helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

JavaScript – przykłady. Biblia

Autor: Danny Goodman
Tłumaczenie: Piotr Rajca
ISBN: 83-7197-671-2
Tytuł oryginału: JavaScript Examples Bible: The Essential Companion to JavaScript Bible
Format: B5, stron: 522

Niniejsza książka stanowi wspaniałe uzupełnienie JavaScript. Biblia

Książka – autora bestsellera „JavaScript. Biblia" – to dziesiątki gotowych przykładów kodu, których możesz użyć na swoich stronach internetowych. Nawet jeśli przedstawionych przykładów nie wykorzystasz bezpośrednio, to zaznajomienie się z nimi pomoże Ci poznać tajniki JavaScriptu. Możesz je porównać ze swoimi skryptami. Ogromna wiedza i umiejętności Danny'ego Goodmana mogą być dla Ciebie wspaniałym punktem odniesienia. Dodatkowo zapoznać się możesz z jego wartościowymi wskazówkami i uwagami towarzyszącymi fragmentom kodu.

Jeśli chcesz wzbogacić swoje strony o interaktywne elementy utworzone w JavaScripcie, powinieneś mieć tę książkę zawsze pod ręką. Oszczędzisz sobie w ten sposób wiele cennego czasu.

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

Spis treści

O Autorze 9

Wstęp 11

Rozdział 1. Obiekty ogólnych elementów HTML (Rozdział 15.) 15

Najważniejsze informacje o przykładach 15
Obiekty ogólne 17
Właściwości 17
Metody 63
Procedury obsługi zdarzeń 107

Rozdział 2. Obiekty window oraz frame (Rozdział 16.) 137

Informacje o przykładach 138
Obiekt window 139
Właściwości 139
Metody 162
Procedury obsługi zdarzeń 196
Obiekt elementu FRAME 198
Właściwości 198
Obiekt elementu FRAMESET 202
Właściwości 202
Obiekt elementu IFRAME 206
Właściwości 206
Obiekt popup 209
Właściwości 209
Metody 210

Rozdział 3. Obiekty location i history (Rozdział 17.) 213

Informacje o przykładach 213
Obiekt location 214
Właściwości 214
Metody 224
Obiekt history 226
Właściwości 226
Metody 226

Rozdział 4. Obiekty document oraz body (Rozdział 18.) 231

Informacje o przykładach 232
Obiekt document 232

Właściwości 232
Metody 249
Procedury obsługi zdarzeń 262
Obiekt elementu BODY 263
Właściwości 263
Metody 267
Procedury obsługi zdarzeń 267

Rozdział 5. Obiekty elementów tekstowych (Rozdział 19.) 269
Informacje o przykładach 270
Obiekt elementu FONT 270
Właściwości 270
Obiekt elementu HR 273
Właściwości 273
Obiekt elementu MARQUEE 277
Właściwości 277
Metody 279
Obiekt Range 280
Właściwości 280
Metody 282
Obiekt selection 295
Właściwości 295
Metody 296
Obiekty Text oraz TextNode 297
Właściwości 297
Metody 297
Obiekt TextRange 301
Właściwości 301
Metody 304
Obiekt TextRectangle 319
Właściwości 319

Rozdział 6. Obiekty Image, Area oraz Map (Rozdział 22.) 323
Informacje o przykładach 323
Obiekt Image oraz obiekt elementu IMG 324
Właściwości 324
Procedury obsługi zdarzeń 335
Obiekt elementu AREA 336
Właściwości 336
Obiekt elementu MAP 337
Właściwość 337

Rozdział 7. Obiekt form oraz obiekty z nim związane (Rozdział 23.) 341
Informacje o przykładach 342
Obiekt FORM 342
Właściwości 342
Metody 345
Procedury obsługi zdarzeń 346
Obiekt elementu LABEL 347
Właściwości 347

Rozdział 8. Obiekt button (Rozdział 24.) 349
Informacje o przykładach 350
Obiekt elementu BUTTON oraz obiekty Button, Submit i Reset 350

Właściwości 350
Metody 351
Procedury obsługi zdarzeń 352
Obiekt checkbox 353
Właściwości 353
Procedury obsługi zdarzeń 355
Obiekt radio 358
Właściwości 358
Procedury obsługi zdarzeń 360

Rozdział 9. Obiekty tekstowych elementów formularzy (Rozdział 25.) 363
Informacje o przykładach 364
Obiekt pola tekstowego 364
Właściwości 364
Metody 369
Procedury obsługi zdarzeń 371
Obiekt elementu TEXTAREA 374
Właściwości 374
Metody 374

Rozdział 10. Obiekty select, option oraz FileUpload (Rozdział 26.) 375
Informacje o przykładach 376
Obiekt elementu SELECT 376
Właściwości 376
Metody 382
Procedury obsługi zdarzeń 382
Obiekt elementu OPTION 384
Właściwości 384
Obiekt elementu OPTGROUP 384
Właściwości 384

