawy przycisk myszy, nic nie stoi na przeszkodzie, aby zablokować także i przycisk główny — lewy. Kod lewego przycisku to 1, czyli blokada powinna wyglądać następująco:

```
if (event.button == 1){
  alert('Blokada lewego przycisku myszy!');
  return false;
}
```

lub

```
if(evt.which == 1){
      alert('Blokada lewego przycisku myszy!');
      return false;
}
```

w zależności od rodzaju przeglądarki.

Skrypt w takiej postaci z pewnością spowoduje, że użytkownicy będą omijać naszą stronę z daleka. Jednak przejęcie obsługi kliknięć może być przydatne, jeśli chcielibyśmy reagować na nie w jakiś szczególny sposób. Dlatego także taki efekt został zaprezentowany.

Skrypt 9

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Blokada lewego przycisku myszy</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function mouseClick(evt)
{
  if(evt){
    if(evt.which == 1){
      alert('Blokada lewego przycisku myszy!');
      return false;
    }
  }
  else{
```

```
    if (event.button == 1){
      alert('Blokada lewego przycisku myszy!');
      return false;
    }
    else{
      return true;
    }
  }
}
document.onmousedown=mouseClick;
// Koniec kodu JavaScript -->
</script>
</head>
<body>
<h2 align="center">Tutaj treść strony</h2>
</body>
</html>
```

Skrypt 10. Wyróżnienie komórki tabeli [E][F][K][N][O]

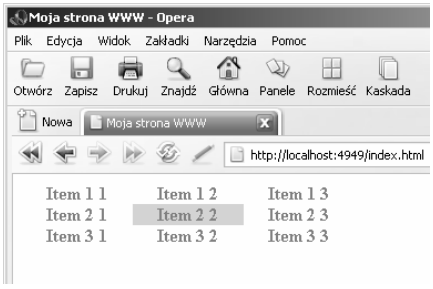
Kolor komórki tabeli możemy zmienić, stosując bezpośrednie przypisanie do zdarzenia `onMuseOver` w postaci:

```
onmouseover="bgColor='kolor'"
```

Na przykład dla koloru czerwonego będzie to:

```
onmouseover="bgColor='red'"
```

Jest to bardzo przydatny efekt pozwalający na wyróżnienie komórki, nad którą aktualnie znajduje się kursor myszy (rysunek 1.8). Dodatkowo można zmienić wygląd kursora, przypisując mu odpowiedni styl.



Rysunek 1.8. *Najechanie kursorem myszy powoduje zmianę koloru tła komórki tabeli*

Ostatecznie definicja pojedynczej komórki tabeli powinna wyglądać następująco:

```
<td  onmouseover="bgColor='kolor1'"
     style="cursor:pointer; cursor:hand"
     onmouseout="bgColor='kolor2' "bgcolor="kolor2"
```

```
     width="szerokość" align="wyrównywanieH"
     valign="wyrównywanieV">
<!--zawartość komórki -->
</td>
```

Znaczenie poszczególnych parametrów jest następujące:

- `kolor1` — kolor tła danej komórki po najechaniu kursorem myszy,
- `kolor2` — kolor domyślny komórki tabeli,
- `szerokość` — szerokość komórki tabeli,
- `wyrównywanieH` — wyrównywanie zwartości komórki w poziomie,
- `wyrównywanieV` — wyrównywanie zwartości komórki w pionie.

Przy takiej realizacji należy jedynie zwrócić uwagę na właściwe zastosowanie cudzysłowów i apostrofów. Nie możemy stosować ich wymiennie. Jeśli nazwa koloru ujęta będzie w apostrofy (`'kolor1'`), to cała konstrukcja modyfikująca styl powinna znajdować się w cudzysłowie (`"bgColor='kolor1'"`). W sytuacji odwrotnej, czyli kiedy nazwa koloru ujęta zostanie w cudzysłów, konstrukcja zmieniająca styl powinna znajdować się w apostrofie.

Skrypt 10

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<body>
<table border="0" cellpadding="0" cellspacing="0">
<tr>
 <td     onmouseover="bgColor='red'"
         style="cursor:pointer; cursor:hand"
         onmouseout="bgColor='white' "bgcolor="white"
         width="100" align="center" valign="middle">
 <b><font color="gray">Item 1 1</font></b>
 </td>
 <td     onmouseover="bgColor='red'"
         style="cursor:pointer; cursor:hand"
         onmouseout="bgColor='white'" bgcolor="white"
         width="100" align="center" valign="middle">
 <b><font color="gray">Item 1 2</font></b>
 </td>
 <td     onmouseover="bgColor='red'"
         style="cursor:pointer; cursor:hand"
         onmouseout="bgColor='white'" bgcolor="white"
         width="100" align="center" valign="middle">
 <b><font color="gray">Item 1 3</font></b>
 </td>
</tr>
<tr>
 <td     onmouseover="bgColor='pink'"
         style="cursor:pointer; cursor:hand"
```

