

Tom Negrino, Dori Smith

→ PO PROSTU

Wydanie VIII

JavaScript

Potencjał JavaScriptu
w zasięgu ręki!



Helion



Tytuł oryginału: JavaScript: Visual QuickStart Guide (8th Edition)

Tłumaczenie: Piotr Pilch

na podstawie „Po prostu JavaScript i Ajax. Wydanie VII” w tłumaczeniu Wojciecha Mocha

Projekt okładki: Maciej Pasek

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

ISBN: 978-83-246-4271-7

Authorized translation from the English language edition, entitled: JAVASCRIPT: VISUAL QUICKSTART GUIDE, Eighth Edition; ISBN 0321772970; by Tom Negrino, and Dori Smith; published by Pearson Education, Inc, publishing as Peachpit Press. Copyright © 2012 by Tom Negrino and Dori Smith.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education Inc.

Polish language edition published by Helion S.A. Copyright © 2012.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/ppjsc8>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem: <ftp://ftp.helion.pl/przyklady/ppjsc8.zip>

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

	Wprowadzenie	II
Rozdział 1.	Pierwsze spotkanie z JavaScriptem	17
	Czym jest JavaScript	18
	JavaScript to nie Java	19
	Skąd się wziął język JavaScript	21
	Co potrafi JavaScript	22
	Czego JavaScript nie zrobi	23
	Czym jest Ajax	24
	Język obiektowy	27
	Obsługa zdarzeń	30
	Wartości i zmienne	31
	Tworzenie kodu HTML dla JavaScriptu	33
	Potrzebne narzędzia	36
Rozdział 2.	Zaczynamy!	37
	Gdzie umieszczać skrypty	39
	Kilka słów o funkcjach	41
	Stosowanie zewnętrznych skryptów	42
	Wstawianie komentarzy do skryptów	45
	Komunikaty dla użytkownika	47
	Potwierdzanie wyboru dokonanego przez użytkownika	49
	Pobieranie tekstu od użytkownika	51
	Przekierowanie użytkownika za pomocą łącza	53
	Stosowanie JavaScriptu do rozbudowy łączy	55
	Używanie wielopoziomowych instrukcji warunkowych	59
	Obsługa błędów	62
Rozdział 3.	Podstawy języka	65
	W kółko, w pętli	66
	Przekazywanie wartości do funkcji	71
	Wykrywanie obiektów	73
	Praca z tablicami	75

	Praca z funkcjami zwracającymi wartość	77
	Aktualizowanie tablic	78
	Stosowanie pętli do/while	80
	Wywoływanie skryptu na kilka różnych sposobów	82
	Łączenie JavaScriptu i CSS	84
	Sprawdzanie stanu	87
	Praca z tablicami ciągów znaków	93
Rozdział 4.	Praca z obrazami	97
	Podmienianie obrazki	99
	Lepsza technika podmiany obrazków	101
	Tworzenie przycisków trójstanowych	107
	Podmiana obrazków poprzez łącze	109
	Podmienianie obrazka z różnych łączy	112
	Podmienianie wielu obrazków z jednego łącza	115
	Tworzenie animowanych banerów	120
	Dodawanie łączy do animowanych banerów	122
	Prezentacje	124
	Losowe wyświetlanie obrazków	127
	Cykliczna zmiana obrazów z losowym obrazem początkowym	129
Rozdział 5.	Okna i ramki	131
	Zapobieganie wyświetleniu strony w ramce	133
	Konfigurowanie elementu docelowego	134
	Ładowanie ramek iframe za pomocą JavaScriptu	136
	Praca z ramkami iframe	137
	Tworzenie dynamicznych elementów iframe	139
	Funkcje wspólne dla kilku dokumentów	141
	Otwieranie nowego okna	143
	Zmiana zawartości nowego okna	147
Rozdział 6.	Obsługa formularzy	149
	Nawigacja „wybierz i przejdź”	151
	Dynamiczne modyfikowanie menu	156
	Tworzenie pól wymaganych	158
	Wzajemne sprawdzanie wartości pól	163
	Wyróżnianie problematycznych pól	165
	Praktyczne wykorzystanie kontroli formularzy	167
	Praca z przyciskami opcji	171
	Wzajemne ustawianie wartości pól	174
	Sprawdzanie kodów pocztowych	177
	Sprawdzanie adresów e-mail	181

Rozdział 7.	Formularze i wyrażenia regularne	187
	Sprawdzanie adresów e-mail za pomocą wyrażeń regularnych	189
	Sprawdzanie nazwy pliku	194
	Wydobywanie ciągów znaków	196
	Formatowanie ciągów znaków	199
	Formatowanie i sortowanie ciągów znaków	203
	Formatowanie i sprawdzanie poprawności ciągów znaków	205
	Podmiana elementów za pomocą wyrażenia regularnego	208
Rozdział 8.	Obsługa zdarzeń	211
	Obsługa zdarzeń okien	212
	Obsługa zdarzeń myszy	220
	Obsługa zdarzeń formularzy	228
	Obsługa zdarzeń klawiatury	232
Rozdział 9.	JavaScript i ciasteczka	235
	Pieczemy pierwsze ciasteczko	237
	Odczytywanie ciasteczka	241
	Wyświetlanie ciasteczek	242
	Wykorzystanie ciasteczek jako liczników	244
	Usuwanie ciasteczek	247
	Obsługa wielu ciasteczek	249
	Informowanie o nowościach na stronie	251
Rozdział 10.	Obiekty i model DOM	257
	Kilka słów o manipulacji węzłami	258
	Dodawanie węzłów	260
	Usuwanie węzłów	262
	Usuwanie określonego węzła	264
	Wstawianie węzłów	267
	Podmiana węzłów	270
	Tworzenie kodu za pomocą literalów obiektów	273
Rozdział 11.	Tworzenie dynamicznych stron	279
	Wpisywanie aktualnej daty na stronie WWW	280
	Manipulowanie dniami	282
	Dostosowywanie wiadomości do pory dnia	283
	Wyświetlanie dat według strefy czasowej	284
	Konwersja czasu 24-godzinny na 12-godzinny	290
	Odliczanie	292
	Wyświetlanie i ukrywanie warstw	296
	Przenoszenie obiektu w dokumencie	299
	Metody obiektu Date	301

Rozdział 12.	JavaScript w akcji	303
	Stosowanie wysuwanych menu	304
	Dodawanie menu rozwijanych	307
	Rozbudowa menu rozwijanych	311
	Pokaz slajdów z podpisami	315
	Generator dziwnych imion	319
	Generator wykresów słupkowych	324
	Podmiany arkuszy stylów	333
Rozdział 13.	Wprowadzenie do technologii Ajax	343
	Ajax: o co tu chodzi	345
	Odczytywanie danych z serwera	349
	Analizowanie danych z serwera	357
	Odświeżanie danych z serwera	364
	Pobieranie danych z serwera	367
	Podgląd łączy w technologii Ajax	371
	Automatyczne uzupełnienie pól formularza	375
Rozdział 14.	Zestawy narzędziowe, szkielety i biblioteki	381
	Dodawanie biblioteki jQuery	383
	Aktualizowanie strony przy użyciu kodu jQuery	386
	Interakcja z biblioteką jQuery	387
	Interakcja i aktualizowanie	389
	Automatycznie uzupełniane pola	392
Rozdział 15.	Tworzenie stron przy użyciu biblioteki jQuery	395
	Wyróżnianie nowych elementów	396
	Budowanie menu harmonijkowych	400
	Tworzenie sprytnych okien dialogowych	404
	Pasiaste tabele	408
	Sortowanie tabel	411
Rozdział 16.	Tworzenie stron przy użyciu biblioteki jQuery	415
	Zastosowanie biblioteki jQuery w roli fundamentu	416
	Użycie narzędzia ThemeRoller do dostosowywania wyglądu witryny	418
	Dodawanie kalendarza do strony	421
	Przeciąganie i upuszczanie elementów	426
	Użycie biblioteki jQuery z danymi zewnętrznymi	429
	Zastosowanie dodatków biblioteki jQuery	438

Rozdział 17.	Skryptozakładki	445
	Pierwsza skryptozakładka	446
	Zmiana koloru tła strony	452
	Zmiana stylów strony	453
	Wyszukiwanie słów	456
	Przeglądanie obrazków	459
	Wyświetlanie znaków z zestawu ISO Latin	461
	Konwersja wartości RGB do postaci szesnastkowej	464
	Konwersja wartości	466
	Kalkulator skryptozakładkowy	467
	Skracanie adresów URL	469
	Sprawdzanie poprawności stron	470
	Wysyłanie stron e-mailem	471
	Zmiana wielkości stron	472
Dodatek A	JavaScript — genealogia i kompendium	473
	Wersje JavaScriptu	474
	ECMAScript	477
	Diagram obiektów	479
	Wielka tabela obiektów	485
Dodatek B	Słowa kluczowe języka JavaScript	497
Dodatek C	Kaskadowe arkusze stylów	501
Dodatek D	Gdzie można dowiedzieć się więcej	509
	Znajdowanie pomocy w sieci	510
	Tradycyjne źródła informacji	513
	Wskazówki dotyczące rozwiązywania problemów	514
	Skorowidz	517

W tym rozdziale:

Podmieniane obrazki	99
Lepsza technika podmiany obrazków	101
Tworzenie przycisków trójstanowych	107
Podmiana obrazków poprzez łącze	109
Podmienianie obrazka z różnych łączy	112
Podmienianie wielu obrazków z jednego łącza	115
Tworzenie animowanych banerów	120
Dodawanie łączy do animowanych banerów	122
Prezentacje	124
Losowe wyświetlanie obrazków	127
Cykliczna zmiana obrazów z losowym obrazem początkowym	129

Jednym z najciekawszych zastosowań JavaScriptu jest ożywanie stron przy użyciu animowanej grafiki. Temu właśnie poświęcimy ten rozdział. Obrazek na stronie zmieniający się w chwili wskazania go myszą — co sprawia, że strona niejako reaguje na czynności podejmowane przez użytkownika — to jedna z najbardziej popularnych i efektywnych metod wykorzystania JavaScriptu. *Podmieniany obrazek* (ang. *rollover*) jest łatwy do utworzenia, a przy tym, co za chwilę zademonstrujemy, można go wykorzystać na wiele sposobów.

Podmieniane obrazki to bardzo przydatne narzędzie, ale JavaScriptu można także użyć do tworzenia obrazków zmieniających się automatycznie lub do opracowania animowanych banerów reklamowych, pokazów slajdów, a nawet wyświetlania na stronie losowo wybieranych obrazków.

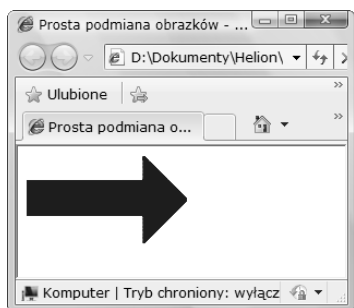
W tym rozdziale napiszemy, jak można wprowadzić na strony różne sztuczki z obrazami, wykonywane za pomocą języka JavaScript. Zabieramy się do pracy!

Tabela 4.1. Podstawy HTML — obrazy

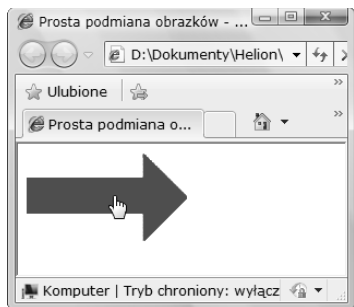
Znacznik	Atrybut	Znaczenie
img		Gromadzi atrybuty opisujące obrazek, jaki ma zostać wyświetlony w przeglądarce.
	src	Zawiera adres URL obrazka, względny w stosunku do adresu URL strony.
	width	Opisuje szerokość (w pikselach), jaką obrazek ma mieć po wyświetleniu w przeglądarce.
	height	Opisuje wysokość (w pikselach), jaką obrazek ma mieć po wyświetleniu w przeglądarce.
	alt	Tekst stosowany w przeglądarkach niewizualnych, który zastępuje sam obrazek.
	id	Jednoznaczny identyfikator pozwalający skryptom JavaScript na wprowadzanie modyfikacji do obrazka.