Rozdział 11. Obiekty tabel i list (Rozdział 27.) 387
Informacje o przykładach 388
Obiekt elementu TABLE 388
Właściwości 388
Metody 396
Obiekty elementów TBODY, TFOOT oraz THEAD 397
Właściwości 397
Obiekty elementów COL oraz COLGROUP 397
Właściwości 397
Obiekt elementu TR 398
Właściwości 398
Obiekty elementów TD oraz TH 399
Właściwości 399
Obiekt elementu OL 401
Właściwości 401
Obiekt elementu UL 402
Właściwości 402
Obiekt elementu LI 402
Właściwości 402

Rozdział 12. Obiekt navigator oraz inne obiekty środowiskowe (Rozdział 28.) 403
Informacje o przykładach 404
Obiekt clientInformation (IE 4+) oraz navigator (wszystkie przeglądarki) 404
Właściwości 404
Metody 411
Obiekt screen 412
Właściwości 412
Obiekt userProfile 413
Metody 413

Rozdział 13. Obiekty zdarzeń (Rozdział 29.) 415
Informacje o przykładach 416
Obiekt event (NN 4) 417
Właściwości 417
Obiekt event (IE 4+) 420
Właściwości 420
Obiekt event (NN 6+) 429

Rozdział 14. Obiekty stylów i arkuszy stylów (Rozdział 30.) 441
Informacje o przykładach 441
Obiekt styleSheet 442
Właściwości 442
Metody 444
Obiekty cssRule oraz rule 446
Właściwości 446

Rozdział 15. Obiekty umiejscowione (Rozdział 31.) 449
Informacje o przykładach 450
Obiekt layer stosowany w przeglądarce NN 4 450
Właściwości 450
Metody 469

Rozdział 16. Obiekty String, Math, Number i Boolean (Rozdziały 34. i 35.) ... 475
Informacje o przykładach 476
Obiekt String 476
Właściwości 476
Metody rozbioru łańcuchów 477
Obiekt Number 489
Właściwości 489
Metody 490

Rozdział 17. Obiekt Array (Rozdział 37.) 491
Informacje o przykładach 492
Metody obiektu Array 492

Skorowidz 501

Rozdział 6.
Obiekty Image, Area oraz Map (Rozdział 22.)

W tym rozdziale:

- Jak pobierać obrazy do pamięci podręcznej i zamieniać obrazy wyświetlane na stronie
- Wykonywanie czynności bezpośrednio po pobraniu obrazu
- Tworzenie interaktywnych map odnośników obsługiwanych po stronie klienta

Obiekty elementów `IMG` są bardzo często wykorzystywane w skryptach, głównie dlatego, iż są one stosowane w rozwiązaniach polegających na zamienianiu obrazów wyświetlanych na stronie. Poza tym możliwość wykorzystania tych obiektów w skryptach była dostępna niemal już w pierwszych wersjach przeglądarek obsługujących język JavaScript. W rozwiązaniach obejmujących zamienianie obrazów pomocniczą rolę odgrywa obiekt `Image`, wykorzystywany w celu pobrania obrazu z serwera od pamięci podręcznej przeglądarki (dzięki czemu obraz wyświetlony na stronie można błyskawicznie zamienić na inny). Choć z punktu widzenia skryptów te obiekty są różne, mają wiele wspólnych właściwości i metod, dzięki czemu poznanie sposobów ich wykorzystania jest znacznie prostsze.

Obiekty elementów `AREA` oraz `MAP` ściśle ze sobą współpracują. W praktyce element `AREA` przypomina nieco obiekt `A` używany do definiowania połączeń. Oba elementy tworzą bowiem na stronie „miejsca", które użytkownik może kliknąć, co zazwyczaj powoduje przejście na inną stronę witryny lub do dowolnego innego miejsca Sieci. Te obiekty mają także wiele wspólnych właściwości związanych z adresami URL.

Informacje o przykładach

- Większość wersji przeglądarki Internet Explorer jest w stanie wyświetlać elementy `IMG` zawierające zarówno obrazy nieruchome, jak i „ruchome" (na przykład, klipy wideo zapisane w formacie MPEG). Przykład przedstawiony na listingu 22.3 pokazuje w jaki sposób, przy użyciu właściwości `dynsrc`, można zastąpić obraz stały obrazem ruchomym.

