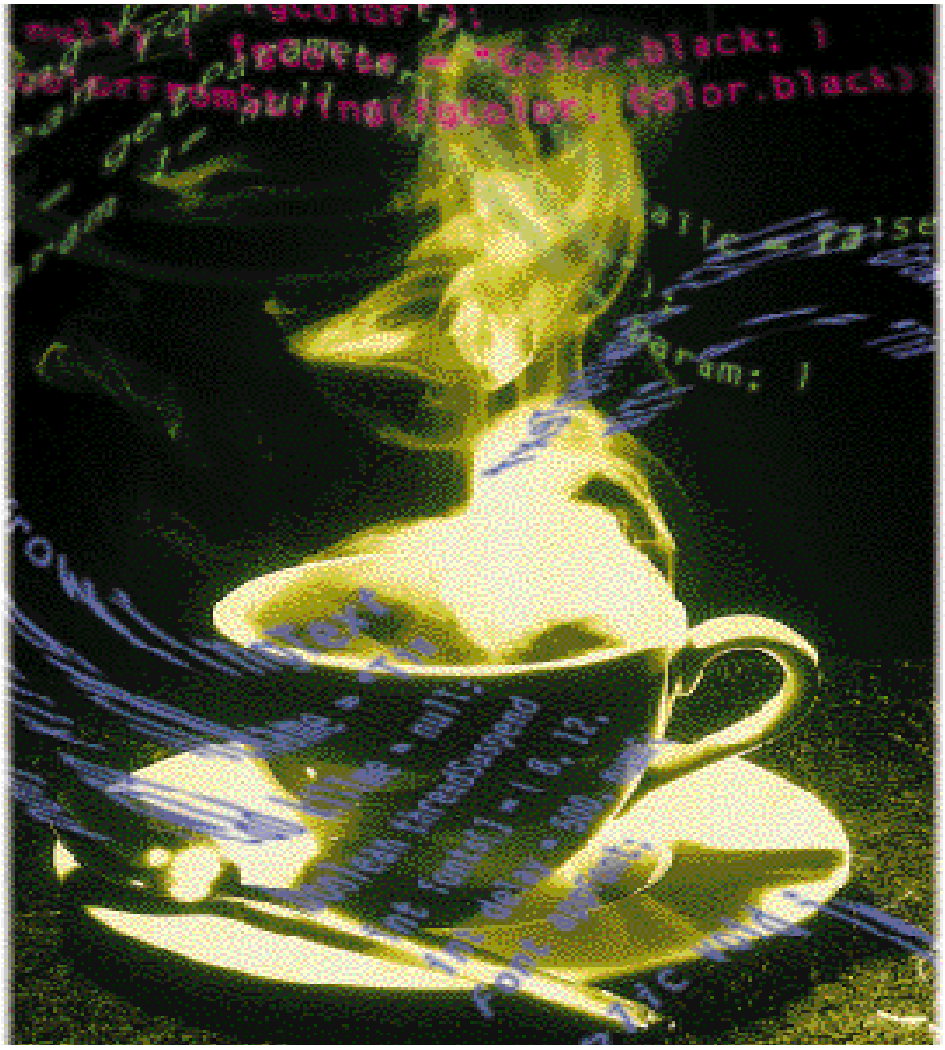


Ramy i ramki

Język JavaScript pozwala skuteczniej niż HTML kontrolować mechanizm ramek (frames). Za jego pomocą nawigację po własnych stronach WWW można zorganizować w bardziej elegancki sposób.

Kolejny odcinek kursu programowania w języku JavaScript poświęcony jest ramkom. Choć w JavaScript zachowują się one niemal tak samo, jak w HTML, istnieje pewna mała, lecz istotna różnica. Używając rozkazu HTML można, klikając hipertekstowy odsyłacz, wyświetlić w innej ramce okna przeglądarki z góry określony plik. Skrypt Javy, reagując na to samo zdarzenie, może spowodować wyświetlenie różnych stron WWW.