Skrypt 4.1. Najprostszy sposób utworzenia animowanego przycisku menu w znaczniku łącza

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Prosta podmiana obrazków</title>
  <link rel="stylesheet"
href="script01.css">
</head>
<body>
  <a href="next.html"
onmouseover="document.images['arrow'].src='i
mages/arrow_on.gif'"
onmouseout="document.images['arrow'].src='im
ages/arrow_off.gif'"></a>
</body>
</html>
```



Rysunek 4.1. Pierwszy obrazek przed wskazaniem go myszką



Rysunek 4.2. Po wskazaniu obrazka myszką skrypt powoduje podmianę obrazków

Podmieniane obrazki

Technika podmieniania obrazków jest bardzo prosta. Potrzebne są dwa obrazki. Pierwszy (*oryginalny*) z nich pobierany jest i wyświetlany wraz z całą stroną. Kiedy użytkownik wskazuje myszą pierwszy obrazek, przeglądarka szybko podmienia wyświetlany obrazek na *obrazek zmiennik*, tworząc złudzenie ruchu lub animacji.

W skrypcie 4.1 prezentujemy podstawy techniki podmieniania obrazków. Całość oparta jest na standardowych odsyłaczach do obrazków. Najpierw ładowana jest niebieska strzałka (rysunek 4.1), która w chwili wskazania jej myszą podmieniana jest na strzałkę czerwoną (rysunek 4.2). Po odsunięciu kursora myszy ponownie wyświetlona zostaje niebieska strzałka.

W elementach na stronie zastosowano kilka stylów. Style te umieszczono w osobnym pliku CSS (skrypt 4.2).

Skrypt 4.2. Ten plik CSS definiuje style elementów w wielu przykładach zamieszczonych w rozdziale

```
Skrypt
body {
  background-color: #FFF;
}

img {
  border-width: 0;
}

img#arrow, img#arrowImg {
  width: 147px;
  height: 82px;
}

#button1, #button2 {
  width: 113px;
  height: 33px;
}

.centered {
  text-align: center;
}

#adBanner {
  width: 400px;
  height: 75px;
}
```

Oto tworzenie podmienianych obrazków:

I. `<a href="next.html"`

Łącze rozpoczyna się od określenia adresu, do którego ma się udać przeglądarka w chwili, gdy użytkownik kliknie obrazek. W tym przypadku jest to strona `next.html`.

```
onmouseover="document.images['arrow'].src=
↳ 'images/arrow_on.gif'"
```

W chwili wskazania strzałki myszą (atrybut `src` identyfikatora `arrow`) w dokumencie zostaje wyświetlony obrazek zmiennik `arrow_on.gif`, który znajduje się w katalogu `images`.

```
onmouseout="document.images['arrow'].src=
↳ 'images/arrow_off.gif'">
```

Po odsunięciu wskaźnika myszy ponownie wyświetlany jest obrazek `arrow_off.gif`.

```

```

Pozostała część łącza definiuje źródło oryginalnego obrazka na stronie. Znacznik obrazka uzupełniliśmy atrybutem `alt` (określa on opis obrazka wykorzystywany przez niegraficzne przeglądarki), ponieważ jest wymagany przez najnowszy standard HTML, a poza tym ułatwia odczytywanie naszej strony użytkownikom niepełnosprawnym, takim jak niewidomi, którzy muszą stosować tak zwane czytniki ekranowe.

Wady przedstawionej techniki podmieniania obrazków

Opisana technika podmieniania obrazków jest bardzo prosta, ale trzeba mieć świadomość kilku związanych z nią problemów.

- ◆ Drugi obrazek pobierany jest z serwera dopiero w chwili wskazania myszą pierwszego obrazka. Z tego powodu, zanim obrazki zostaną zamienione miejscami, bardzo prawdopodobne jest zaistnienie zauważalnego opóźnienia szczególnie wtedy, kiedy użytkownik korzysta z modemu, a nie łącza szerokopasmowego.
- ◆ Wykorzystanie tej metody powoduje komunikaty o błędach w starszych przeglądarkach, takich jak Netscape 2.0, Internet Explorer 3.0 lub America Online 2.7. Na szczęście, dzisiaj już praktycznie nikt nie korzysta z tak starych przeglądarek, więc tego ograniczenia nie należy uważać za poważny problem.

Zamiast przedstawionej tu techniki polecamy sposób z następnego podrozdziału, który rozwiązuje obydwa opisane wyżej problemy.

[illegible]

Aby uzyskać rzeczywiste wrażenie animacji, musimy zadbać o to, aby obrazek zmiennik pojawił się natychmiast, bez zwłoki spowodowanej pobieraniem z serwera. W tym celu wykorzystamy JavaScript do załadowania do bufora przeglądarki wszystkich obrazków (tak aby znajdowały się już na dysku twardym komputera, kiedy będą potrzebne), które umieścimy w zmiennych skryptu. Dzięki temu w chwili wskazania obrazka skrypt szybko podmieni jedną zmienną zawierającą obrazek na drugą. Przykład przedstawiony jest w skrypcie 4.3. Widoczny efekt jest taki sam jak na rysunkach 4.1 i 4.2, jednak animacja przebiega sprawniej.

```

window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.images.length;
        i++) {
        if (document.images[i].parentNode.
            tagName == "A") {
            setupRollover(document.images[i]);
        }
    }
}

function setupRollover(thisImage) {
    thisImage.outImage = new Image();
    thisImage.outImage.src = thisImage.src;
    thisImage.onmouseout = function() {
        this.src = this.outImage.src;
    }

    thisImage.overImage = new Image();
    thisImage.overImage.src = "images/" +
        thisImage.id + "_on.gif";
    thisImage.onmouseover = function() {
        this.src = this.overImage.src;
    }
}

```

W celu ułatwienia obsługi kodu JavaScript wydzieliliśmy go z dokumentu HTML i umieściliśmy w zewnętrznym pliku `.js`, który został przedstawiony w skrypcie 4.4 (więcej informacji o plikach `.js` podawaliśmy w rozdziale 3.).

l. `<script src="script02.js"></script>`

```
2. <a href="next1.html"><img src=
   ↳ "images/button1_off.gif" alt="button1"
   ↳ id="button1"></a>&nbsp;&nbsp; 
```

Ciąg dalszy skryptu 4.3 zawiera typowe znaczniki łączy z obrazkowymi przyciskami. Atrybut href opisuje adres strony, która zostanie wyświetlona po tym, jak użytkownik kliknie przycisk. W znaczniku atrybut src definiuje ścieżkę do obrazka wyświetlanego przed dokonaniem podmiany. Należy zauważyć, że każdy z tych dwóch przycisków ma też swój unikalny identyfikator (o którym mowa była w rozdziale 1.). Identyfikatory te pozwolą skryptowi JavaScript na dokonanie podmiany obrazków.

Przechodzimy do skryptu 4.4. Funkcja obsługi zdarzenia `window.onload` wywoływana jest zaraz po zakończeniu ładowania strony. W ramach obsługi zdarzenia wywoływana jest funkcja `rolloverInit()`.

Zdarzenie to służy do upewnienia się, że funkcja nie zostanie uruchomiona przed zakończeniem ładowania strony. Po prostu próby odwoływania się do elementów niezaladowanej w całości strony mogą spowodować błędy, jeżeli żądany element nie zostanie jeszcze załadowany.

```
4. function rolloverInit() {
    for (var i=0; i<document.images.
        ↳length; i++) {
```

Funkcja `rolloverInit()` przegląda wszystkie obrazki na stronie i sprawdza, czy są otoczone znacznikami `<a>`, co wskazuje na to, że są łącami. Funkcja zaczyna się od pierwszego z podanych wierszy kodu. W drugim wierszu tworzona jest pętla `for`, przeglądająca wszystkie obrazki ze strony. Na początku zmiennej `i` przypisywana jest wartość zero, a następnie pętla kontynuuje swoje obiegi tak długo, jak długo wartość zmiennej `i` jest mniejsza od liczby obrazków na stronie. Przy każdym obiegu wartość zmiennej `i` jest inkrementowana.

```
5. if (document.images[i].parentNode.
    ↳tagName == "A") {
```

To właśnie w tym wierszu sprawdzane jest, czy dany obrazek otoczony jest znacznikiem łącza. Wykonywane jest to poprzez pobranie odpowiedniego obiektu i sprawdzenie, czy jego nazwą jest znak "A" (nazwa znacznika łącza). Spróbujmy rozłożyć taki obiekt na części. Zapis `document.images[i]` oznacza aktualny obrazek. Właściwość `parentNode` wskazuje na znacznik otaczający ten obrazek. Z kolei właściwość `tagName` podaje nazwę tego znacznika. Oznacza to, że zapis w nawiasach można by przetłumaczyć na język polski tak: „Czy znacznik otaczający bieżący obrazek nazywa się »A«?”.

```
6. setupRollover(document.images[i]);
```

Jeżeli wynik testu z kroku 5. będzie pozytywny, wywoływana jest funkcja `setupRollover()`, której w parametrze przekazywany jest bieżący obrazek.

7. function setupRollover(thisImage) {

Warto poświęcić chwilkę i przyjrzeć się całej funkcji, zanim zaczniemy analizować ją wiersz po wierszu. Oto krótki przegląd: funkcja do każdego przekazanego jej obrazka dodaje dwie właściwości. Są to właściwości `outImage` (domyślna wersja obrazka) oraz `overImage` (wersja obrazka pojawiająca się po wskazaniu go myszą), które same w sobie są również obiektami obrazków. Dzięki temu po ich utworzeniu możemy dodać do nich atrybuty `src`. Atrybut `src` z obiektu `outImage` będzie kopią atrybutu `src` bieżącego obrazka, z kolei dla obiektu `overImage` wartość atrybutu jest wyliczana na podstawie wartości identyfikatora oryginalnego obrazka.

Ten wiersz rozpoczyna funkcję `rolloverInit()`, pobierającą w parametrze obiekt obrazka.

8. thisImage.outImage = new Image();

W tym wierszu do przekazanego funkcji obrazka dodawana jest nowa właściwość `outImage`, do której przypisywany jest obiekt nowego obrazka. Dzięki temu, że właściwości można dodawać do dowolnych obiektów, a dodatkowo są one kolejnymi obiektami, możemy po prostu przypisać do nowej właściwości nowo utworzony obiekt. Nawiasy za nazwą tworzonego obiektu obrazka są opcjonalne, ale ich stosowanie należy do dobrych praktyk programistycznych. Jeśli trzeba, można w nie wpisać właściwości nowo tworzonego obiektu.

9. `thisImage.outImage.src = thisImage.src;`

Teraz definiowane jest źródło nowego obrazka. Jak widać, jest ono tożsame ze źródłem obrazka oryginalnego. Domyślna postać obrazka umieszczanego na stronie widoczna będzie zawsze wtedy, gdy kursor myszy znajduje się poza nim.

10. `thisImage.onmouseout = function() {
 this.src = this.outImage.src;
}`

W tym wierszu definiowana jest tak zwana funkcja *anonimowa*, czyli funkcja nieposiadająca nazwy. Moglibyśmy nazwać ją na przykład `rollOut()`, ale ze względu na to, że składa się z tylko jednego wiersza, możemy pominąć jej nazwę.

W tym miejscu informujemy przeglądarkę, co powinna zrobić w momencie, gdy użytkownik przesunie wskaźnik myszy poza obrazek. W takiej sytuacji chcemy, żeby przywrócona została początkowa wersja obrazka, zapisana w zmiennej `outImage`.

11. `thisImage.overImage = new Image();
thisImage.overImage.src = "images/"
 ↪ + thisImage.id + "_on.gif";`

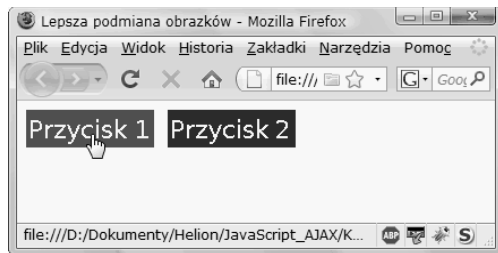
W pierwszym wierszu tworzymy nowy obiekt obrazu, który będzie zawierać wersję obrazka wyświetlaną po wskazaniu go myszą. Drugi wiersz definiuje źródło obrazka dla obiektu `overImage`. Nazwa pliku budowana jest na bieżąco przez złożenie nazwy katalogu `images/`, identyfikatora obrazka podstawowego (pamiętamy, że w skrypcie 4.3 przyciskom nadaliśmy identyfikatory `button1` i `button2`) i uzupełnienie całości o przyrostek `"_on.gif"`.