- Strona przedstawiona na listingu 22.4 pozwala na porównanie efektywności zamieniania obrazów przechowywanych w pamięci podręcznej przeglądarki i pobieranych bezpośrednio z serwera. Skrypt pokazuje także w jaki sposób można cyklicznie zamieniać obrazy w określonych odstępach czasu.
- Listing 22.5 pokazuje w jaki sposób można wykonywać czynności w odpowiedzi na zdarzenie `onLoad` obrazu.
- Na listingu 22.7 został przedstawiony skrypt o bardzo dużych możliwościach, pokazujący w jaki sposób, po zmianie obrazu wyświetlanego w elemencie `IMG`, można zastosować nową mapę odnośników obsługiwaną po stronie przeglądarki.

Obiekt Image oraz obiekt elementu IMG

Właściwości

align

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność				✓			✓	✓	✓

Przykład

Strona przedstawiona na listingu 22.1 pozwala na przypisywanie różnych wartości właściwości `align`, określającej położenie wewnątrzwierszowego obrazu w stosunku do sąsiadującego z nim tekstu. Spróbuj zmieniać wielkość okna przeglądarki, aby tekst wyświetlany na stronie był w różny sposób dzielony i przyjrzyj się, jaki wpływ na układ strony mają wybierane wartości wyrównania obrazu. Nie wszystkie przeglądarki udostępniają unikatowe możliwości wyrównania odpowiadające wszystkim podanym opcjom, a zatem spróbuj przetestować stronę w wielu różnych przeglądarkach.

Listing 22.1. *Testowanie właściwości align obrazów*

```
<HTML>
<HEAD>
<TITLE>Właściwość align obiektu IMG</TITLE>
<SCRIPT LANGUAGE="JavaScript">

function setAlignment(sel) {
    document.myIMG.align = sel.options[sel.selectedIndex].text
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Właściwość align obiektu IMG</H1>
<HR>
<FORM>
Wybierz sposób wyrównania:
<SELECT onChange="setAlignment(this)">
```

```
    <OPTION>absbottom
    <OPTION>absmiddle
    <OPTION>baseline
    <OPTION SELECTED >bottom
    <OPTION >left
    <OPTION>middle
    <OPTION>right
    <OPTION>texttop
    <OPTION>top
</SELECT>
</FORM>
<HR>
<P>Lorem ipsum dolor sit amet, consectetaur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. <IMG NAME="myIMG" SRC="desk1.gif" HEIGHT=90 WIDTH=120>
Ut enim adminim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.</P>
</BODY>
</HTML>
```

alt

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność				✓			✓	✓	✓

Przykład

Określanie wartości właściwości `alt` obrazów można wypróbować na stronie *Tester* (przedstawionej w rozdziale 13. książki „JavaScript. Biblia"). Celem tego przykładu jest przypisanie łańcucha znaków właściwości `alt` obiektu `document.myIMG`. Rozpocznij od usunięcia obrazu wyświetlonego na stronie; w tym celu właściwości `src` przypisz nazwę nieistniejącego pliku:

```
document.myIMG.src = "brak_tego_pliku.gif"
```

Jeśli przewiniesz stronę, zobaczysz na niej puste miejsce po obrazie. Teraz określ wartość właściwości `src`:

```
document.myIMG.alt = "Buzia Freda"
```

Przewiń zawartość strony, aby przekonać się czy alternatywny tekst został wyświetlony. Jeśli w łańcuchu znaków zapisywanym we właściwości `alt` chcesz umieścić apostrof bądź cudzysłów (lub jeden z kilku innych znaków, które w języku JavaScript mają specjalne znaczenie), to musisz poprzedzić go znakiem odwrotnego ukośnika.

border

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Skutki określania wartości właściwości `border` obrazów możemy wypróbować na stronie *Tester* (posługując się zdefiniowanym na niej obrazem `document.myIMG`). Przeprowadzając testy, przypisuj tej właściwości liczby całkowite o różnej wartości.

complete

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Przed otwarciem strony z listingu 22.2 prezentującej zastosowanie właściwości `image.complete`, zamknij i ponownie otwórz przeglądarkę (w ten sposób będziesz miał pewność, że żadne obrazy nie znajdują się w jej pamięci podręcznej). Podczas pobierania każdego z obrazów sprawdzaj stan właściwości `complete` obiektu `image` — klikaj w tym celu przycisk *Czy obraz już pobrano*. Do momentu zakończenia pobierania obrazu właściwość ta będzie mieć wartość `false`, potem przyjmie wartość `true`. Na naszej przykładowej stronie używane są dwa obrazy, przy czym obraz prezentujący łuk skalny jest większy. Być może pomiędzy kolejnymi próbami będziesz musiał zamykać i ponownie otwierać przeglądarkę, aby usunąć obraz z pamięci podręcznej (lub opróżnić całą pamięć podręczną przeglądarki). Jeśli wykorzystanie właściwości `complete` we własnych skryptach będzie Ci przysparzać problemów, to spróbuj zdefiniować procedurę obsługi zdarzeń `onLoad` w znaczniku `<IMG>` (podobnie jak na listingu 22.2, może to być zupełnie pusta procedura obsługi zdarzeń).