Poprawne posługiwanie się ramkami zademonstruje przykład małego leksykonu online. W lewej części okna słownika program wyświetla ramkę, zawierającą cztery hasła zdefiniowane jako odsyłacze hipertekstowe. Kliknięcie któregokolwiek z nich spowoduje pojawienie się w prawej ramce odpowiedniego objaśnienia.

W ramce z listą haseł widnieją jeszcze dwa przyciski nawigacyjne. Za ich pomocą można „kartkować" leksykon, przemieszczając się w przód, czyli od hasła bieżącego do następnego lub wstecz – do poprzedniego. Do zdefiniowania ramek i wyświetlenia objaśnień do haseł wystarczą polecenia HTML. JavaScript potwierdza swą przydatność dopiero w przyciskach nawigacyjnych. Skorzystamy z tego, by dowiedzieć się czegoś o tablicach zmiennych (**array**) i o obiekcie **location**.

Tworzenie ramek: plus dla HTML

Właściwości i atrybuty, za pomocą których definiuje się ramki w HTML, odpowiadają analogicznym właściwościom i atrybutom z JavaScript. Czytelnicy zainteresowani programowaniem stron Pajęczyny przypuszczalnie mieli już kontakt z poświęconym tej tematyce zeszytem specjalnym CHIP-a zatytułowanym „*HTML i Java*". Począwszy od strony 32 opisano tam definiowanie i używanie ramek. Krótkie zestawienie wiadomości na ten temat znajduje się także w notce „*Tworzenie ramek w HTML*".

Do budowy ramek leksykonu użyjemy HTML. Oto wydruk pliku **indeks.html**:

```
<html>
<frameset cols "20%,*"
border=2>
<frame src="menu.html"
name="hasla">
<frame src="tytuł.html"
name="objasnienie"
</frameset>
```

Definiujemy dwie ramki: pierwsza nosi nazwę **hasla**, zdefiniowaną przy pomocy atrybutu **name="hasla"**, i wyświetla plik **menu.html**. Druga ramka nazywa się **objasnienie**, a związana jest z plikiem **tytuł.html**. Ten z kolei wita przybyłego surfera oraz objaśnia mu, jak posługiwać się paskiem menu z hasłami. W naszym przykładzie jest to prosty plik HTML zawierający tekstowy opis. Polecenie:

```
border=2
```

ustala grubość paska rozdzielającego ramki na dwa piksele.

Plik **menu.html** jest najważniejszy: wyświetla zestawienie dostępnych w leksykonie haseł. Hasła te zdefiniowano jako odsyłacze. Na obrazie ekranu znajdującego się na sąsiedniej stronie widać również przyciski do przeglądania leksykonu w przód i wstecz. Zrealizowano je za pomocą formularzy. Korzystanie z nich omówiono w poprzedniej, trzeciej części warsztatu JavaScript (patrz CHIP 7/97).

Pierwszy odsyłacz, wskazujący na hasło „abnegacja", zdefiniowano następująco:

```
<a href="abnegacja.html"
target="objasnienie"
onClick="ustawStrone(1)">
abnegacja</a><br>
```

▶ 101

Workshop w skrócie

CHIP 5/97: podstawy JavaScript, wyświetlanie tekstu na stronie WWW, funkcja daty
CHIP 6/97: funkcje, zmienne, formularze, funkcja eval()
CHIP 7/97: obliczenia naukowe z wykorzystaniem obiektu math
CHIP 8/97: praca z ramkami
CHIP 9/97: wykorzystanie cookies, rozważania nad przyszłością JavaScript

Plik, który powinien być wyświetlony po kliknięciu odsyłacza, nazywa się **abnegacja.html**. Pojawi się on w ramce, która początkowo zawiera powitanie. Odpowiedzialna jest za to nazwa **objasnienie**: ponieważ podano ją przy atrybucie **target**, cały wygenerowany obraz zostanie przekazany do ramki o tej właśnie nazwie. Trzeci atrybut
onClick="ustawStrone(1)"
powoduje, że po kliknięciu myszą odsyłacza zostanie wykonana funkcja JavaScriptu o nazwie **ustawStrone()**, po uprzednim przekazaniu jej jako argumentu liczby 1.