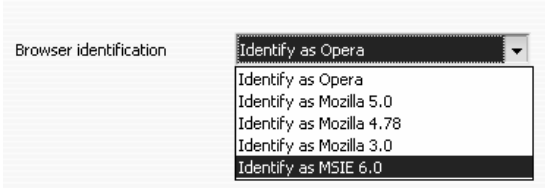
```
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 2 1</font></b>
 </td>
 <td     onmouseover="bgColor='pink'"
        style="cursor:pointer; cursor:hand"
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 2 2</font></b>
 </td>
 <td     onmouseover="bgColor='pink'"
        style="cursor:pointer; cursor:hand"
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 2 3</font></b>
 </td>
</tr>
<tr>
 <td     onmouseover="bgColor='blue'"
        style="cursor:pointer; cursor:hand"
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 3 1</font></b>
 </td>
 <td     onmouseover="bgColor='blue'"
        style="cursor:pointer; cursor:hand"
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 3 2</font></b>
 </td>
 <td     onmouseover="bgColor='blue'"
        style="cursor:pointer; cursor:hand"
        onmouseout="bgColor='white'" bgcolor="white"
        width="100" align="center" valign="middle">
 <b><font color="gray">Item 3 3</font></b>
 </td>
</tr>
</table>
</body>
</html>
```

Skrypt 11. Rozpoznanie rodzaju przeglądarki

[E][F][K][N][O]

Typ przeglądarki możemy rozpoznać, badając ciąg znaków znajdujący się we właściwości `userAgent` obiektu `navigator`. Jest to jedna z bardziej skutecznych metod działająca nawet w przypadku przeglądarek, które umożliwiają zmianę sposobu identyfikacji, jak np. Opera (rysunek 1.9). Liczba dostępnych przeglądarek na rynku jest bardzo duża, skrypt ograniczać się będzie zatem do rozpoznawania najpopularniejszych:

Rysunek 1.9. *Opera potrafi udawać inne przeglądarki*



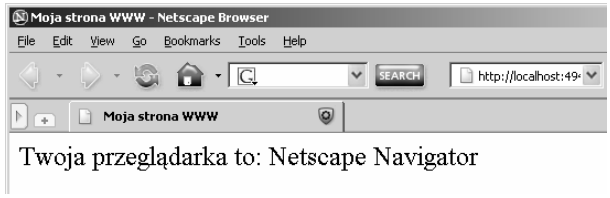
- FireFox,
- Internet Explorer,
- Konqueror,
- Netscape Navigator,
- Opera.

Każda z wymienionych przeglądarek zgłasza we właściwości `userAgent` inny ciąg znaków, który jest zależny od typu i wersji aplikacji. Przykładowo:

```
mozilla/5.0 (windows; u; windows nt 5.0; en-us; rv:1.7.5) gecko/20050729
netscape/8.0.3.3
mozilla/5.0 (compatible; konqueror/3.4; linux) khtml/3.4.0
mozilla/4.0 (compatible; msie 6.0; windows nt 5.0; pl) opera 8.02
mozilla/5.0 (x11; u; linux i686; pl-pl; rv:1.7.8) gecko/20050524 fedora/1.0.4-4
firefox/1.0.4
mozilla/4.0 (compatible; msie 6.0; windows nt 5.0; .net clr 1.1.4322; fdm)
```

Można w nich wyróżnić pewne elementy charakterystyczne. Najłatwiej wyodrębnić po prostu nazwy: *FireFox*, *Konqueror*, *Netscape*, *Opera* i *MSIE*. Do przeszukiwania ciągów służy metoda `indexOf`. Zwraca ona indeks, pod którym dane słowo występuje, lub wartość `-1`, jeśli nie występuje ono w ciągu. Pozwala to zbudować skrypt rozpoznający najpopularniejsze przeglądarki. Przykładowy efekt działania tego skryptu jest widoczny na rysunku 1.10.