Listing 22.2. *Skrypt wykorzystujący właściwość image.complete*

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript1.1">
function loadIt(theImage,form) {
    form.result.value = ""
    document.images[0].src = theImage
}
function checkLoad(form) {
    form.result.value = document.images[0].complete
}
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="cpu2.gif" WIDTH=120 HEIGHT=90 onLoad="">
<FORM>
<INPUT TYPE="button" VALUE="Wyświetl klawiaturę"
↳onClick="loadIt('cpu2.gif',this.form)">
<INPUT TYPE="button" VALUE="Wyświetl zdjęcie łuku skalnego"
↳onClick="loadIt('arch.gif',this.form)"><P>
<INPUT TYPE="button" VALUE="Czy obraz już pobrano" onClick="checkLoad(this.form)">
<INPUT TYPE="text" NAME="result">
```

```
</FORM>
</BODY>
</HTML>
```

dynsrc

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność							✓	✓	✓

Przykład

Aby w obiekcie `IMG` wyświetlać obrazy nieruchome bądź ruchome, wystarczy odpowiednio określać wartości właściwości `src` oraz `dynsrc` — zapisywać nazwę pliku obrazu w jednej z nich, a pusty łańcuch znaków w drugiej. Bardzo prosty przykład, wyświetlający na przemian jeden obraz stały oraz jeden klip wideo, został przedstawiony na listingu 22.3. Przedstawiony przykład demonstruje poprawny sposób zamieniania obrazów nieruchomych na ruchome (i na odwrót), umożliwiający uniknięcie błędów najczęściej popełnianych w takich sytuacjach (opisanych w tekście książki).

Listing 22.3. *Naprzemienne wyświetlanie obrazu nieruchomego i ruchomego*

```
<HTML>
<HEAD>
<TITLE>Właściwość dynsrc obiektu IMG</TITLE>
<SCRIPT LANGUAGE="JavaScript">

var trainImg = new Image(160,120)
trainImg.src = "amtrak.jpg"
trainImg.dynsrc = "amtrak.mpg"

function setLoop() {
    var selector = document.forms[0].looper
    document.myIMG.loop = selector.options[selector.selectedIndex].value
}

function setImage(type) {
    if (type == "jpg") {
        document.myIMG.dynsrc = ""
        document.myIMG.src = trainImg.src
    } else {
        document.myIMG.src = ""
        document.myIMG.start = "fileopen"
        setLoop()
        document.myIMG.dynsrc = trainImg.dynsrc
    }
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Właściwość dynsrc obiektu IMG</H1>
<HR>
<FORM>
Wybierz typ obrazu:
```

```
<INPUT TYPE="radio" NAME="imgGroup" CHECKED onClick="setImage('jpg')">Nieruchomy
<INPUT TYPE="radio" NAME="imgGroup" onClick="setImage('mpg')">Wideo
<P>Ile razy odtworzyć klip wideo po pobraniu: <SELECT NAME="looper"
↳onChange="setLoop()">
    <OPTION VALUE=1 SELECTED>raz
    <OPTION VALUE=2>dwa razy
    <OPTION VALUE=-1>w nieskończoność
</SELECT></P>
</FORM>
<HR>
<IMG NAME="myIMG" SRC="amtrak.jpg" HEIGHT=120 WIDTH=160>
</BODY>
</HTML>
```

Jeśli skrypt jawnie nie przypisze właściwości `start` wartości `fileopen` (jak w powyższym przykładzie), to aby odtworzyć klip, użytkownicy przeglądarek Internet Explorer dla komputerów Macintosh będą musieli kliknąć go dwa razy (IE 4) lub raz (IE 5).

fileCreatedDate
fileModifiedDate
fileSize

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność							✓	✓	✓

Przykład

Wszystkie trzy omawiane tu właściwości są podobne od analogicznych właściwości obiektu `document`. Przykład ich wykorzystania został przedstawiony na listingu 18.4. Jeśli chcesz wypróbować omawiane właściwości obiektu obrazu, to skopiuj stronę przedstawioną na listingu 18.4, dodaj do niej element `IMG`, a następnie zastąp odwołania do właściwości obiektu `document` odwołaniami do odpowiedniego obiektu elementu `IMG`.

height
width

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Właściwości `height` oraz `width` można wypróbować na stronie *Tester* (przedstawionej w rozdziale 13. książki „JavaScript. Biblia"). Zacznij od pobrania domyślnych wartości tych właściwości. W tym celu w górnym polu tekstowym formularza wpisz następujące wyrażenia:

```
document.myIMG.height
document.myIMG.width
```

Następnie zmień wysokość obrazu z 90 do 180 pikseli:

```
document.myIMG.height = 180
```

Gdy przewiniesz stronę i wyświetlisz obraz, okaże się, że został on przeskalowany (aby zachować oryginalne proporcje). W końcu podaj nową szerokość obrazu:

```
document.myIMG.width = 400
```

i przyjrzyj się efektom wprowadzonych modyfikacji.

hspace
vspace

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Do sprawdzenia właściwości `hspace` i `vspace` można użyć strony *Tester* (opisanej w rozdziale 13. książki „JavaScript. Biblia"). W pierwszej kolejności upewnij się, że obraz wyświetlony na samym końcu strony nie ma żadnych marginesów i jest wyświetlony tuż przy lewej krawędzi okna przeglądarki. Następnie zmień szerokość poziomego marginesu na 30 pikseli:

```
document.myIMG.hspace = 30
```

Wykonanie powyższej instrukcji spowoduje odsunięcie obrazu od lewej krawędzi strony na odległość 30 pikseli. Margines o tej samej szerokości został umieszczony także z prawej strony obrazu, choć nie jest on widoczny na stronie.

isMap

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność				✓			✓	✓	✓

Przykład

Obraz wyświetlony u dołu strony *Tester* nie został zdefiniowany jako mapa odnośników. Z tego względu, jeśli w górnym polu tekstowym formularza wyświetlonego na tej stronie wpiszesz poniższe wyrażenie i klikniesz przycisk *Oblicz*, to w polu poniżej zostanie wyświetlona wartość `false`:

```
document.myIMG.isMap
```

loop

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność							✓	✓	✓

Przykład

Przykład wykorzystania właściwości `loop` został przedstawiony na listingu 22.3, zamieszczonym w opisie właściwości `dynsrc`.

lowsrc
lowSrc

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Wpływ właściwości określających plik źródłowy obrazu na przetwarzanie zdarzeń został przedstawiony na listingu 22.5, podanym w opisie procedury obsługi zdarzeń `onLoad`.

name

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność	✓	✓	✓	✓	(✓)		✓	✓	✓

Przykład

Do sprawdzenia wartości właściwości `name` można wykorzystać stronę *Tester* (przedstawioną w rozdziale 13. książki „JavaScript. Biblia"). W górnym polu tekstowym formularza wpisz następujące wyrażenie:

```
document.myIMG.name
```

Oczywiście, powyższy przykład jest nieco bezsensowny, gdyż nazwa obrazu stanowi fragment użytego odwołania.

nameProp

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność								✓	✓

Przykład

Do porównania wartości właściwości `src` i `nameProp` w przeglądarce IE 5+ dla systemu Windows można użyć strony *Tester* (przedstawionej w rozdziale 13. książki „JavaScript. Biblia"). W górnym polu tekstowym formularza wpisz każde z poniższych wyrażeń:

```
document.myIMG.src
document.myIMG.nameProp
```

protocol

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność							✓	✓	✓

Przykład

Do sprawdzenia wartości właściwości `protocol` obiektu reprezentującego obraz można użyć strony *Tester*. W górnym polu tekstowym formularza wpisz poniższe wyrażenie:

```
document.myIMG.protocol
```

src

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓	(✓)		✓	✓	✓

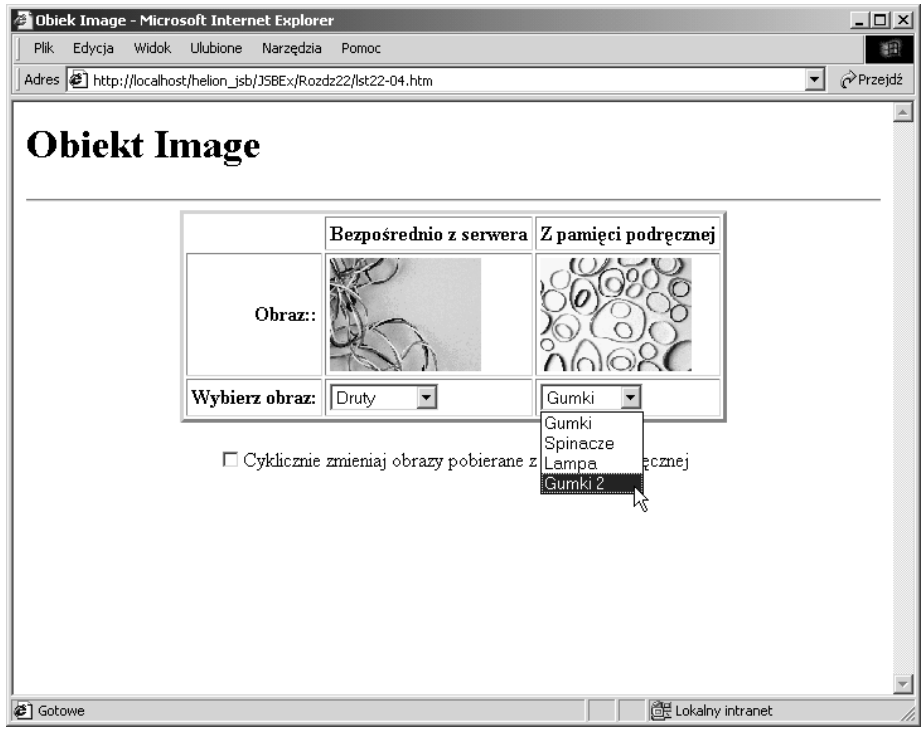
Przykład

W przykładzie przedstawionym na listingu 22.4 pokazanych zostało kilka różnych zastosowań obiektów reprezentujących obrazy. Podstawowym celem tego przykładu jest przedstawienie różnic w efektywności zamieniania obrazów przechowywanych w pamięci podręcznej przeglądarki oraz pobieranych bezpośrednio z serwera. Dodatkowo skrypt pokazuje w jaki sposób automatycznie zamieniać wyświetlane obrazy w podanych odstępach czasu. Rozwiązanie to jest bardzo często stosowane na witrynach wyświetlających paski z reklamami.

W momencie pobierania strony, w zmiennej globalnej zapisywana jest tablica obiektów `Image`. Poszczególne komórki tej tablicy są indeksowane łańcuchami znaków reprezentującymi nazwy obrazów (na przykład: `"desk1"`, `"desk2"` i tak dalej). Rozwiązanie to zostało wykorzystane celowo, gdyż skrypt będzie się odwoływał do poszczególnych komórek tablicy właśnie przy użyciu tych nazw. Każdemu obiektowi `Image` zapisanemu w tablicy zostaje przypisany adres URL obrazu. Dzięki temu obrazy zostają pobrane i zapisane w pamięci podręcznej przeglądarki.

Na stronie zostały zdefiniowane dwa elementy `IMG` (patrz rysunek 6.1). Pierwszy z nich wyświetla obrazy pobierane bezpośrednio z serwera, a drugi — obrazy pobierane z pamięci podręcznej. Poniżej każdego z elementów `IMG` zostaje wyświetlona rozwijana lista pozwalająca na wybranie jednego z czterech obrazów, który ma zostać wyświetlony w danym elemencie `IMG`. W procedurach obsługi zdarzeń `onChange` elementów `SELECT` wywoływane są dwie różne funkcje — w przypadku obrazów pobieranych z serwera wywoływana jest funkcja `loadIndividual()`, a w przypadku obrazów pobieranych z pamięci podręcznej — funkcja `loadCached()`. W wywołaniach obu funkcji podawany jest jeden argument — odwołanie do formularza zawierającego elementy `SELECT`.

Za cykliczną zmianę obrazów co 5 sekund odpowiada funkcja `checkTimer()`, która w pierwszej kolejności sprawdza czy jest zaznaczone odpowiednie pole wyboru. Jeśli pole zostało



Rysunek 6.1. *Strona demonstrująca wykorzystanie obiektów Image (Obrazy © Aris Multimedia Inc., 1994)*

zaznaczone, to funkcja pobiera wartość właściwości `selectedIndex` listy służącej do wyboru obrazu pobieranego z pamięci podręcznej. Następnie wartość ta jest inkrementowana (lub zmniejszana do zera, jeśli przekroczyła pewną maksymalną wartość), a odpowiednia lista zostaje zaktualizowana. Po wykonaniu powyższych czynności skrypt może już wywołać funkcję `loadCached()`, która określa wybrany element listy i wyświetla odpowiedni obraz.

Dodatkowo, w znaczniku `<BODY>` została zdefiniowana procedura obsługi zdarzenia `onUnload`, która wywołuje funkcję `resetSelects()`. Ta ogólna funkcja przegląda wszystkie formularze zdefiniowane na stronie i odnajduje w nich wszystkie elementy `SELECT`. W każdym z odnalezionych elementów, funkcja przypisuje właściwości `selectedIndex` wartość `0`. Dzięki temu, jeśli użytkownik odświeży stronę lub wróci na nią w wyniku kliknięcia przycisku *Wstecz* (*Back*), to obrazy będą odtwarzane w oryginalnej kolejności. Procedura obsługi zdarzenia `onLoad` zapewnia, że wyświetlane obrazy będą odpowiadać opcjom wybranym na odpowiadającym im listach, a po 5 sekundach zostanie wywołana funkcja `checkTimer()`. Należy jednak zwrócić uwagę, iż obrazy pobierane z pamięci podręcznej przeglądarki są zamieniane wyłącznie wówczas, gdy użytkownik zaznaczy pole wyboru wyświetlone u dołu strony.

Listing 22.4. *Obsługa obiektu image za pomocą skryptu i zamienianie obrazów*