W podobny sposób skonstruowano pozostałe odsyłacze: we wszystkich jako ramkę docelową (**target**) podano **objasnienie**, w każdym jest także uaktywniana funkcja **ustawStrone()**.

Przyciski do przeglądania: plus dla JavaScript

Przyciski do przeglądania leksykonu w przód i wstecz sporządzono jako formularze:

```
<form>
<input type=button
name="wstecz"
value=" < "
onClick="idzWstecz()">
<input type=button
name="naprzód"
value=" > "
onClick="idzNaprzod()">
</form>
```

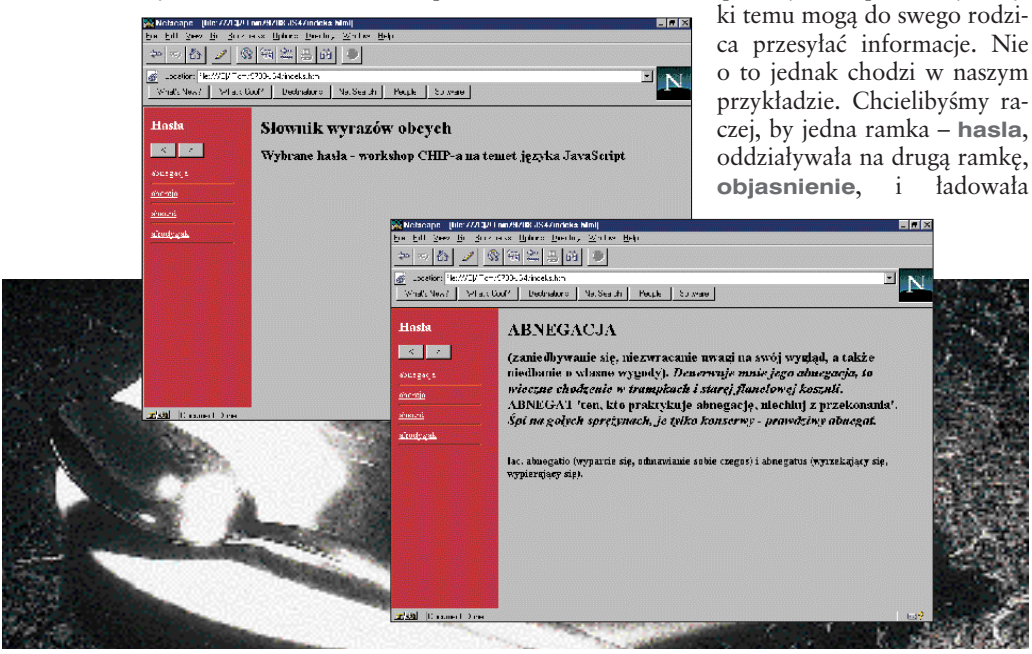
Przyciski nazywają się **wstecz** i **naprzód**. Atrybut **value** definiuje, co będzie widać na ich powierzchni. Szerokość przycisków określa się, wstawiając odpowiednią liczbę spacji w tym samym atrybucie. Tutaj też do głosu dochodzi JavaScript. Przyciski powiązane są z funkcjami reagującymi na kliknięcie myszą. Przycisk do przeglądania do przodu uaktywnia metodę **idzNaprzod()**, przycisk do przeglądania wstecz – **idzWstecz()**. Uwaga na osobliwość JavaScriptu: w definicjach oraz w wywołaniach obu metod muszą być podane nawiasy, chociaż nie są przy tym przekazywane żadne parametry.