```
12. thisImage.onmouseover = = function() {
    this.src = this.overImage.src;
}
```

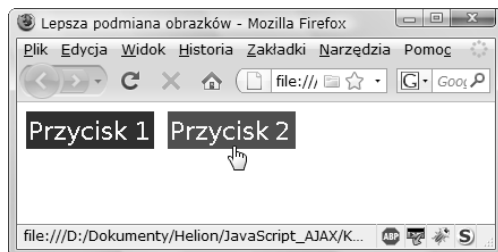
Tutaj mamy kolejną funkcję anonimową. Nakazuje ona przeglądarce wyświetlić obrazek zapisany w zmiennej `overImage` w chwili, gdy użytkownik przesunie nad niego wskaźnik myszy (rysunki 4.3 i 4.4).

Wskazówki

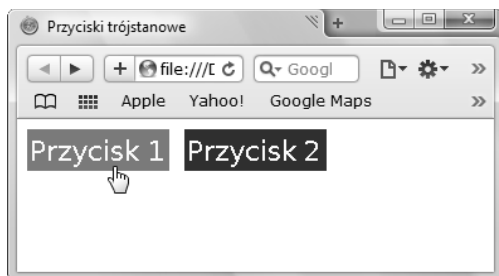
- W czasie przygotowywania pary podmienianych obrazków trzeba przypilnować, żeby obrazki GIF lub PNG *nie* były przezroczyste. Spod przezroczystych obrazków widać będzie obrazki, które miały być przez nie zasłonięte, a w końcu nie o to chodzi.
- Oba rysunki muszą mieć takie same rozmiary. Jeśli tego nie dopilnujemy, przeglądarka rozszerzy mniejszy obrazek do rozmiarów większego — rzadko wygląda to dobrze.
- W poprzednim przykładzie podmiana wykonywana była po wskazaniu kursorem myszy samego łącza. Tym razem następuje po wskazaniu obrazka, czyli w ramach zdarzeń `onmouseover` i `onmouseout`, powiązanych ze znacznikiem `<img>`, a nie ze znacznikiem `<a>`. Obie metody zazwyczaj dają te same rezultaty, ale niektóre starsze przeglądarki (Netscape 4 i wcześniejsze, IE 3 i wcześniejsze) nie obsługują zdarzeń `onmouseover` i `onmouseout` w znaczniku `<img>`.
- Można sobie pomyśleć, że ze względu na to, iż wszystkie znaczniki HTML na stronie zapisane są małymi literami, właściwość `tagName` powinna być porównywana do małej litery „a”. Trzeba jednak pamiętać, że właściwość ta zawsze zwraca tekst zapisany wielkimi literami.



Rysunek 4.3. Na jednej stronie można mieć wiele podmienianych obrazków



Rysunek 4.4. Wskazujemy drugi obrazek



Rysunek 4.5. Po kliknięciu przycisku pojawia się trzeci obrazek (na czarno-białych stronach książki będzie to mało widoczne, lepiej uruchomić przykład w przeglądarce)

Skrypt 4.5. Umieszczenie kodu JavaScript w zewnętrznym pliku pozwala zastosować na stronie przyciski trójstanowe bez jakiegokolwiek modyfikacji kodu HTML

[illegible]

Tworzenie przycisków trójstanowych

Przycisk trójkątny jest podmienianym obrazkiem, który ma trzy różne wersje. Oprócz oryginalnego obrazka oraz wersji pojawiającej się po wskazaniu przycisku kursorem myszy, tworzona jest jeszcze trzecia wersja, która wyświetla się po kliknięciu przycisku, tak jak na rysunku 4.5.

Skrypt 4.5 to plik HTML wyglądający niemal tak samo jak plik ze skryptu 4.3. Różnica między nimi polega na różnych tytułach i nazwach zewnętrznych plików JavaScript stosowanych na stronie. To wszystko. Jest to doskonały przykład tego, że umieszczenie całego kodu JavaScript w zewnętrznym pliku jest bardzo potężną techniką pozwalającą na dodawanie do stron nowych funkcji bez konieczności modyfikowania samego kodu HTML.

W skrypcie 4.6 można zobaczyć zawartość zewnętrznego pliku JavaScript. Znajduje się w nim zaledwie kilka zmian w stosunku do kodu ze skryptu 4.4. Nie będziemy zatem od początku analizować całości, ale skoncentrujemy się na wprowadzonych zmianach. Omawiane części skryptu wyróżnione zostały pogrubieniem.

Oto przygotowanie przycisku trójstanowego:

```
1. thisImage.clickImage = new Image();
   thisImage.clickImage.src = "images/"
   + thisImage.id + "_click.gif";
```

W funkcji `setupRollover()` musimy dodać trzecią właściwość obiektu obrazka, opisującą stan po kliknięciu. W pierwszym wierszu tworzony jest nowy obiekt obrazka, który będzie przechowywał dodatkowy obraz. W drugim wierszu definiowane jest źródło obrazu `clickImage`. Nazwa pliku obrazka tworzona jest przez złożenie nazwy katalogu `images/` z identyfikatorem oryginalnego obrazka i dopiskiem `_click.gif`.

```
2. thisImage.onclick = function() {
    this.src = this.clickImage.src;
}
```

Ten wiersz to informacja dla przeglądarki, co należy zrobić, gdy użytkownik kliknie obrazek. W tym przypadku chodzi o podmianę obrazka na wersję wskazywaną przez obiekt `clickImage`.

Skrypt 4.6. Skrypt obsługujący przyciski trójstanowe

```
Skrypt

window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.images.length;
        i++) {
        if (document.images[i].parentNode.
            tagName == "A") {
            setupRollover(document.images[i]);
        }
    }
}

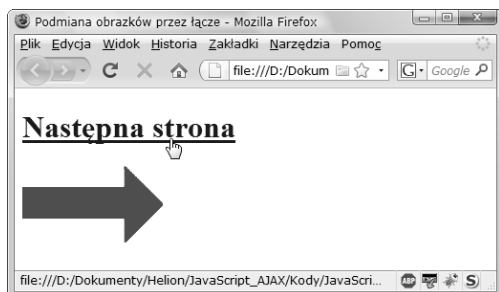
function setupRollover(thisImage) {
    thisImage.outImage = new Image();
    thisImage.outImage.src = thisImage.src;
    thisImage.onmouseout = function() {
        this.src = this.outImage.src;
    }

    thisImage.clickImage = new Image();
    thisImage.clickImage.src = "images/" +
        thisImage.id + "_click.gif";
    thisImage.onclick = function() {
        this.src = this.clickImage.src;
    }

    thisImage.overImage = new Image();
    thisImage.overImage.src = "images/"
        + thisImage.id + "_on.gif";
    thisImage.onmouseover = function() {
        this.src = this.overImage.src;
    }
}
```



Rysunek 4.6. W tekstowym łączu zawarty jest mechanizm powodujący podmianianie obrazków



Rysunek 4.7. W chwili wskazania łącza zmienia się obrazek

Skrypt 4.7. Podany kod tworzy stronę HTML do podmiany obrazków za pomocą łącza

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Podmiana obrazka przy użyciu
    ↪ łącza</title>
  <script src="script04.js"></script>
  <link rel="stylesheet" href="script01.css">
</head>
<body>
  <h1><a href="next.html"
    ↪ id="arrow">Następna strona</a></h1>
  
</body>
</html>
```

Podmiana obrazków poprzez łącze

We wcześniejszych przykładach użytkownik powodował zmianę rysunków, wskazując rysunek kursorem myszy. Można także sprawić, aby zamiana rysunków dokonywała się w chwili umieszczenia kursora nad łączem, co pokazano na rysunkach 4.6 i 4.7. Kod HTML użyty w tym przykładzie tworzy mało ciekawą stronę z jednym łączem i jednym obrazkiem (skrypt 4.7). Podmiany obrazków dokonamy przez zmodyfikowanie skryptu z poprzednich przykładów (skrypt 4.8).

Oto podmiana obrazka przy użyciu łącza:

```
1. function rolloverInit() {
    for (var i=0; i<document.links.
        length; i++) {
```

Na początku funkcji rolloverInit() rozpoczynana jest pętla, podobna do tej z poprzednich przykładów. Tym razem jednak nie szukamy obrazków (document.images.length), ale łączy, jakie znajdują się w dokumencie (document.links.length). Pętla rozpoczyna się od przypisania zera do zmiennej i. Po każdym obiegu, jeżeli wartość tej zmiennej jest mniejsza od liczby łączy w dokumencie, jest ona inkrementowana.

```
2. var linkObj = document.links[i];
```

Tutaj tworzona jest zmienna linkObj, do której wpisujemy aktualne łącze.

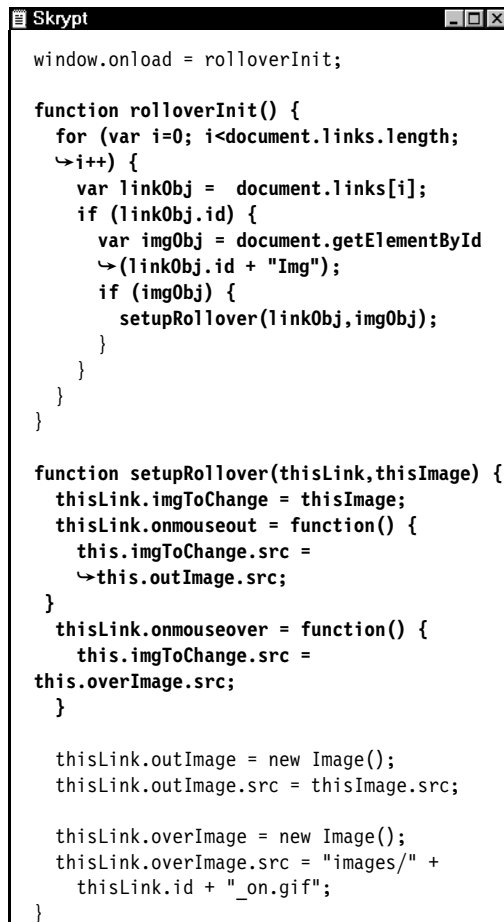
```
3. if (linkObj.id) {
    var imgObj = document.
        ↳getElementById(linkObj.id + "Img");
```

Jeżeli element linkObj ma identyfikator, sprawdzamy, czy na stronie dostępny jest inny element o takim samym identyfikatorze uzupełnionym o dopisek Img. Jeżeli taki się znajdzie, umieszczamy go w nowo utworzonej zmiennej imgObj.

```
4. if (imgObj) {
    setupRollover(linkObj, imgObj);
```

Jeżeli istnieje obiekt imgObj, wywoływana jest funkcja setupRollover(), której w parametrach są przekazywane obiekt obrazka i łącza.

Skrypt 4.8. Oto kod JavaScript powodujący podmianę obrazków przez łącze



```
Skrypt
window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.links.length;
        ↳i++) {
        var linkObj = document.links[i];
        if (linkObj.id) {
            var imgObj = document.getElementById
                ↳(linkObj.id + "Img");
            if (imgObj) {
                setupRollover(linkObj, imgObj);
            }
        }
    }
}

function setupRollover(thisLink, thisImage) {
    thisLink.imgToChange = thisImage;
    thisLink.onmouseout = function() {
        this.imgToChange.src =
            ↳this.outImage.src;
    }
    thisLink.onmouseover = function() {
        this.imgToChange.src =
            this.overImage.src;
    }

    thisLink.outImage = new Image();
    thisLink.outImage.src = thisImage.src;

    thisLink.overImage = new Image();
    thisLink.overImage.src = "images/" +
        thisLink.id + "_on.gif";
}
```



```
5. function setupRollover(thisLink,
    ↪thisImage) {
    thisLink.imgToChange = thisImage;
```

Funkcja `setupRollover()` zaczyna się od pobrania parametrów opisujących łącze i obrazek, które przekazywane są jej w kroku 4. Następnie do obiektu łącza dodawana jest nowa właściwość o nazwie `imgToChange`. Skrypt musi w jakiś sposób „dowiedzieć” się, jaki obrazek ma zostać zmieniony po wskazaniu łącza kursorem myszy. Informacja ta zapisywana jest właśnie w tej właściwości.

```
6. thisLink.onmouseout = function() {
    this.imgToChange.src = this.
        outImage.src;
    }
    thisLink.onmouseover = function() {
        this.imgToChange.src = this.
            ↪overImage.src;
    }
```

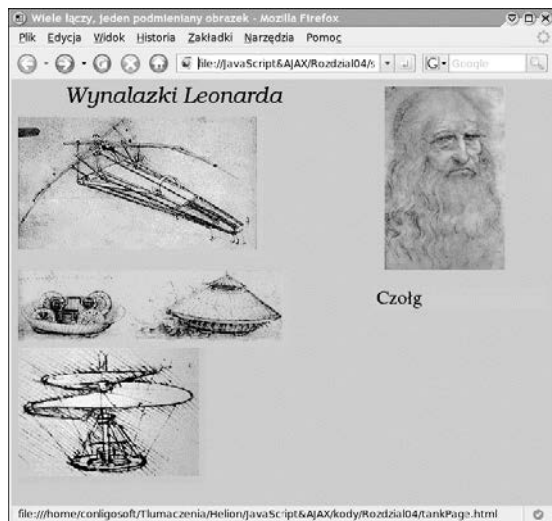
W momencie wywołania zdarzenia `mouseover` lub `mouseout` obserwujemy działanie nieco inne od prezentowanego w poprzednich przykładach. Tym razem modyfikowana jest właściwość `this.imgToChange.src`, a nie właściwość `this.src`, tak jak to było robione poprzednio.