Rysunek 1.10. *Identyfikacja typu przeglądarki*



Skrypt 11

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
var agent = navigator.userAgent.toLowerCase();
var nazwa = "nieznany";
```

```
if(agent.indexOf('firefox') != -1){
  nazwa = "FireFox";
}
if(agent.indexOf('opera') != -1){
  nazwa = "Opera";
}
else if(agent.indexOf('konqueror') != -1){
  nazwa = "Konqueror";
}
else if(agent.indexOf('netscape') != -1){
  nazwa = "Netscape Navigator";
}
else if(agent.indexOf('msie') != -1){
  nazwa = "Internet Explorer";
}
// Koniec kodu JavaScript -->
</script>
<body>
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptów
document.write("Twoja przeglądarka to: " + nazwa);
// Koniec kodu JavaScript -->
</script>
</body>
</html>
```

Skrypt 12. Strona zależna od rodzaju przeglądarki [E][F][K][N][O]

Badanie właściwości `userAgent` obiektu `navigator`, np. takie jak w przykładzie 11., umożliwia utworzenie skryptu, który będzie ładował różne wersje strony w zależności od wykrytego typu przeglądarki. Wykrywane będą najpopularniejsze programy: FireFox, Konqueror, Internet Explorer, Netscape Navigator oraz Opera i, w zależności od tego, z którego z nich będzie korzystał użytkownik, wczytywana będzie jedna ze stron: *firefox.html*, *konqueror.html*, *msie.html*, *navigator.html* lub *opera.html*. W przypadku gdyby wykrycie typu przeglądarki się nie powiodło, wczytana zostanie strona *default.html*.

Załadowanie jednej z wymienionych stron będzie wykonywane przez przypisanie jej nazwy do obiektu `windows.location.href`, czyli wczytanie strony np. *msie.html* odbędzie się po wykonaniu instrukcji:

```
window.location.href = 'msie.html'
```

Jeżeli strony zawierające wersje kodu dla poszczególnych przeglądarek będą znajdowały się w różnych katalogach, nazwy tych katalogów muszą być również uwzględnione, np.:

```
window.location.href = 'explorer/msie.html'
```

Istnieje również możliwość podania pełnej, bezwzględnej ścieżki do wskazanych dokumentów, np.:

```
window.location.href = 'http://moja.domena/firefox/firefox.html'
```

Skrypt 12

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
var agent = navigator.userAgent.toLowerCase();
if(agent.indexOf('opera') != -1){
  window.location.href = 'opera.html'
}
else if(agent.indexOf('firefox') != -1){
  window.location.href = 'firefox.html'
}
else if(agent.indexOf('konqueror') != -1){
  window.location.href = 'konqueror.html'
}
else if(agent.indexOf('msie') != -1){
  window.location.href = 'msie.html'
}
else if(agent.indexOf('netscape') != -1){
  window.location.href = 'navigator.html'
}
else{
  window.location.href = 'default.html'
}
// Koniec kodu JavaScript -->
</script>
<body>
</body>
</html>
```

Skrypt 13. Strona zależna od systemu operacyjnego [E][F][K][N][O]

Jeżeli zachodzi konieczność załadowania różnych wersji strony WWW w zależności od używanego przez użytkownika systemu operacyjnego, należy wykorzystać właściwość `userAgent` obiektu `navigator`, podobnie jak miało to miejsce w przypadku skryptów 11. i 12. Zapisany w tej właściwości ciąg znaków zawiera bowiem również określenie systemu, na tej podstawie można więc dokonać rozróżnienia.

Do odszukania słowa *linux* lub *windows* wykorzystujemy metodę `indexOf`. Jeśli dane słowo jest zawarte w ciągu znaków zapisanym we właściwości `userAgent`, jest zwracany indeks jego wystąpienia, w przeciwnym wypadku zwracana jest wartość `-1`. Jeśli odnalezione zostanie słowo *windows*, wczytywana będzie strona *windows.html*, a jeśli będzie to *linux*, użyta zostanie strona *linux.html*.