Dopiero teraz w pełnym świetle widać przewagę języka skryptów nad HTML: JavaScript pozwala na sterowanie obcymi ramkami. W tym celu w metodzie **idzNaprzod()** umieszczamy polecenia:

```
function idzNaprzod() {
 if (strona < 4) {
  strona = strona + 1;
  parent.objasnienie.location.href
  = tekst[strona];
 }
}
```

JavaScript traktuje ramkę jako obiekt istniejący w określonej hierarchii. Plik tworzący ramkę określa się mianem „rodzica" (**parent**). W rozpatrywanym przykładzie jest to plik **indeks.html** z etykietą **<frameset>**. Generowane przez niego ramki, **hasla** i **objasnienie**, są jego potomkami (child). Między ramką rodzicielską a potomną istnieje bezpośredni związek, pozwalający im – za pośrednictwem JavaScript – wymieniać między sobą informacje.

Przyciski w pliku **menu.html** należą również do pewnej ramki potomnej i dzięki temu mogą do swego rodzica przesyłać informacje. Nie o to jednak chodzi w naszym przykładzie. Chcielibyśmy raczej, by jedna ramka – **hasla**, oddziaływała na drugą ramkę, **objasnienie**, i ładowała do niej nowy plik HTML, wyświetlający objaśnienie wybranego hasła. Wiemy, że między ramkami potomnymi nie ma bezpośredniego związku. Cóż więc można zrobić?



Ramki w akcji: dzięki przyciskom nawigacyjnym w przykładowej aplikacji wygodnie żeglujemy wśród stronic leksykonu

Sformułowanie „nie ma bezpośredniego związku" sugeruje istnienie związku pośredniego. Skoro ramka rodzicielska powiązana jest z obiema ramkami potomnymi jednocześnie, komunikacja między tymi ostatnimi przebiegać może poprzez ramkę-rodzica. Właśnie w tym celu definicje obiektów potomnych zawierają atrybuty **name**. Odwołania do nich będą realizowane za pośrednictwem zdefiniowanych w ten sposób nazw.

W odwołaniach do ramek korzysta się z powszechnie przyjętej notacji kropkowej, odzwierciedlającej hierarchię obiektów JavaScript. W naszym przykładzie cofamy się do ramki rodzicielskiej, dostęp do której uzyskać można pisząc po prostu **parent** lub **top**, i stąd – oddzieliwszy ją kropką – idziemy do drugiej ramki potomnej:
parent.objasnienie

▶ 102

Tablice w JavaScript

Tablica – pojęcie znane z pewnością z innych języków programowania – to zbiór zmiennych o tej samej nazwie. By można było sięgać do poszczególnych elementów tablicy, każdy z nich scharakteryzowany jest unikatową wartością zwaną indeksem.

JavaScript traktuje tablice jak obiekty. Dlatego też tworzone są one poleceniem **new**, po którym następuje typ obiektu. W naszym przykładzie leksykonu tablica zmiennych **tekst** została utworzona poleceniem:

```
tekst = new Array;
```

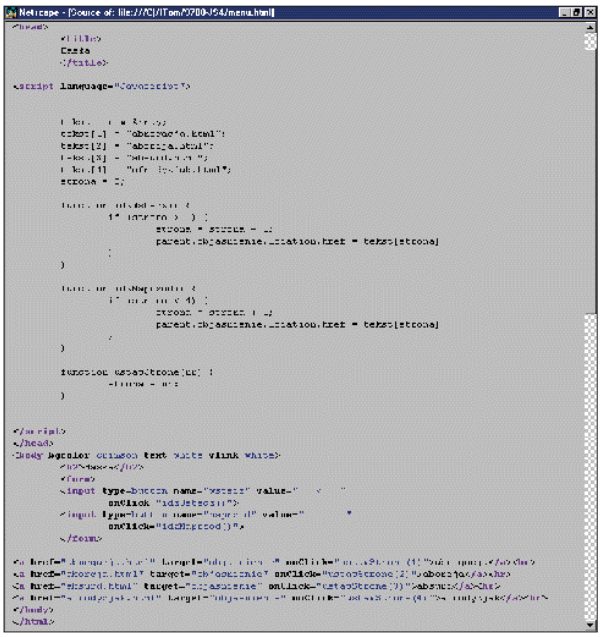
Nie trzeba przy tym obowiązkowo – jak to ma miejsce w wielu innych językach – podawać liczby potrzebnych elementów, gdyż JavaScript automatycznie dodaje nowy element do tablicy za każdym razem, gdy zostaje on użyty. Zwykle po zadeklarowaniu tablicy poszczególnym jej elementom przypisywane są wartości. Przypisanie jest dokonywane poprzez umieszczony w nawiasie klamrowym indeks. Na przykład polecenie:

```
tekst[1] = "abnegacja.html"
```