Wskazówka

- Technika ta przydaje się, gdy chcemy poinformować użytkownika, co zobaczy, kiedy kliknie wskazywane w tej chwili łącze. Załóżmy, że prowadzimy stronę biura podróży opisującą wycieczki do Szkocji, na Hawaje i do Cleveland. Po lewej stronie można by umieścić kolumnę z tekstowymi linkami do wybranych miejsc, a z prawej — obszar podglądu, w którym pojawiałyby się odpowiednie zdjęcia. W momencie wskazania łącza do danego miejsca po prawej stronie pojawiałby się jego podgląd. Kliknięcie łącza prowadziłoby użytkownika do strony ze szczegółami dotyczącymi miejsca wycieczki.

Podmienianie obrazka z różnych łączy

Do tej pory efekt zmiany rysunku wywoływany był po wskazaniu myszą pojedynczego obrazka bądź łącza tekstowego. Można jednak również utworzyć stronę, w której efekt ten będzie wywoływany z wielu różnych miejsc — takie rozwiązanie jest idealne do opisywania treści rysunków na stronie. W naszym przykładzie opisaliśmy w ten sposób trzy obrazy projektów Leonarda da Vinci. Po wskazaniu któregoś z nich w polu tekstowym po prawej stronie pojawia się opis obiektu na rysunku. W rzeczywistości opis ten również jest rysunkiem, a dokładniej — jednym z trzech różnych rysunków (po jednym dla każdego wynalazku). Działanie skryptów 4.9 (HTML), 4.10 (CSS) i 4.11 (JavaScript) przedstawiono na rysunku 4.8. Podobnie jak inne skrypty w książce, są one tworzone na bazie poprzednich, w związku z czym skupimy się tylko na nowych rozwiązaniach. Skrypty 4.8 i 4.11 różnią się tylko kilkoma wierszami kodu.



Rysunek 4.8. Na stronie znajdują się trzy rysunki przedstawiające projekty wynalazków — samolotu, czołgu oraz helikoptera. Po wskazaniu któregoś z nich w polu tekstowym pojawia się opis

Skrypt 4.9. Należy zauważyć, że łącza i obrazki na tej stronie mają swoje identyfikatory

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Wiele łączy, jeden podmieniany
  obrazek</title>
  <script src="script05.js"></script>
  <link rel="stylesheet"
  href="script02.css">
</head>
<body>
  <div id="captionDiv">
    
    
  </div>
  <div id="inventionDiv">
    
    <a href="flyPage.html"
    class="captionField" id="flyer"></a>
    <a href="tankPage.html"
    class="captionField" id="tank"></a>
    <a href="heliPage.html"
    class="captionField" id="helicopter"></a>
  </div>
</body>
</html>
```


Skrypt 4.10. Ten skrypt pozwala podmieniać jeden obrazek poprzez wiele łącz

```
Skrypt
body {
    background-color: #EC9;
}

img {
    border-width: 0;
}

#captionDiv {
    float: right;
    width: 210px;
    margin: auto 50px;
}

#captionField {
    margin: 20px auto;
    width: 208px;
    height: 27px;
}

#inventionDiv {
    width: 375px;
    margin-left: 20px;
}

#heading {
    margin-bottom: 20px;
    width: 375px;
    height: 26px;
}
```

Skrypt 4.11. Ten skrypt pozwala podmieniać jeden obrazek poprzez wiele łącz

```
Skrypt
window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.links.length;
        ↪i++) {
        var linkObj = document.links[i];
        if (linkObj.className) {
            var imgObj = document.getElementById
                ↪(linkObj.className);
            if (imgObj) {
                setupRollover(linkObj,imgObj);
            }
        }
    }

    function setupRollover(thisLink,textImage) {
        thisLink.imgToChange = textImage;
        thisLink.onmouseout = function() {
            this.imgToChange.src =
                ↪this.outImage.src;

            thisLink.outImage = new Image();
            thisLink.outImage.src = textImage.src;

            thisLink.overImage = new Image();
            thisLink.overImage.src = "images/" +
                ↪thisLink.id + "Text.gif";
        }
    }
}
```

Oto sposób, by wiele łączy mogło podmieniać jeden obrazek:

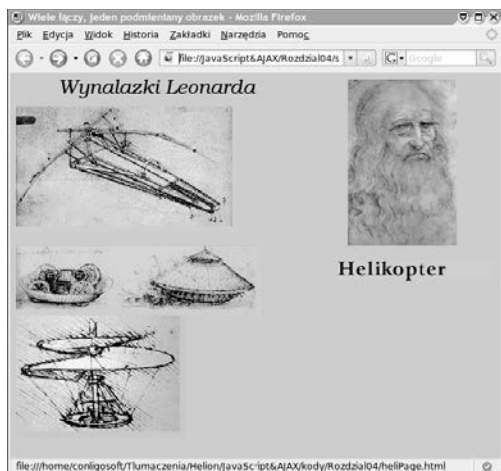
```
1. if (linkObj.className) {
    var imgObj = document.
        ↳getElementById(linkObj.className);
```

Nie możemy skorzystać z identyfikatorów obrazków w celu wyznaczenia identyfikatora obrazka podmienianego. Po prostu identyfikatory muszą być unikalne. Z tego powodu każdy z obrazków musi mieć jakąś wartość opisującą umiejscowienie podmienianego obrazka, a zatem musimy skorzystać z atrybutu `class` (na stronie może znajdować się wiele elementów o takiej samej wartości tego atrybutu). W tym wierszu kodu przeszukujemy właściwości `className` obiektów łączy.

```
2. function setupRollover(thisLink,
    ↳textImage) {
    thisLink.imgToChange = textImage;
```

Funkcja `setupRollover()` otrzymuje w parametrach obiekt aktualnego łącza (`thisLink`) oraz obiekt obrazka, który tutaj nazywany jest `textImage`. Warto zauważyć, że w czasie wywoływania tej funkcji przekazywane jej obiekty (można o nich myśleć jak o zmiennych) miały nazwy `linkObj` i `imgObj`.

Pozostała część skryptu działa dokładnie tak samo jak w poprzednich przykładach z tego rozdziału.



Rysunek 4.9. Po wskazaniu jednego z rysunków w polu tekstowym pojawia się jego opis, a sam rysunek otrzymuje obramowanie

Podmienianie wielu obrazków z jednego łącza

Co należy zrobić, by łącze podmieniające jeden z rysunków na stronie samo w sobie również było rysunkiem zmieniającym swój wygląd po wskazaniu myszą? Jak można zobaczyć na rysunku 4.9, dodaliśmy tę funkcję do skryptu przedstawionego w poprzednim przykładzie. Podobnie jak poprzednio, po wskazaniu myszą jednego z rysunków w polu tekstowym pojawia się jego opis, a dodatkowo rysunek zostaje zastąpiony innym, w którym dodano obramowanie. Dzięki temu użytkownik otrzymuje dodatkową informację na temat tego, co właśnie wskazuje (w przypadku gdyby kursor myszy nie był wystarczający). W skrypcie 4.12 został przedstawiony kod HTML strony (praktycznie bez żadnych zmian, z wyjątkiem nazwy zewnętrznego pliku z kodem JavaScript), natomiast w skrypcie 4.13 można zobaczyć niewielkie modyfikacje, jakie zostały wprowadzone do kodu z poprzedniego przykładu.

Skrypt 4.12. Ten skrypt HTML jest dokładnie taki sam jak skrypt 4.9, jedyne różnice to inny tytuł i odwołanie do innego pliku JavaScript

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Wiele łączy, wiele podmienianych obrazków</title>
  <script src="script06.js"></script>
  <link rel="stylesheet" href="script02.css">
</head>
<body>
  <div id="captionDiv">
    
    
  </div>
  <div id="inventionDiv">
    
    <a href="flyPage.html" class="captionField" id="flyer"></a>
    <a href="tankPage.html" class="captionField" id="tank"></a>
    <a href="heliPage.html" class="captionField" id="helicopter"></a>
  </div>
</body>
</html>
```

Skrypt 4.13. Ten skrypt obsługuje wiele podmienianych obrazków

```

Skrypt
window.onload = rolloverInit;

function rolloverInit() {
    for (var i=0; i<document.links.length; i++) {
        var linkObj = document.links[i];
        if (linkObj.className) {
            var imgObj = document.getElementById (linkObj.className);
            if (imgObj) {
                setupRollover(linkObj,imgObj);
            }
        }
    }
}

function setupRollover(thisLink,textImage) {
    thisLink.imgToChange = new Array;
    thisLink.outImage = new Array;
    thisLink.overImage = new Array;

    thisLink.imgToChange[0] = textImage;
    thisLink.onmouseout = rolloOut;
    thisLink.onmouseover = rolloOver;

    thisLink.outImage[0] = new Image();
    thisLink.outImage[0].src = textImage.src;

    thisLink.overImage[0] = new Image();
    thisLink.overImage[0].src = "images/" + thisLink.id + "Text.gif";

    var rolloverObj = document.getElementById (thisLink.id + "Img");
    if (rolloverObj) {
        thisLink.imgToChange[1] = rolloverObj;

        thisLink.outImage[1] = new Image();
        thisLink.outImage[1].src = rolloverObj.src;

        thisLink.overImage[1] = new Image();
        thisLink.overImage[1].src = "images/" + thisLink.id + "_on.gif";
    }
}

function rolloOver() {
    for (var i=0;i<this.imgToChange.length; i++) {
        this.imgToChange[i].src = this.overImage[i].src;
    }
}

function rolloOut() {
    for (var i=0;i<this.imgToChange.length; i++) {
        this.imgToChange[i].src = this.outImage[i].src;
    }
}

```

Oto podmiana wielu obrazków jednocześnie:

```
1. thisLink.imgToChange = new Array;
   thisLink.outImage = new Array;
   thisLink.overImage = new Array;
```

Wiersze te zostały dopisane dlatego, że teraz skrypt musi działać z większą liczbą obrazków (dwa na każdy podmieniany obrazek).