Skrypt 13

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
var agent = navigator.userAgent.toLowerCase();
if(agent.indexOf('windows') != -1){
  window.location.href = 'windows.html'
}
else if(agent.indexOf('linux') != -1){
  window.location.href = 'linux.html'
}
else{
  window.location.href = 'default.html'
}
// Koniec kodu JavaScript -->
</script>
<body>
</body>
</html>
```

Skrypt 14. Dynamiczne przyciski [E][F][K][N][O]

Często spotykanym efektem jest wymiana znajdującego się na stronie obrazka na inny (po najechaniu na niego kursorem myszy). Istnieje klika metod realizacji takiego zadania, jedną z nich prezentuje skrypt 14. W kodzie HTML umieścić należy obraz za pomocą standardowego znacznika `<img>`. Temu znacznikowi przypisujemy parametr `src` równy *obrazek1_off.gif*. Oznacza to, że po wczytaniu strony do przeglądarki pojawi się na niej obraz zapisany w pliku *obrazek1_off.gif*. Dodatkowo znacznikowi przypisujemy procedury obsługi zdarzeń `onMouseOver` i `onMosueOut`. Pierwsze z nich powstaje, kiedy kursor myszy znajdzie się nad obrazem, a drugie kiedy opuści jego obszar.

W naszym przypadku zdarzenia te są obsługiwane przez funkcje `img_act` oraz `img_deact`. Pierwsza z nich jest odpowiedzialna za zmianę obrazu na zapisany w pliku *obrazek1_on.gif*, a druga na ten zawarty w *obrazek1_off.gif*. Aby jednak skrypt działał prawidło-

wo, oba pliki należy wczytać wcześniej do pamięci. Na samym początku kodu JavaScript definiujemy zatem dwie zmienne — `obrazek1_on` oraz `obrazek1_off` — i przypisujemy im nowo utworzone obiekty typu `Image`:

```
obrazek1_on = new Image(100, 50);
obrazek1_off = new Image(100, 50);
```

W ten sposób powstały dwa puste obrazy o rozdzielczości 100x150 pikseli. Aby do tak utworzonych obiektów została wczytana grafika z plików, modyfikujemy ich (obiektów) właściwości `src`:

```
obrazek1_on.src = "obrazek1_on.gif";
obrazek1_off.src = "obrazek1_off.gif";
```

Od tej chwili obiekty `obrazek1_on` i `obrazek1_off` mogą być wykorzystywane przez funkcje `img_deact` i `img_deact`.

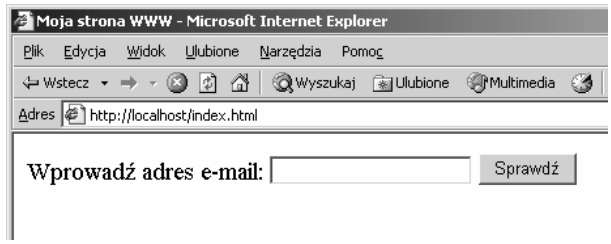
Skrypt 14

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
obrazek1_on = new Image(100, 50);
obrazek1_on.src = "obrazek1_on.gif";
obrazek1_off = new Image(100, 50);
obrazek1_off.src = "obrazek1_off.gif";

function img_act(pic)
{
  document[pic].src = eval(pic + "_on.src");
}
function img_deact(pic)
{
  document[pic].src = eval(pic + "_off.src");
}
// Koniec kodu JavaScript -->
</script>
<body>
<img
  name="obrazek1"
  alt="opis obrazka"
  src="obrazek1_off.gif"
  onMouseOver="img_act('obrazek1')"
  onMouseOut="img_deact('obrazek1')"
>
</body>
</html>
```

Skrypt 15. Walidacja adresu e-mail [E][F][K][N][O]

Wykorzystując JavaScript, możemy w prosty sposób napisać funkcję, której zadaniem będzie weryfikacja wprowadzonego przez użytkownika adresu e-mail. Często się przecież zdarza, że użytkownik myli się lub też wpisuje zupełnie nieprawdopodobne dane. Nie uda nam się co prawda sprawdzić, czy podany e-mail faktycznie istnieje, ale można bez problemu zweryfikować jego formalną poprawność. Utworzymy zatem formularz z polem tekstowym i przyciskiem (rysunek 1.11). Po kliknięciu przycisku zostanie wykonana funkcja `checkEmail`, która wyświetli okno dialogowe z informacją, czy wprowadzone dane składają się na poprawny adres pocztowy, czy też nie.



Rysunek 1.11. *Skrypt weryfikujący poprawność adresu e-mail*

Jak sprawdzić formalną poprawność adresu? Trzeba sobie przypomnieć, jakie obowiązują w nim zasady. Są one następujące:

- w adresie mogą występować jedynie znaki cyfr, liter oraz myślnik, podkreślenie, kropka i @,
- w adresie musi występować dokładnie jeden znak @,
- adres nie może zaczynać się od kropki lub @,
- nazwa domenowa nie może zaczynać się od kropki,
- nazwa domenowa musi składać się z co najmniej dwóch członów,
- ostatni człon nazwy domenowej musi zawierać dokładnie dwie, trzy lub cztery litery.