nadaje pierwszemu elementowi tablicy (o indeksie 1) wartość wyrażoną łańcuchem znakowym "abnegacja.html".
W przedstawionym przypadku każdy element tablicy otrzymuje jako wartość łańcuch znakowy. Generalnie jednak elementy tablicy mogą w JavaScript mieć rozmaite typy danych. Nic nie stoi na przeszkodzie, by kolejny element, **tekst[2]**, przechowywał wartość całkowitą.

Po tej części nazwy obiektu pojawia się w wywołaniu kolejna kropka, a po niej **location**. Location oznacza nowy typ obiektu. Każde okno przeglądarki i każda ramka posiadają własny obiekt **location**. Za pośrednictwem szeregu właściwości zapewnia on dostęp do strony załadowanej w danej chwili do okna lub ramki. Wśród właściwości tych najważniejszą jest **href**. Określa ona łańcuch znakowy, odpowiadający identyfikatorowi URL załadowanej strony HTML. Inne właściwości, np. **hostname** lub **protocol**, zawierają poszczególne części tego łańcucha. Właściwość **href** można nie tylko odczytać – można jej także przypisać wartość, czyli dowolny inny URL. Wówczas do związanego z **location** okna lub ramki zostanie załadowany plik HTML, na który wskazuje przypisany łańcuch. Kompletny wiersz programu, służący do odwoływania się do drugiej ramki, wygląda zatem następująco:

```
parent.objasnienie.location.href =
tekst[strona]
```



Plik menu.html: kompletny listing ramki nawigacyjnej leksykonu

Ramce **objasnienie** przypisujemy nazwę pliku umieszczoną w zmiennej **tekst[strona]**. Tym samym zapoznaliśmy się z nową, bardzo przydatną cechą JavaScript – tablicami zmiennych (**array**). Warto także przeczytać notkę „*Tablice w JavaScript*” na następnej stronie. W nagłówku pliku **menu.html** zadeklarowano tablicę zmiennych o nazwie **tekst**, której czterem elementom przypisano następnie łańcuchy znakowe, przechowujące nazwy plików HTML z objaśnieniami leksykalnymi.

```
tekst = new Array;
tekst[1] = ”abnegacja.html”;
tekst[2] = ”aborcja.html”;
tekst[3] = ”absurd.html”;
tekst[4] = ”afrodyzjak.html”;
strona = 0;
```

W ostatnim wierszu fragmentu pojawia się zmienna całkowita – **strona**. **Strona** przechowuje indeks nazwy pliku załadowanego do ramki **objasnienie** – jednego z tych, których nazwy zostały wcześniej umieszczone w tablicy **tekst[]**. Indeks ten jest na wstępie metody **idzNaprzod()** powiększany o jednostkę, co na poziomie programu odpowiada przejściu do następnego hasła leksykonu, czyli do przodu. Analogicznie, w metodzie **idzWstecz()** zmienna **strona** jest zmniejszana o 1. Dopiero potem następuje załadowanie wskazanego przez nią pliku do ramki.