W poszczególnych wierszach tworzone są nowe właściwości obiektu `thisLink`. Każda z tych właściwości jest osobnym obiektem nazywanym tablicą (ang. *array*).

```
2. thisLink.imgToChange[0] = textImage;
```

W poprzednim zadaniu właściwość `imgToChange` była obrazkiem, ale tym razem jest tablicą przechowującą obrazki. Jak widać, wartość zmiennej `textImage` zapisywana jest jako pierwszy element tablicy `imgToChange`.

```
3. thisLink.outImage[0] = new Image();
   thisLink.outImage[0].src = textImage.src;
```

Podobnie jak poprzednio, musimy zachować też nieaktywną wersję obrazka, ale tym razem zapisujemy go jako pierwszy element tablicy `outImage`.

```
4. thisLink.overImage[0] = new Image();
   thisLink.overImage[0].src = "images/" +
   ↪thisLink.id + "Text.gif";
```

Aktywna wersja obrazka wyliczana jest tak jak w poprzednich przykładach i zapisywana jako pierwszy element tablicy `overImage`.

```
5. var rolloverObj = document.  
   getElementById(thisLink.id + "Img");  
   if (rolloverObj) {
```

Teraz musimy sprawdzić, czy podmiana dotyczyć będzie wielu obrazków, czy też tylko pojedynczego. W takim przypadku na stronie będzie znajdować się element o takim samym identyfikatorze jak ten, ale uzupełnionym o dopisek `Img`. Oznacza to, że w przypadku, gdy aktualny element ma identyfikator `flyer`, na stronie powinien znajdować się element o identyfikatorze `flyerImg`. Jeżeli tak jest, jest on zapisywany do zmiennej `rolloverObj` i wykonywane są trzy kolejne kroki.

```
6. thisLink.imgToChange[1] = rolloverObj;
```

Przedstawiony wyżej sposób pracy z elementem `imgToChange[0]` powtarzamy teraz dla elementu `imgToChange[1]`, czyli przypisujemy mu wartość zmiennej `rolloverObj`. W momencie wywołania funkcji obsługujących zdarzenia `onmouseover` lub `onmouseout` oba rysunki zostaną zastąpione właściwymi zastępnikami.

```
7. thisLink.outImage[1] = new Image();  
   thisLink.outImage[1].src =  
   ↪rolloverObj.src;
```

Te instrukcje definiują drugi element tablicy `outImage`, przechowującej nieaktywne wersje obrazka.

```
8. thisLink.overImage[1] = new Image();  
   thisLink.overImage[1].src = "images/" +  
   ↪thisLink.id + "_on.gif";
```

W tym miejscu wyznaczana jest aktywna wersja obrazka, a następnie wpisywana jako drugi element tablicy `overImage`.

Gdybyśmy chcieli jednocześnie podmieniać jeszcze trzeci obrazek, należałoby powtórzyć kroki od 6. do 8., wprowadzając do nich odpowiednie modyfikacje.



```
9. for (var i=0;i<this.imgToChange.
    ↳length; i++) {
    this.imgToChange[i].src =
    ↳this.overImage[i].src;
  }
```

Wewnątrz funkcji `rollOver()` wykonywana jest zamiana obrazków. Podmiany wymagać może jeden lub więcej obrazków, a zatem musimy sprawdzić, ile z nich zostało zapisanych w tablicy. Ta informacja zapisana jest we właściwości `this.imgToChange.length`. W tym przypadku otrzymamy wartość 2, ponieważ musimy zmienić tylko dwa obrazki. Pętla będzie miała zatem dwa obiegi, w których najpierw wykorzystamy wartości zapisane w elemencie `imgToChange[0]`, a następnie w elemencie `imgToChange[1]`.

```
10. for (var i=0;i<this.imgToChange.
    ↳length; i++) {
    this.imgToChange[i].src =
    ↳this.outImage[i].src;
  }
```

Funkcja `rollOut()` działa niemal dokładnie tak samo jak funkcja z poprzedniego kroku. Różnica polega na tym, że tym razem przywracane są oryginalne obrazki.

Wskazówki

- Należy bardzo uważać, by nazwy podmienianych rysunków nie powtarzały się — każdy z nich musi posiadać własną, unikalną nazwę.
- Co zrobić, jeżeli chcemy, żeby pewne łącza przełączały jednocześnie wiele obrazków, a pozostałe związane były z pojedynczym obrazkiem? To żaden problem. Nie trzeba zmieniać nawet jednego wiersza kodu JavaScript. Wystarczy, że instrukcja w kroku 5. nie znajdzie na stronie odpowiedniego identyfikatora. W takiej sytuacji funkcja nie zapisze drugiego elementu do tablicy, a funkcje `rollOver()` i `rollOut()` będą przełączały wyłącznie podstawowy obrazek.

Tworzenie animowanych banerów

Podczas oglądania stron WWW często można natknąć się na reklamowe banery, w których co chwila zmieniają się wyświetlane obrazki. Większość z nich to animowane pliki GIF, w których znajduje się kilka kolejno wyświetlanych obrazków. Gdybyśmy chcieli zbudować stronę, na której wyświetla się kilka takich obrazków — animowanych bądź nie — możemy użyć języka JavaScript. Przykład przedstawiamy w skrypcie 4.15. Wykorzystano w nim trzy kolejno wyświetlane obrazki GIF (można je zobaczyć na rysunkach 4.10, 4.11 oraz 4.12). Kod prostej strony HTML przedstawiony został w skrypcie 4.14.

Oto tworzenie cyklicznie wyświetlanych banerów:

1. `var thisAd = 0;`

Nasz skrypt zaczyna się od utworzenia zmiennej `thisAd` i nadania jej wartości początkowej.

2. `function rotate() {`
`var adImages = new Array("images/`
`↳reading1.gif", "images/reading2.gif",`
`↳"images/reading3.gif");`

Ten wiersz rozpoczyna nową funkcję o nazwie `rotate()`. W kolejnym wierszu tworzona jest tablica o nazwie `adImages`, która będzie przechowywała nazwy trzech plików GIF tworzących cyklicznie zmieniający się baner.

3. `thisAd++;`

Pobiera wartość zmiennej `thisAd` i powiększa ją o 1.

4. `if (thisAd == adImages.length) {`
`thisAd = 0;`

Ten kod sprawdza, czy licznik banerów (zmienna `thisAd`) osiągnął ogólną liczbę elementów w tablicy, a jeżeli tak, wpisuje do zmiennej wartość zero.

Skrypt 4.14. Kod HTML ładuje pierwszy obrazek banera, a resztą działań zajmuje się JavaScript

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Animowane banery</title>
  <script src="script07.js"></script>
  <link rel="stylesheet"
    ↳href="script01.css">
</head>
<body>
  <div class="centered">
    
  </div>
</body>
</html>
```

Skrypt 4.15. Kod JavaScript służy do cyklicznego podmieniania banerów

```
Skrypt
window.onload = rotate;

var thisAd = 0;

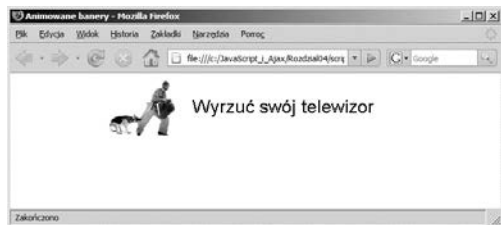
function rotate() {
  var adImages = new Array("images/
    ↳reading1.
    ↳gif", "images/reading2.gif", "images/
    ↳reading3.gif");

  thisAd++;
  if (thisAd == adImages.length) {
    thisAd = 0;
  }
  document.getElementById("adBanner").src =
    ↳adImages[thisAd];

  setTimeout(rotate, 3 * 1000);
}
```




Rysunek 4.10. Pierwszy obrazek, od którego rozpoczyna się animowany baner



Rysunek 4.11. Drugi obrazek...



Rysunek 4.12. ...ostatni obrazek. Po załadowaniu strony rozpoczynają się podmiiany banerów, które nie wymagają żadnej interwencji ze strony użytkownika

5. `document.getElementById("adBanner")`
`↪.src = adImages[thisAd];`

Znajdujący się na stronie obrazek, który będzie poddawany podmianom, ma identyfikator `adBanner`. Odpowiednia nazwa została zdefiniowana w znaczniku `<img>`, o czym można się przekonać w skrypcie 4.14. Ten wiersz kodu przepisuje adres źródła obrazka z tablicy `adImages` z pozycji wyznaczonej przez zmienną `thisAd`.

6. `setTimeout(rotate, 3 * 1000);`

Ten wiersz skryptu wyznacza częstotliwość zmian obrazków w banerze reklamowym. Wbudowane polecenie `setTimeout()` pozwala określić, że po pewnym czasie ma zostać wykonana instrukcja wpisana w poleceniu. W tym przypadku jest to funkcja `rotate()`, która będzie wywoływana co 3000 milisekund, czyli co trzy sekundy.

Wskazówki

- Można się tu zastanawiać, po co konstruować animowane banery za pomocą JavaScriptu, zamiast wykorzystać po prostu animowane obrazki GIF. JavaScript pozwala na zastosowanie w animowanych banerach plików w formacie JPG lub PNG, które umożliwiają tworzenie obrazków o dużo lepszej jakości. Dzięki temu banery mogą mieć niemal fotograficzną jakość.
- W przeciwieństwie do przedstawianych wcześniej przykładów, obrazki banerów nie są ładowane wcześniej do bufora. Każdy z nich ładowany jest dopiero wtedy, gdy ma zostać wyświetlony. Po prostu w tablicy banerów może znajdować się dowolna liczba obrazów, a zmuszanie przeglądarki do pobierania od razu na przykład stu obrazów spowodowałoby spadek prędkości wyświetlania strony. Poza tym nie miałyby to większego sensu, jeżeli użytkownik zobaczyłby najwyżej dwa lub trzy obrazki, po czym przeszedł na inną stronę.

Dodawanie łączy do animowanych banerów

Animowane banery są bardzo często wykorzystywane w reklamie — z pewnością warto wiedzieć, w jaki sposób można utworzyć baner będący łączem, które po kliknięciu przeniesie oglądającego na inną stronę. Sposób rozwiązania tego problemu przedstawiliśmy w skrypcie 4.16, bazującym na poprzednim przykładzie (wokół znacznika `<img>` dodaliśmy znacznik `<a>`).

W skrypcie 4.17 można zobaczyć lekką wariację na temat skryptu z poprzedniego przykładu. Jak widać, dodaliśmy też nową tablicę. Znajdują się w niej adresy stron docelowych, na które użytkownik ma trafić po kliknięciu danego banera. W naszym przykładzie użytkownik po kliknięciu banera „Darmowe obiady” trafi na stronę <http://helion.pl/>, po kliknięciu „Java na gorąco” — na <http://java.pl/>, a po kliknięciu „Uniwersalny eliksir na zgagę” — na <http://microsoft.com/poland> (proszę spojrzeć na rysunek 4.13). Rzecz jasna, adresy nie są z naszej strony żadnym komentarzem.

Aby dodać łączy do cyklicznie podmienianych banerów:

1. `window.onload = initBannerLink;`

Po zakończeniu ładowania strony wywoływana jest funkcja `initBannerLink()`.

2. `if (document.getElementById("adBanner").`
`↳parentNode.tagName == "A") {`
`document.getElementById("adBanner").`
`↳parentNode.onclick = newLocation;`
`}`
`rotate();`

Ten kod, znajdujący się w funkcji `initBannerLink()`, sprawdza najpierw, czy element `adBanner` znajduje się wewnątrz znacznika łączy. Jeżeli tak jest, po jego kliknięciu wywoływana będzie funkcja `newLocation()`. Na koniec wywoływana jest funkcja `rotate()`.

Skrypt 4.16. Kod HTML wymagany do wyświetlania banerów

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Cyklicznie zmieniane banery
  ↳z łączy</title>
  <script src="script08.js"></script>
  <link rel="stylesheet" href="script01.css">
</head>
<body>
  <div class="centered">
    <a href="linkPage.html"></a>
  </div>
</body>
</html>
```

Skrypt 4.17. Ten skrypt jest przykładem tego, jak można zmienić zwykłe cyklicznie podmieniane banery w prawdziwe reklamy

```
Skrypt
window.onload = initBannerLink;

var thisAd = 0;

function initBannerLink() {
  if (document.getElementById("adBanner").
↳parentNode.tagName == "A") {
    document.getElementById("adBanner")
↳parentNode.onclick = newLocation;
  }

  rotate();
}

function newLocation() {
  var adURL = new Array("negrino.com",
↳"sun.com", "microsoft.com");
  document.location.href = "http://www."
↳adURL[thisAd];
  return false;
}

function rotate() {
  var adImages = new Array("images/
↳banner1.gif", "images/
↳banner2.gif", "images/banner3.gif");
  ↳thisAd++;
  if (thisAd == adImages.length) {
    thisAd = 0;
  }
  document.getElementById("adBanner")
↳.src = adImages[thisAd];
  setTimeout(rotate, 3 * 1000);
}
```



Rysunek 4.13. Każdy z tych trzech rysunków jest łączem, po kliknięciu którego zostaniemy skierowani do różnych stron WWW

```
3. function newLocation() {
    var adURL = new Array("negrino.com",
        "sun.com", "microsoft.com");
```