Badanie, czy wprowadzony przez użytkownika adres spełnia te warunki, można przeprowadzić, wykorzystując serię instrukcji warunkowych, jednak lepszym i zdecydowanie prostszym w realizacji sposobem jest wykorzystanie wyrażeń regularnych. Wystarczy, że zbudujemy wyrażenie opisujące wymienione zasady i wykorzystamy metodę `match`. Wyrażenie takie będzie miało postać:

```
/^[a-z0-9_-]+(\.[a-z0-9_-]+)*@([a-z0-9_-]+)(\.[a-z0-9_-]+)*(\.[a-z]{2,4})$/i
```

Zawarty na jego końcu znak `i` oznacza, że analiza ma być przeprowadza bez uwzględniania wielkości liter.

W funkcji `checkEmail` zostaną zatem wykonane następujące czynności:

1. Wprowadzone przez użytkownika dane zostaną zapisane w pomocniczej zmiennej o nazwie `email`.
2. Wyrażenie regularne zostanie zapisane w zmiennej `re`.
3. Wykonana zostanie metoda `match` obiektu `email` w postaci `email.match(re)`.

Jeśli metoda `match` zwróci wartość `null`, będzie to oznaczać, że adres e-mail nie spełnia warunków opisanych wyrażeniem regularnym zapisanym w zmiennej `re`, a zatem jest nieprawidłowy. W przeciwnym wypadku, czyli kiedy metoda `match` zwróci wartość różną od `null`, adres spełnia warunki opisane wyrażeniem regularnym, więc jest prawidłowy.

Skrypt 15

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function checkEmail()
{
  var email = document.getElementById('email').value;
  var re = /^[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)*@([a-zA-Z0-9_-]+)(\.[a-zA-Z0-9_-
]+)*(\.[a-zA-Z]{2,4})$/i;
  if(email.match(re) == null)
    alert('Ten adres jest nieprawidłowy.');
  else
    alert('Ten adres jest prawidłowy.');
}

// Koniec kodu JavaScript -->
</script>
<body>
Wprowadź adres e-mail:
<input
 type="text"
 id="email"
>
<input
 type="button"
 value = "Sprawdź"
 onClick = "checkEmail()"
>
</body>
</html>
```

Skrypt 16. Antyspam [E][F][K][N][O]

Bardzo często podajemy na stronie adres e-mail. Niestety równie często jest on przechwytywany przez różnego rodzaju programy skanujące sieć w poszukiwaniu danych. Bardzo szybko też zaczynamy dostawać na naszą skrzynkę niechciane listy reklamowe. Na szczęście, przynajmniej w pewnym stopniu, możemy się przed takimi skanującymi robotami bronić, jeśli nie będziemy wpisywać adresu bezpośrednio w kodzie HTML, a zastosujemy funkcje ukrywające jego prawdziwą postać.

Wystarczy, jeśli podzielimy e-mail na części i wyświetlimy go na stronie za pomocą instrukcji `document.write`. Za wykonanie tej czynności odpowiadać będzie funkcja `printEmail`. Jej działanie będzie całkowicie przezroczyste dla użytkownika, adres będzie wyglądał tak, jakby był bezpośrednio wprowadzony w kodzie HTML (rysunek 1.12). Sposób jest dosyć skuteczny, choć niestety nie zadziała w przypadku bardziej zaawansowanych skanerów, które przetwarzają odczytaną stronę, wykonując funkcje JavaScript.

Rysunek 1.12. *Ten adres został wyświetlony za pomocą specjalnej funkcji JavaScript*

Skrypt 16

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
<title>Moja strona WWW</title>
<!-- Skrypt pochodzi z książki "101 praktycznych skryptów"  -->
<!-- autor: Marcin Lis               http://marcinlis.com  -->
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
function printEmail()
{
  var email = "adres" + "@" + "domena" + "." + "com";
  document.write(email)
}
// Koniec kodu JavaScript -->
</script>
<body>
Mój adres to:
<script type="text/javascript">
<!-- Ukrycie przed przeglądarkami nieobsługującymi JavaScriptu
printEmail();
// Koniec kodu JavaScript -->
</script>
</body>
</html>
```