Powyższy sposób użycia zmiennej **strona** wymaga, by przed pierwszym wywołaniem przypisano jej wartość 0. Przypuśćmy bowiem, że użytkownik załaduje stronę tytułową i od razu kliknie

Tworzenie ramek w HTML

Przy pomocy ramek (**frame**) można podzielić okno przeglądarki WWW i w każdej z części wyświetlić niezależnie inny plik HTML. Aby utworzyć stronę WWW z dwiema ramkami, potrzebne są trzy pliki HTML. Pierwszy z nich, tutaj **indeks.html**, definiuje podział okna, sam jednak nie jest wyświetlany. To do tego pliku trzeba się odwołać, by zobaczyć całą podzieloną stronę. Dwa pozostałe pliki stanowią zawartość obu ramek. Plik **indeks.html** wygląda następująco:

```
<html>
<frameset cols=”12%,88%”>
<frame src=”menu.html”>
<frame src=”tytuł.html”>
</frameset>
</html>
```

indeks.html definiowany jest z pomocą etykiety **<html>** jako standardowa strona HTML z tą jednak różnicą, że nie wymaga etykiet **<head>** i **<body>**. Istotna jest etykieta **<frameset>** rozpoczynająca definicję ramki, którą później zakończy **</frameset>**.

Dopasowywanie

Etykieta **<frameset>** określa także, ile miejsca zajmą poszczególne ramki. Służą do tego właściwości (atrybuty) **cols** i **rows**. Polecenie:

```
<frameset cols=”12%,88%”>
```

mówi, że okna mają utworzyć pionowe kolumny (**cols** oznacza „columns”, czyli kolumny), i że pierwsza, lewa ramka powinna zająć 12% całej szerokości okna. Wartość 88% definiuje szerokość drugiej ramki. Gdybyśmy chcieli podzielić okno na dwie poziome ramki, etykieta **<frameset>** musiałaby mieć postać:

```
<frameset rows=”12%,88%”>
```

(**rows** oznacza „wiersze”). Prócz określania względnej wielkości ramki w procentach, możliwe jest także podawanie wymiarów bezwzględnych w pikselach.
Obliczenie wielkości drugiej ramki można też powierzyć przeglądarce. W tym celu określamy jedynie szerokość dla pierwszej ramki, zaś drugą wartość podajemy jako ”*”. Gdyby na przykład lewa ramka miała być szeroka na 100 pikseli, a prawa obliczana automatycznie, wpisalibyśmy:

```
<frameset rows=”100,*”>
```

Opcja ta funkcjonuje również dla procentów:

```
<frameset rows=”12%,*”>
```

Taki sam podział uzyskamy pisząc:

```
<frameset rows=”*,88%”>
```

Pamiętaj o zawartości

Teraz trzeba podać nazwy plików, które wypełnią ramki treścią. Definiuje się je kolejno za pomocą etykiety **<frame>**:

```
<frame src = ”menu.html”>
<frame src = ”tytuł.html”>
```

Powyższe wiersze przypisują pierwszej ramce plik **menu.html**, zaś drugiej – **tytuł.html**. Kolejność wierszy odpowiada przy tym kolejności, w jakiej wielkości ramek zdefiniowano w etykiecie **<frameset>**. W naszym przykładzie leksykonu, **menu.html** zawiera pasek nawigacyjny, natomiast **tytuł.html** wyświetla stronę tytułową w momencie pierwszego wywołania strony WWW.

przycisk do kartkowania w przód; wówczas miejsce tytułu musi zająć objaśnienie pierwszego hasła. Jednak nazwa pliku z pierwszym hasłem znajduje się w elemencie tablicy **tekst[1]**, zatem inkrementacja musi w tym wypadku dać wartość 1. Indeksom tablic można zresztą nadawać dowolne zakresy wartości, na przykład od 5 do 8 lub od -110 do -107. Trzeba tylko odpowiednio dopasować wartość początkową zmiennej **strona**. Pod tym względem JavaScript jest bardzo elastyczny.

Co jednak zrobić, gdy **strona** ma wprawdzie wartość 0, lecz użytkownik kliknie przycisk do przeglądania wstecz? Zmienna **strona** przyjmie wtedy wartość -1, a zmienna **tekst[-1]** oczywiście nie istnieje. Podobny problem pojawia się, gdy **strona** osiągnie wartość 4 i użytkownik ponownie użyje przycisku **naprzód** – przeglądarka wyświetli wtedy mało elegancki komunikat o błędzie.