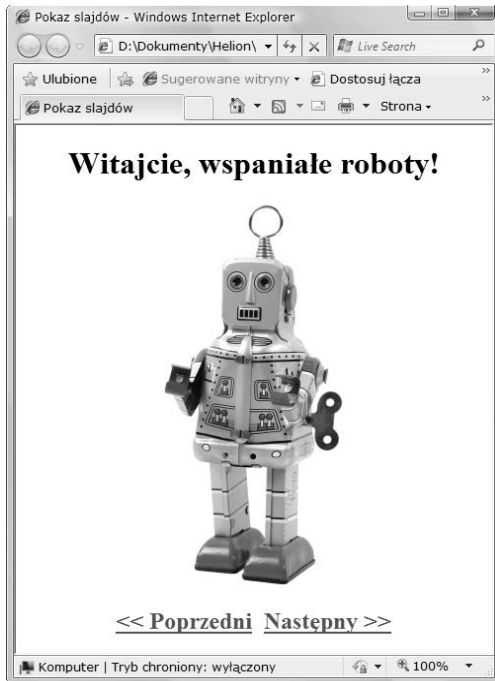
W nowej funkcji `newLocation()` do zmiennej `adURL` przypisujemy tablicę zawierającą trzy elementy. Znajdują się w niej tylko nazwy domenowe stron docelowych — pełny adres URL będzie tworzony w innym miejscu programu.

```
4. document.location.href =
    ↪ "http://www." + adURL[thisAd];
    return false;
```

Funkcja `newLocation` przypisuje obiektowi `document.location.href` (czyli innymi słowy, bieżącemu oknu przeglądarki) łańcuch znaków `http://www.` (należy zwrócić uwagę na kropkę), do którego zostaje dodana wartość zmiennej `banerURL`. Zmienna `banerURL` jest tablicą, a zatem należy określić jej element. W naszym przykładzie robimy to za pomocą zmiennej `thisAd`, która przechowuje numer aktualnie wyświetlanego banera reklamowego. Dzięki temu oglądający zostanie skierowany na różne strony, w zależności od banera, który kliknie. Na koniec zwracana jest wartość `false`, *zakazująca* przeglądarce ładowania strony spod adresu zapisanego we właściwości `href`. W naszym skrypcie zrobiliśmy już wszystko, co trzeba, więc takie dodatkowe ładowanie nie jest konieczne.

Wskazówka

- Aby skrypt działał poprawnie, tablica `adURL` musi mieć dokładnie tyle samo elementów, ile tablica `adImages`.



Oto sposób tworzenia pokazu slajdów:

1. `window.onload = initLinks;`

Po zakończeniu ładowania strony zostanie wywołana funkcja `initLinks()`.

```
2. function initLinks() {
    document.getElementById(
        "prevLink").onclick =
        processPrevious;
    document.getElementById(
        "nextLink").onclick =
        processNext;
}
```

Definiujemy tu funkcje obsługi zdarzenia `onclick` dla łączy *Poprzedni* i *Następny*.



Rysunek 4.14. Kliknięcie łączy *Poprzedni* lub *Następny* spowoduje wyświetlenie odpowiednio poprzedniego lub następnego obrazka

```

3. function processPrevious() {
    if (thisPic == 0) {
        thisPic = myPix.length;
    }
}

```

Za pomocą tej funkcji przewijamy pokaz slajdów do tyłu. Na początku sprawdzane jest, czy zmienna `thisPic` ma wartość zero. Jeżeli tak, zmiennej przypisywana jest liczba obrazków w tablicy `myPix`.

```

4. thisPic--;
document.getElementById
  ↪("myPicture").src = myPix[thisPic];

```

Pierwszy wiersz zmniejsza wartość zmiennej `thisPic` o 1, a następny ustala wartość właściwość `src` obiektu `myPicture`, pobierając ją z tablicy `myPix` z pozycji wskazywanej przez zmienną `thisPic`.

```

5. thisPic++;
if (thisPic == myPix.length) {
    thisPic = 0;
}
document.getElementById
  ↪("myPicture").src = myPix[thisPic];

```

Ten kod znajduje się w funkcji `processNext()`. Przewija on pokaz slajdów do przodu, czyli w kierunku przeciwnym niż funkcja `processPrevious()`. Na początku powiększa wartość zmiennej `thisPic` o 1, następnie sprawdza, czy wartość jest równa liczbie elementów w tablicy. Jeżeli tak, zmienna `thisPic` otrzymuje wartość zero. Na koniec odpowiednia wartość wpisywana jest do właściwości `src` obiektu `myPicture`.

Skrypt 4.20. Prosty kod HTML stanowi podstawę dla wyświetlania losowego obrazka

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Losowy obrazek</title>
  <script src="script10.js"></script>
  <link rel="stylesheet" href="script01.css">
</head>
<body>
  
</body>
</html>
```

Skrypt 4.21. Skrypt wyświetlający losowe obrazki. Wykorzystana w nim została funkcja *Math.random*, generująca liczby losowe

```
Skrypt
window.onload = choosePic;

function choosePic() {
  var myPix = new Array("images/lion.jpg",
    "images/tiger.jpg", "images/bear.jpg");
  randomNum = Math.floor((Math.random() *
    myPix.length));
  document.getElementById("myPicture").src =
    myPix[randomNum];
}
```

Losowe wyświetlanie obrazków

Jeżeli na stronie znajduje się dużo grafiki lub prowadzimy galerię sztuki cyfrowej, prawdopodobnie zainteresuje nas możliwość wyświetlania obrazka losowo wybranego z kolekcji przy każdorazowym odwiedzeniu strony. Ponownie okaże się pomocny JavaScript. Sposób realizacji zadania przedstawiony został w skryptach 4.20 (HTML) i 4.21 (JavaScript). Na rysunku 4.15 można zobaczyć efekt działania skryptu. W tym przypadku wyświetlane są zdjęcia lwa, tygrysa i niedźwiedzia (o retyle!).

Oto sposób na wyświetlanie losowo wybranego obrazka:

1. `var myPix = new Array("images/lion.jpg", "images/tiger.jpg", "images/bear.jpg");`

Łatwo się domyślić, że wewnątrz funkcji `choosePic()` trzeba utworzyć tablicę trzech obrazów zapisaną w zmiennej `myPix`.

2. `randomNum = Math.floor((Math.random() * myPix.length));`

Zmienna o nazwie `randomNum` otrzymuje wartość zapisanego tu wyrażenia matematycznego. Funkcja `Math.random()` generuje liczbę losową z przedziału od 0 do 1, która jest mnożona przez wartość `myPix.length` (oznacza ona liczbę elementów w tablicy). Funkcja `Math.floor()` powoduje zaokrąglenie wyniku, dzięki czemu do zmiennej `randomNum` zapisana zostanie liczba losowa z zakresu od 0 do 2.

```
3. document.getElementById
   ↳ ("myPicture").src = myPix[randomNum];
```

W tym wierszu źródło obiektu myPicture zostaje pobrane z tablicy myPix, z której, w zależności od wartości zmiennej randomNum, wybierana jest określona wartość.



Rysunek 4.15. W zależności od wartości wylosowanej liczby prezentowane są zdjęcia lwa, tygrysa lub niedźwiedzia

Skrypt 4.22. W tym pliku HTML zastosowano małą plik GIF, który tymczasowo zajmuje miejsce przeznaczone na banery

```
Skrypt
<!DOCTYPE html>
<html>
<head>
  <title>Losowo wybrany pierwszy baner</title>
  <script src="script11.js"></script>
  <link rel="stylesheet" href="script01.css">
</head>
<body>
  <div class="centered">
    
  </div>
</body>
</html>
```

Cykliczna zmiana obrazów z losowym obrazem początkowym

Kiedy mamy do dyspozycji wiele obrazów i chcielibyśmy je wyświetlać cyklicznie, można się pokusić o losowe wybieranie pierwszego z wyświetlanych. Skrypt 4.22 zawiera kod HTML strony, a skrypt 4.23 jest połączeniem kodu JavaScript wykorzystywanego już wcześniej do podmiiany banerów i losowania obrazka.

Skrypt 4.23. Ten skrypt pozwala na rozpoczęcie pokazu banerów od losowego obrazka

```
Skrypt

window.onload = choosePic;

var adImages = new Array("images/reading1.gif", "images/reading2.gif", "images/reading3.gif");
var thisAd = 0;

function choosePic() {
  thisAd = Math.floor((Math.random() * adImages.length));
  document.getElementById("adBanner").src = adImages[thisAd];

  rotate();
}

function rotate() {
  thisAd++;
  if (thisAd == adImages.length) {
    thisAd = 0;
  }
  document.getElementById("adBanner").src = adImages[thisAd];

  setTimeout(rotate, 3 * 1000);
}
```

A tak trzeba rozpocząć pokaz od losowo wybranego banera:

```
1. var adImages = new Array("images/  
   ↳reading1.gif","images/reading2.gif",  
   "images/reading3.gif");
```

Podobnie jak w poprzednich przykładach, definiujemy tablicę z obrazkami i przypisujemy ją do zmiennej.

```
2. function choosePic() {
```

Ta funkcja jest dokładnie taka sama jak funkcja `choosePic()` ze skryptu 4.21. Wszystkie wyjaśnienia można znaleźć właśnie tam.

```
3. function rotate() {
```

Ta funkcja jest dokładnie taka sama jak funkcja `rotate()` ze skryptu 4.15. Wszystkie wyjaśnienia można znaleźć właśnie tam.

A

adres
 e-mail, 189
 serwera, 236
 URL pliku, 351
 adresy interesujących stron, 509
 Ajax, Asynchronous JavaScript and XML, 24, 344, 374, 512
 Ajaxian, 512
 aktualizowanie tablicy, 78
 aktualizowanie strony, 386
 aktualny obrazek, 103
 analiza składniowa, 432
 analiza danych z serwera, 357, 359
 animacja, 101
 animowane pliki GIF, 120
 animowany baner reklamowy, 97
 aplet Javy, 20, 474
 aplikacja
 Gmail, 26
 Google Calendar, 26
 Google Documents, 26
 Google Maps, 26
 aplikacje interaktywne, 24
 arkusz stylów, *Patrz* CSS
 arytmetyka bitowa, 86
 ASCII, 322
 atrybut
 action, 155
 alt, 100
 autocomplete, 376
 class, 34, 85, 114, 160, 388
 className, 86
 href, 57, 83, 102
 id, 34, 43, 60
 language, 40
 readonly, 231
 rel, 335
 src, 42, 102, 104
 style, 86

target, 134
 type, 40
 value, 60
 z-index, 296
 automatyczne
 aktualizacje koloru, 388
 odświeżenie, 429
 określenie wartości pola, 174
 uzupełnianie, 376, 392

B

baner, 120
 baner będący łączem, 122
 bezpieczeństwo, 23
 bezpieczeństwo IE, 451
 BBEdit, 36
 biblioteka jQuery, 382, 394
 biblioteka jQuery UI, 395
 binarna reprezentacja liczby, 92
 bingo, 69
 bit, 86
 blok, *Patrz* sekcja
 blokowanie wyskakujących okienek, 145
 błędy składni, 145
 bufor przeglądarki, 44, 101

C

CDN, Content Delivery Network, 385
 CGI, Common Gateway Interface, 149
 ciasteczko, 235
 data wygaśnięcia, 338, 339
 domena witryny, 237
 nazwa, 237
 odczytywanie, 241
 parametry opcjonalne, 237
 ścieżka URL, 237
 usuwanie, 247
 wartość, 238
 wyświetlanie, 242

CSS, Cascading Style Sheets, 25, 33, 68, 84, 86, 307, 345, 501
 czas ważności ciasteczka, 338
 czynnik ekranowy, 100

D

dane dynamiczne, 324
 data, 280
 data odwiedzin strony, 251
 data wygaśnięcia ciasteczka, 239
 debugger Firebug, 514
 debugowanie skryptów, 510
 definiowanie
 ciasteczka, 237
 nagłówków żądań, 356
 zachowań elementów, 35
 deklaracja tablicy, 75
 DHTML, 58
 diagram obiektów, 479
 dodatki biblioteki jQuery, 417, 438
 dodawanie
 biblioteki jQuery, 383
 elementów do strony, 430
 kalendarza, 421, 423
 łączy, 122
 odtwarzacza dźwięku, 439
 parametrów do okien, 146
 tekstu, 271
 węzłów, 260

dokumentacja JavaScript, 510
 DOM, Document Object Model, 25, 258
 dostęp do dowolnego elementu, 416
 dostęp do kanału informacyjnego, 430
 Dreamweaver, 36, 154
 duplikaty liczb, 79
 dynamiczne
 elementy iframe, 139
 modyfikowanie menu, 156
 wyświetlanie, 345
 wyświetlenie daty, 281
 dziwne imię, 322

E

edytor tekstu, 36
 efekt animacji, 402
 efekty, 399
 element
 blokowy, 34
 chartArea, 331
 div, 34, 310

document, 384
 iframe, 131, 134, 139
 nadrzędny, 142
 nadrzędny, parent, 166
 span, 34, 422
 wierszowy, 34

F

Firebug, 514
 format
 adresu e-mail, 184
 JSON, 25, 277, 367, 370
 XHTML, 346
 XML, 25, 345, 353
 formatowanie ciągów znaków, 199
 formatowanie przepływu danych, 345
 formularz, 149, 158
 kontrola poprawności, 160, 170
 pola wymagane, 158
 typowe elementy, 167
 funkcja, 41
 \$.ajax(), 417
 \$.getJSON(), 417, 430
 addOnload(), 215
 anonimowa, 105, 106, 152, 430
 anotherCard(), 83
 append(), 386
 attr(), 388
 checkWin(), 87
 chgChart(), 325, 328
 clickHandler(), 313
 confirm(), 247
 cookieVal(), 252
 crossCheck(), 164, 170
 css(), 390
 doAction(), 275
 fieldCheck(), 229, 230
 floor, 70
 getActiveStylesheet(), 338, 340
 getNewFile(), 349
 getPix(), 364
 getPixVal(), 358, 365
 getPreferredStylesheet(), 337
 getPreview(), 372
 getText(), 354
 hidePreview(), 372
 hover(), 390, 391
 init(), 275
 initAll(), 69, 371
 initBannerLink(), 122

initDate(), 280
 initLinks(), 141
 initStyle(), 337
 invalidLabel(), 166
 isNum(), 178
 jsonFlickrFeed(), 368
 jumpPage(), 153
 makeChoice(), 380
 makeRequest(), 350
 Math.floor(), 365
 Math.random(), 127, 142, 365
 mouseout(), 391
 moveHandler(), 223
 moveup(), 219
 newCard(), 83
 newCheck(), 253
 newFunction(), 215
 newLocation(), 122
 newSlide(), 318
 newWindow(), 144, 226
 newWinLinks(), 144
 nodeChanger(), 266, 276
 obsługi zdarzenia, 43
 open(), 351
 parseDate(), 425
 processNext(), 126
 processPrevious(), 126
 radioPicked(), 172
 ready(), 384
 rollOut(), 119
 rollover(), 119
 rolloverInit(), 102, 103
 rotate(), 121, 130
 saySomething(), 60
 searchSuggest(), 377
 send(), 351
 setActiveStylesheet(), 341
 setBanner(), 142
 setContent(), 136
 setCookie(), 238, 240
 setTimeout(), 215, 364
 setupRollover(), 103, 114
 showAmPm(), 291
 showPictures(), 357
 showTheDays(), 294
 showTheDaysTill(), 292
 split(), 393
 toggleColor(), 84, 87
 toggleMenu(), 305, 309, 313
 twitDataCallback(), 430
 validEmail(), 192

validBasedOnClass(), 170
 validEmail(), 181
 validTag(), 159
 window.getSelection(), 471
 writeContent(), 138
 funkcje
 literałów obiektów, 274
 matematyczne, 467
 obsługi zdarzeń, 30, 211
 wspólne, 141
 wyszukiwania, 188
 z przekazywanym parametrem, 384
 zwracające wartość, 77

G

Garret Jesse James, 25
 generator dziwnych imion, 319
 generator wykresów, 324, 326
 GMT, Greenwich Mean Time, 284
 grupowanie przycisków opcji, 172

H

hasło, 164
 hierarchia okien przeglądarki, 133
 HTML, 25, 345

I

identyfikacja komputera, 236
 identyfikator, 34
 adBanner, 121
 bodyText, 397
 flyer, 118
 helloMessage, 43
 komórki, 72
 łącza, 112
 na stronie, 241
 obrazka, 112
 reload, 82
 informowanie o nowościach na stronie, 251
 inicjacja skryptu, 82
 inspektor DOM, 39, 479
 instrukcja
 break, 61
 catch, 63
 if, 49, 73
 return false, 148
 switch, 59
 try, 62, 467

instrukcja
 warunkowa, 49, 73
 while, 81
 wielopoziomowa, 59
 instrukcje
 if/then/else, 49, 73
 switch/case, 59
 try/throw/catch, 62, 467
 interakcja z biblioteką jQuery, 390
 interfejs użytkownika, 418
 interpreter JavaScript, 54
 ISO Latin, 461

J

JavaScript, 58
 język
 C, 19
 C#, 19
 C++, 19
 Java, 19
 JavaScript, 474
 JScript, 21, 475, 511
 LiveScript, 21
 obiektowy, 27
 Perl, 23
 PHP, 23
 skryptowy, 18, 474
 XHTML, 134
 JSON, JavaScript Object Notation, 25, 277, 367, 370

K

kalendarz, 421
 kalkulator, 467
 kanał informacyjny, 429, 432
 karta Roll Your Own, 419
 kaskadowe arkusze stylów, 501
 klasa, 34
 invalid, 166
 menuLink, 305
 newWin, 144
 over, 410
 sortUp, 413
 tr.even, 410
 winningBG, 90
 kod
 HTML, 33, 53
 konfigurujący obrazki, 368
 odczytujący pliki z serwera, 349
 pocztowy, 178

przekierowania, 54
 skryptozaładki, 445
 skryptozaładki w funkcji, 455
 XHTML, 33
 kody przypisane klawiszom, 234
 kolejność wywołań open(), 366
 kolor etykiety, 165
 kolor nagłówka, 387
 komentarz, 45
 jednowierszowy, 46
 wielowierszowy, 46
 kompozycja dowolna, 406
 kompozycja Redmond, 404
 komunikat, 47
 o błędzie, 194
 o nowościach, 251
 konfigurowanie łączy, 141
 konsorcjum W3C, 33, 258, 470
 kontrola poprawności kodu pocztowego, 177
 kontrola wyboru, 174
 kontrolka ActiveX, 356
 konwersja
 czasu, 290
 wartości, 466
 wartości RGB, 464
 krok inicjacji, 70
 krok inkrementacji, 70
 krok ograniczenia, 70

L

lewy ukośnik, 138, 190
 liczby losowe, 76
 licznik dni, 292, 293
 licznik wejść na stronę, 244
 lista
 podpowiedzi do zapytania, 380
 przeglądarek, 347
 słów kluczowych, 31
 z obrazami, 427
 listy wypunktowane, 310
 literały obiektów, 273, 277
 losowe wybieranie banera, 130
 losowe wyświetlanie obrazków, 127

Ł

ładowanie
 dynamicznego elementu iframe, 139
 ramek iframe, 136
 strony do ramki, 136

łańcuch znaków `http://www.`, 123
 łańcuchy tekstowe, 75
 łącze, 55, 109, 304
 na stronie, 135
 typu `mailto`, 471
 łączenie tekstu, 31

M

Macromedia Flash, 20, 347
 manipulacja węzłami, 257, 270
 mechanizm
 Ajax, Ajax engine, 346
 ciasteczek w przeglądarce, 241
 menu
 harmonijkowe, `accordion menu`, 400
 modyfikowane dynamicznie, 156
 o tradycyjnym wyglądzie, 307
 rozwijane, 151, 156, 307, 311
 skoków, `jump menu`, 154
 wysuwane, 304, 305
 metaznaki, 192
 metoda, 28
 `accordion()`, 402, 403
 `actionType()`, 275
 `alert()`, 48
 `appendChild()`, 260, 269
 `blur()`, 148
 `charCodeAt()`, 321
 `confirm()`, 49
 `docBody.removeChild()`, 263
 `document.write()`, 44
 `effect()`, 398
 `focus()`, 148
 `getDay()`, 281
 `getElementById()`, 43, 60, 72, 479
 `getFullYear()`, 295
 `getHours()`, 283
 `getTime()`, 295
 `hide()`, 397
 `indexOf()`, 182, 378
 `insertBefore()`, 269
 `isNaN()`, 63
 `Math.ceil()`, 295
 `parse()`, 287
 `parseInt()`, 157, 244, 287
 `prompt()`, 51
 `replace()`, 133, 197
 `replaceChild()`, 270
 `setTimeout()`, 300
 `show()`, 398
 `sort()`, 204

`split()`, 196, 238
`substring()`, 253, 254, 255
`tablesorter()`, 412
`toggle()`, 398
`toGMTString()`, 239
`window.open()`, 457
 wykrywania obiektów, 74
 metody obiektu `Date`, 301
 modalne okno dialogowe, 405
 model DOM, 258, 345, 479
 moduł CGI, 23
 modyfikator `i`, 195
 modyfikatory wyrażeń regularnych, 192
 modyfikowanie
 arkusza stylów, 333
 atrybutów CSS, 84
 drzewa, 29
 zawartości tablicy, 78

N

NaN, Not a Number, 468
 narzędzia pastebin, 515
 narzędzie
 JSBin, 515
 JSFiddle, 515
 JSLint, 515
 ThemeRoller, 418
 nawias kwadratowy, 190
 nawias okrągły, 190
 nawigacja po stronie, 155
 nazwa
 ciasteczka, 238
 okna, 144
 pliku obrazka, 108
 podmienianego rysunku, 119
 zbioru przycisków, 172

O

obiekt, 27
 `Date`, 301
 `document`, 220
 `document.getElementById`, 74
 `document.layers`, 216
 `document.oncontextmenu`, 222
 `expireDate`, 239
 `RegExp`, 201
 `this`, 60
 `thisLink`, 117
 `window`, 350
`XMLHttpRequest`, 25, 345, 350, 376, 417

obiekty języka JavaScript, 485
 obrazek zmiennik, 99
 obrazki GIF, 106
 obrazki PNG, 106
 obsługa
 błędów, 62
 czasu, 281
 elementów formularzy, 167
 kodu JavaScript, 101
 menu, 312
 obiektu XMLHttpRequest, 350
 wielu ciasteczek, 249
 zdarzeń, 30
 zadań, 377
 oczy śledzące kursor myszy, 223
 odczytywanie ciasteczka, 241
 odczytywanie danych z serwera, 349
 odsyłacz do obrazka, 99
 odświeżanie danych z serwera, 364
 odtwarzacz dźwięku, 439, 443
 okienka wyskakujące, pop-up windows, 145
 okno, 133
 parent, 133
 podglądu, 372
 z komunikatem, 48
 operacja and, 86, 92
 operacja or, 88
 operatory, 31
 otwieranie nowego okna, 144
 oznaczanie elementów, 297

P

parametr resizable, 405
 parametry, 51
 pętla
 do/while, 80
 loop, 66, 70
 wypełniająca tabelę, 69
 plik
 index.html, 420
 jquery.js, 396, 410
 jquery.min.js, 384
 jquery-ui.js, 396