Trudność tę rozwiąże konstrukcja **if**. W metodzie **idzWstecz()** dekrementacja, czyli zmniejszenie numeru strony o 1, a następnie wyświetlenie zdefiniowanego w **tekst[strona]** pliku HTML wykonywane są tylko wtedy, gdy numer strony jest większy od 1. Analogicznie, w metodzie **idzNaprzod()** zmienna ta jest zwiększana o 1 tylko wówczas, gdy **strona** nie osiągnęła jeszcze wartości 4. Zatem warunki **if** troszczą się o to, by przy kliknięciu, które wywołałoby nie istniejącą zmienną **tekst[]**, nic się nie stało.

Metoda ustawStrone()

Użytkownik może przeglądać kolejne hasła pojedynczo, korzystając z przycisków nawigacyjnych, może też przeskakiwać do nich bezpośrednio, używając hipertekstowych odsyłaczy. Gdyby skorzystał z tego ostatniego sposobu, należy uaktualnić wartość zmiennej **strona**, by po takim skoku można było nadal korzystać z obu przycisków do kartkowania. Służy do tego trzecia metoda – **ustawStrone()**.

Jest ona dodawana za pomocą atrybutu **onClick** w trakcie definiowania odsyłaczy. Liczba przekazywana jej jako argument odpowiada wartości indeksu określającego położenie nazwy związanego z odsyłaczem pliku HTML w tablicy **tekst[]**. Następnie w funkcji **ustawStrone()** zmiennej **strona** przypisywana jest przekazana w ten sposób wartość (czyli numer). Dzięki temu wartość zmiennej **strona** zawsze odpowiada numerowi wyświetlanej strony.

Ostatnia strona: „Dalej się nie da”

Żeby po dojściu do pierwszej lub ostatniej strony leksykonu kartkowanie – odpowiednio wstecz lub naprzód – nie skończyło się, możemy tak zmodyfikować listing programu, by przy przeglądaniu do przodu po ostatniej stronie leksykonu pojawiła się pierwsza – i na odwrót. Alternatywnie można wstawić komunikat z ostrzeżeniem w rodzaju „To była ostatnia strona. Dalej już nic nie ma”. Z pewnością Czytelnik potrafi już teraz samodzielnie dokonać odpowiednich zmian.

Przedstawiony tu prosty przykład leksykonu demonstruje jedynie postawy tego, co można osiągnąć za pomocą JavaScript. Ów język skryptów umożliwia także sterowanie zagnieżdżonymi ramkami w sposób podobny do pokazanego powyżej. Trzeba go jedynie dostosować do istniejącej w danym przypadku hierarchii ramek i utworzyć odpowiednie, czasem wielopoziomowe, odwołania.

W następnej i ostatniej części kursu dowiemy się nieco na temat złowieszczych „cookies” oraz perspektyw rozwoju języka JavaScript.

Tomasz Czarnecki (pk)

JavaScript w literaturze

Języki JavaScript i HTML omawiają następujące książki:

CHIP-Special: HTML i Java, Vogel Publishing, Wrocław 1997
P.J.Perry: **JAVA. Tworzenie appletów WWW**, Oficyna Wydawnicza Read Me, Warszawa 1997
S.J.Walter, A. Weiss: **Język JavaScript nie tylko dla orłów**, Intersoftland, Warszawa 1996
S.Lalani, K.Jamsa: **Java biblioteka programisty**, Zakład Nauczania Informatyki „MIKOM”, Warszawa 1997
J.Castro: **Po prostu HTML**, Wydawnictwo Helion, Gliwice 1996
W.Macewicz: **HTML – język opisu dokumentu hipertekstowego**, Zakład Nauczania Informatyki „MIKOM”, Warszawa 1996