 lilGreen.gif, 330
 sansStyle.css, 336
 serifStyle.css, 336
 pliki
 CSS, 33, 68, 99, 332
 GIF, 90
 HTML, 33

JavaScript, 33
 JSON, 369
 tekstowe, 349
 XML, 349
 XML pobierane z serwera, 355
 zewnętrzne .js, 101, 107
 .css, 36
 .html, 36
 .js, 36, 42
 pobieranie
 danych, 358
 danych z serwera, 367
 odpowiedzi od użytkownika, 49
 podgląd dynamiczny, 419
 podgląd łączy, 371
 podmienianie
 arkusza stylów, 333
 banerów, 129
 obrazka przy użyciu łączy, 110
 obrazka z różnych łączy, 112
 rysunku, 119
 tekstu, 271
 węzłów, 270
 wielu obrazków, 115
 podmieniany obrazek, rollover, 97
 pokaz slajdów, 97, 124, 232
 pokaz slajdów z opisem, 317
 pola wymagane, 158
 pole adresu e-mail, 230
 pole formularza, 375
 polecenie
 open(), 146
 setTimeout(), 121
 split("; "), 242
 pomoc, 516
 porównania, 32
 porównywanie wartości pól, 163
 portal My Yahoo!, 26
 pozycjonowanie elementu, 135
 prezentacja, 124
 prezentacja strony, 33
 prędkość wyświetlania strony, 121
 problemy z buforowaniem, 154
 program
 BBEdit, 36
 Dreamweaver, 36, 154
 Emacs, 36
 Firebug, 514
 Macromedia Flash, 20, 347
 TextEdit, 36
 TextWrangler, 36

programy strony klienta, 22
 programy strony serwera, 22
 progresywne ulepszanie, progressive enhancement, 58
 przechwytywanie błędów, 468
 przeciąganie i upuszczanie elementów, 382, 426, 427
 przeglądanie obrazków, 459
 przeglądarka
 Camino, 474
 Chrome, 44
 Firefox, 29
 Internet Explorer, 29
 MSIE, 476
 Navigator, 21
 Netscape, 44
 Safari, 29
 przekazywanie wartości do funkcji, 71, 72
 przekierowanie, redirect, 53
 przesuwanie obiektu, 299
 przesyłanie danych, 347
 przycisk
 animowany, 22
 opcji, 171
 Przejdź, 151
 trójstanowy, 107, 108
 Wstecz, 348
 przypisania, 32

Q

QuirksMode, 512

R

ramka, 134
 iframe, 137
 potomna, 133
 referencja własnego obiektu, 397
 reklama, 298
 przesuwanie, 299
 ukrywanie, 298
 wyświetlanie, 298
 RGB, 464
 rollover, 22, 97
 różne wersje kodu, 458

S

sekcja
 <body>, 18, 416
 <head>, 18, 416
 default, 61, 162
 do, 81

finally, 63
 try{}, 468
 serwer w trybie asynchronicznym, 346
 serwis Flickr, 360
 serwis Google Maps, 24
 serwis Google Suggest, 380
 składnia kropkowa, dot syntax, 28
 skracanie adresów URL, 469
 skryptozakładki, bookmarklets, 445, 447
 skrypty
 CGI, 149, 192
 DOM, 58
 nienatarczywe, unobtrusive scripting, 58
 wewnętrzne, 42
 zewnętrzne, 42
 słowa kluczowe JavaScriptu, 497
 słownik, 456
 słowo invalid, 158, 166
 słowo kluczowe
 function, 41
 this, 56, 57
 var, 51
 słowo slow, 399
 sortowanie, 203
 malejące, 414
 rosnące, 413
 sortowanie tabel, 411
 według kolumny, 414
 spacja, 145
 sprawdzanie
 adresu e-mail, 181, 189
 adresu URL, 195
 daty, 282
 kodów pocztowych, 177
 liczb, 80
 nazwy pliku, 194
 poprawności stron, 470
 pory dnia, 283
 pól, 85
 wartości, 85
 standard ECMAScript, 473, 477
 stopień złożoności kodu CSS, 420
 strefa czasowa, 285
 strona z formularzem, 231
 strony dynamiczne, 279
 struktura dokumentu, 29, 33
 struktura drzewiasta, 29, 259
 styl
 elementów, 99
 elementów odtwarzacza, 440
 menu, 311

styl

- okienka podglądu, 372
- podstawowy strony, 335
- wykresu, 325

Ś

- ścieżka do obrazka, 102
- ścieżka URL w ciasteczku, 237

T

tabela, 67

- obiektów, 485
- standardowa, 408

tablica, array, 75

- adImages, 123
- allClasses, 160
- ciasteczek, 242
- ciągów znaków, 93
- dayName, 281
- monthDays, 157
- overImage, 117
- przechowująca obrazki, 117
- usedWords, 94
- userNums[], 81
- winners, 90
- znaczników, 159

technologia

- Ajax, 24, 344, 374, 512
- Flash, 438

testowanie skryptów, 146

tezaurus, 456

tryb asynchroniczny, 351

twarda spacja, 67

tworzenie

- animowanych banerów, 120
- atrybutu, 454
- banerów, 120
- ciasteczka, 238
- elementu iframe, 134
- klasyczne stron, 34
- kodu HTML, 33
- menu, 152
- niestandardowej kompozycji, 419
- nowego okna, 143
- okna dialogowego, 404
- pasiastej tabeli, 409
- podglądu, 371
- podmienianych obrazków, 100, 101
- pokazu slajdów, 124

- pola formularza, 375
- przycisku trójstanowego, 108
- skryptozaładki, 446–448, 450
- struktury, 345
- węzłów tekstowych, 260
- zawartości ramki iframe, 138

typ Boolean, 86

typy wartości, 31

U

układ zawartości strony, 428

ukrywanie obiektu, 298

uruchamianie skryptu, 82

user agent, 74

ustawianie elementu docelowego, 135

usunięcie strony z pamięci podręcznej, 152

usuwanie

- akapitów, 266
- ciasteczek, 247
- elementów HTML, 416
- skryptozaładki, 447
- tekstu, 271
- węzłów, 262
- wybranego węzła, 264
- znaków białych, 81

UTC, Coordinated Universal Time, 284, 301

W

W3C, *Patrz* konsorcjum W3C

warstwy, 296

wartości, 31

wartość

- ciasteczka, 238
- null, 51
- pusta (""), 157
- winningBG, 90

węzeł, node, 29, 259

- elementu, element node, 29, 259
- tekstowy, text node, 29, 259

węzły

- dodawanie, 260
- podmienianie, 270
- usuwanie, 262
- wstawianie, 267

widget datepicker, 421, 424

widget obsługujący datę, 422

wielkość liter, 31

witryna Stack Overflow, 516

- właściwości
 - arkuszy CSS, 501
 - dźwiękowe, 501
 - obiektów, 28
 - obiektu RegExp, 201
 - prevWin.innerHTML, 373
- właściwość
 - event, 85
 - evt, 85
 - firstChild, 377
 - imgToChange, 117
 - innerHTML, 44, 243, 257
 - message, 63
 - outImage, 104
 - self.location, 133
 - src, 142
 - srcElement, 85
 - style.left, 300
 - tagName, 103, 106
 - target, 135
 - this.imgToChange.length, 119
 - this.imgToChange.src, 111
 - this.src, 111
 - thisPage, 138
 - xhr.status, 373
- współużytkowanie pliku, 141
- wstawianie węzła, 267
- wtyczka, plug-ins, 474
- wtyczka tablesorter, 411, 414
- wybór samochodu, 168
- wykres słupkowy, 329, 330
- wykresy, 324
- wykrywanie obiektów, 73, 82
- wykrywanie przeglądarki, 74
- wymuszenie typu MIME, 356
- wypełnianie pól formularza, 159
- wyrażenie regularne, regular expression, 187
 - formatowanie ciągu znaków, 200
 - podmiana elementu, 209
 - sortowanie ciągów znaków, 204
 - sprawdzanie nazwy pliku, 194
 - sprawdzanie wartości, 205
 - sprawdzenie adresu e-mail, 189
 - sprawdzenie adresu URL, 195
- wyrażenie
 - \S+, 195
 - \w, 190
 - \w+, 191
- wyróżnianie
 - elementu, 396
 - pól, 165
- wiersza, 409
- znaku, escaping, 190
- wysyłanie stron e-mailem, 471
- wyszukiwanie ciągów znaków, 196
- wyszukiwanie słowa, 456
- wyświetlanie
 - ciasteczek, 242
 - losowych obrazków, 127
 - stylów CSS, 310
 - tabeli, 413
 - znaków ISO Latin, 461
- wywołanie void(), 455
- wywołanie zwrotne, 370
- wywoływanie skryptu, 82
- wzorzec, 192

X

XHTML, 346
XML, 25, 345, 353

Z

zabezpieczanie strony, 133
zachowanie strony, 33
zakaz ładowania strony, 123
zamienianie czcionek, 335
zasięg zmiennej, 52
zawartość kanału informacyjnego, 433
zawartość okna, 147
zaznaczanie pola, 161
zdarzenia, 30

- formularzy, 228
- klawiatury, 232
- okien, 212

zdarzenie

- document.onmousedown, 222
- mouseover, 409
- onabort, 217
- onblur, 148, 219, 229, 238
- onchange, 228
- onclick, 30, 54, 138, 229, 378
- oncontextmenu, 220
- ondblclick, 226
- onfocus, 148, 218, 231
- onkeydown, 232
- onkeypress, 234
- onkeyup, 234
- onload, 30, 152, 213, 215
- onmousedown, 220
- onmousemove, 223

zdarzenie

- onmouseout, 106, 226
- onmouseover, 106, 226
- onmouseup, 223
- onmove, 217
- onpagehide, 154
- onpageshow, 154
- onreadystatechange, 351
- onreset, 228
- onresize, 216
- onselect, 228
- onsubmit, 158, 228
- onunload, 152
- window.oncontextmenu, 222

- zestaw narzędziowy, 382, 391

- zewnętrzny arkusz stylów, 33, 316, 334

zmiana

- klasy, 86
- koloru tła, 452
- rozmiarów okna, 405
- stylów strony, 453, 454
- wielkości strony, 472
- wyglądu strony, 86

- zmienianie obrazków, 368

zmienna, 31

- allTags, 159
- banerURL, 123
- catWindow, 144
- classBack, 162
- lastVisit, 250
- newPage, 153
- radioSet, 172
- setSquares, 87, 92
- this.href, 147
- winners, 87
- winningOption, 87

zmienne

- globalne, 52
- lokalne, 52

znacznik

- <a>, 57, 103
- <body>, 29
- <h1>, 43
- <head>, 29
- <html>, 29
- <div>, 34
- , 102
- <input>, 164
- , 310
- <link>, 335
- <noscript>, 48

- <object>, 20
- <p>, 262
- <script>, 18, 39, 133
- , 34
- , 310
- form, 155
- layer, 296
- łącza, 103
- option, 154

znaczniki

- końcowe, 39
- początkowe, 39

znak

- @, 182
- biały, 196
- daszka, 190
- dodawania, 190
- dolara, 192, 383
- dwukropka, 273
- gwiazdki, 191
- kropki, 35, 190
- krzyżyka, 35
- lewego ukośnika, 138, 190
- myślnika, 190
- przecinka, 273
- średnika, 39, 192
- ukośnika, 189
- wykrzyknika, 79
- zapytania, 191

znaki

- && i ||, 90
- /*, 45
- akcentowane, 463

ż

- źródła informacji, 513

- źródło obrazka, 100, 105

ż

- zadania HTTP, 351
 - typu GET, 351
 - typu HEAD, 351
 - typu POST, 351
- żółte tło, 396

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**



PO PROSTU

Wydanie VIII

JavaScript

Przygotuj swoją pierwszą aplikację internetową!

Język JavaScript po paru latach banicji wrócił do łask – jeszcze silniejszy, jeszcze lepszy, wręcz niezbędny! To dzięki niemu współczesne aplikacje internetowe nie przypominają tych tradycyjnych, które znaliśmy do tej pory, ale są od nich o wiele lepsze! Dostępne z każdego miejsca na świecie, odporne na awarie oraz kuszące atrakcyjnym interfejsem użytkownika...

Dzięki tej książce błyskawicznie poznasz możliwości JavaScriptu oraz zaczniesz stosować go w codziennej pracy. Znajdziesz tu wszystkie niezbędne informacje, począwszy od podstaw tworzenia interaktywnej witryny internetowej, a skończywszy między innymi na dodawaniu zaawansowanych funkcji, oferowanych przez bibliotekę jQuery. Już po krótkiej lekturze tchniesz życie w swoje strony internetowe, a jeśli poświęcisz trochę więcej czasu, bez problemu stworzysz efektowną aplikację internetową!

W trakcie lektury poznasz:

- składnię języka i budowę skryptów JavaScript
- metody osiągania interakcji między kodem w HTML-u i JavaScriptcie
- możliwości i ograniczenia tego języka
- najlepsze sposoby korzystania z technologii AJAX

helion.pl
księgarnia
internetowa

Nr katalogowy: **8562**

Księgarnia internetowa:
<http://helion.pl>

Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

Sprawdź najnowsze promocje:

👉 **<http://helion.pl/promocje>**

Książki najchętniej czytane:

👉 **<http://helion.pl/bestsellery>**

Zamów informacje o nowościach:

👉 **<http://helion.pl/novosci>**

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-4271-7



Cena 69,00 zł

Informatyka w najlepszym wydaniu