```
<HTML>
<HEAD>
<TITLE>Obiek Image</TITLE>
```

```
<SCRIPT LANGUAGE="JavaScript">
// globalne deklaracje tablicy obrazów
var imageDB

// obrazy przechowywane w pamięci podręcznej
if (document.images) {
    // dla wygody zapisujemy tekstowe indeksy tablicy w formie listy
    var deskImages = new Array("desk1", "desk2", "desk3", "desk4")
    // tworzymy tablicę obiektów Image i pobieramy obrazy do
    // pamięci podręcznej
    imageDB = new Array(4)
    for (var i = 0; i < imageDB.length ; i++) {
        imageDB[deskImages[i]] = new Image(120,90)
        imageDB[deskImages[i]].src = deskImages[i] + ".gif"
    }
}

// zmieniamy obraz pobierany bezpośrednio z serwera
function loadIndividual(form) {
    if (document.images) {
        var gifName = form.individual.options[form.individual.selectedIndex].value
        document.thumbnail1.src = gifName + ".gif"
    }
}

// zmieniamy obraz przechowywany w pamięci podręcznej
function loadCached(form) {
    if (document.images) {
        var gifIndex = form.cached.options[form.cached.selectedIndex].value
        document.thumbnail2.src = imageDB[gifIndex].src
    }
}

// jeśli pole wyboru zostało zaznaczone, to pobieramy
// kolejny obraz przechowywany w pamięci podręcznej
function checkTimer() {
    if (document.images && document.Timer.timerBox.checked) {
        var gifIndex = document.selections.cached.selectedIndex
        if (++gifIndex > imageDB.length - 1) {
            gifIndex = 0
        }
        document.selections.cached.selectedIndex = gifIndex
        loadCached(document.selections)
        var timeoutID = setTimeout("checkTimer()",5000)
    }
}

// podczas usuwania strony z przeglądarki odtwarzamy oryginalny
// stan list rozwijanych
function resetSelects() {
    for (var i = 0; i < document.forms.length; i++) {
        for (var j = 0; j < document.forms[i].elements.length; j++) {
            if (document.forms[i].elements[j].type == "select-one") {
                document.forms[i].elements[j].selectedIndex = 0
            }
        }
    }
}

// inicjujemy działanie skryptu
```

```
function init() {
    loadIndividual(document.selections)
    loadCached(document.selections)
    setTimeout("checkTimer()",5000)
}
 </SCRIPT>
</HEAD>

<BODY onLoad="init()" onUnload="resetSelects ()">
<H1>Obiekt Image</H1>
<HR>
<CENTER>
<TABLE BORDER=3 CELLPADDING=3>
<TR><TH></TH><TH>Bezpośrednio z serwera</TH><TH>Z pamięci podręcznej</TH></TR>
<TR><TD ALIGN=RIGHT><B>Obraz::</B></TD>
<TD><IMG SRC="cpu1.gif" NAME="thumbnail1" HEIGHT=90 WIDTH=120></TD>
<TD><IMG SRC="desk1.gif" NAME="thumbnail2" HEIGHT=90 WIDTH=120></TD>
</TR>
<TR><TD ALIGN=RIGHT><B>Wybierz obraz:</B></TD>
<FORM NAME="selections">
<TD>
<SELECT NAME="individual" onChange="loadIndividual(this.form)">
<OPTION VALUE="cpu1">Druty
<OPTION VALUE="cpu2">Klawiatura
<OPTION VALUE="cpu3">Dyskietki
<OPTION VALUE="cpu4">Kable
</SELECT>
</TD>
<TD>
<SELECT NAME="cached" onChange="loadCached(this.form)">
<OPTION VALUE="desk1">Gumki
<OPTION VALUE="desk2">Spinacze
<OPTION VALUE="desk3">Lampa
<OPTION VALUE="desk4">Gumki 2
</SELECT></TD>
</FORM>
</TR></TABLE>
<FORM NAME="Timer">
<INPUT TYPE="checkbox" NAME="timerBox" onClick="checkTimer()">Cyklicznie zmieniaj
↳obrazy pobierane z pamięci podręcznej
</FORM>
</CENTER>
</BODY>
</HTML>
```

start

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność							✓	✓	✓

Przykład

Sposób wykorzystania właściwości `start` na stronie, która wyświetla klip wideo w elemencie `IMG`, został przedstawiony na listingu 22.3, zamieszczonym we wcześniejszej części tego rozdziału.

x
y

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność			✓						

Przykład

Aby przewinąć dokument w taki sposób, by obraz znalazł się o kilka pikseli poniżej górnej krawędzi okna przeglądarki, można użyć następującego wywołania:

```
window.scrollTo(document.images[0].x, (document.images[0].y - 3))
```

Procedury obsługi zdarzeń

onAbort
onError

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Na listingu 22.5 została zdefiniowana procedura obsługi zdarzeń `onAbort`. Jeśli chcesz zapobiec wyświetlaniu obrazu, który już znajduje się w pamięci podręcznej przeglądarki, będziesz musiał ją zamknąć i ponownie uruchomić. W przedstawionym przykładzie zapewniłem możliwość odświeżania całej strony. Sposób obsługi zdarzenia w głównej mierze zależy od projektu strony. Zawsze należy jednak dołożyć wszelkich starań, aby w jak największym stopniu ułatwić użytkownikowi rozwiązywanie problemów, które może napotkać korzystając ze strony.

onLoad

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność		✓	✓	✓			✓	✓	✓

Przykład

Przed wypróbowaniem przykładu przedstawionego na listingu 22.5 powinieneś zamknąć i ponownie uruchomić przeglądarkę. Gdy dokument jest otwierany po raz pierwszy, początkowo zostaje wyświetlony obraz określony za pomocą atrybutu `LOWSRC` (przedstawiający, w tym przypadku, gumki do ołówków), a dopiero później zdjęcie klawiatury. Gdy zdjęcie gumek zostanie już pobrane, procedura obsługi zdarzenia `onLoad` wyświetla w polu tekstowym słowo `"done"`. Dzieje się tak niezależnie od tego, że główny obraz (określony przy użyciu atrybut `SRC`) nie został jeszcze pobrany. Kolejny eksperyment

może polegać na pobraniu zdjęcia przedstawiającego łuk skalny. Pobranie tego obrazu zabiera znacznie więcej czasu, a zatem obraz tymczasowy określany dynamicznie za pomocą atrybutu LOWSRC zostanie wyświetlony znacznie wcześniej.

Listing 22.5. *Procedura obsługi zdarzeń onLoad obrazów*

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function loadIt(theImage,form) {
    if (document.images) {
        form.result.value = ""
        document.images[0].lowsrc = "desk1.gif"
        document.images[0].src = theImage
    }
}
function checkLoad(form) {
    if (document.images) {
        form.result.value = document.images[0].complete
    }
}
function signal() {
    if(confirm("Przerwałeś pobieranie obrazu. Czy chcesz pobrać go jeszcze raz?")) {
        location.reload()
    }
}
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="cpu2.gif" LOWSRC="desk4.gif" WIDTH=120 HEIGHT=90 onLoad="if
↳(document.forms[0].result) document.forms[0].result.value='done'" onAbort="signal()">
<FORM>
<INPUT TYPE="button" VALUE="Wyświetl klawiaturę"
↳onClick="loadIt('cpu2.gif',this.form)">
<INPUT TYPE="button" VALUE="Wyświetl łuk skalny"
↳onClick="loadIt('arch.gif',this.form)"><P>
<INPUT TYPE="button" VALUE="Czy obraz już pobrano?" onClick="checkLoad(this.form)">
<INPUT TYPE="text" NAME="result">
</FORM>
</BODY>
</HTML>
```

Obiekt elementu AREA

Właściwości

coords
shape

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność				✓			✓	✓	✓

Przykład

Sposób wykorzystania właściwości `coords` oraz `shape` w skryptach obsługujących obiekty elementów `MAP` został przedstawiony na listingu 22.7.

Obiekt elementu MAP

Właściwość

areas

	NN 2	NN 3	NN 4	NN 6	IE 3/J1	IE 3/J2	IE 4	IE 5	IE 5.5
Zgodność				✓			✓	✓	✓

Przykład

Listing 22.7 demonstruje jak można zmieniać obiekty elementów `AREA` umieszczone wewnątrz elementu `MAP`. Początkowo na stronie jest wyświetlany obraz przedstawiający klawiaturę. Obraz ten jest skojarzony z mapą odnośników `keyboardMap` obsługiwaną po stronie klienta i definiującą trzy wybrane obszary na klawiaturze. Jeśli użytkownik zamieni obraz wyświetlany na stronie wewnątrz elementu `IMG`, skrypt zmieni także wartość właściwości `useMap` obiektu elementu `IMG`, tak aby wskazywała na drugi element `MAP`, znacznie lepiej dostosowany do nowego obrazu. Spróbuj przesuwać wskaźnik myszy w obszarze obu obrazów i sprawdź adresy URL skojarzone z poszczególnymi obszarami mapy, wyświetlane na pasku stanu (w tym przykładzie, strony docelowe określane przez te adresy nie istnieją).

Kliknięcie przycisku *Wygeneruj mapę* powoduje jednak wywołanie funkcji `makeAreas()` tworzącej cztery nowe obiekty elementów `AREA` i dodającej je do mapy skojarzonej z obrazem (przy użyciu odwołań charakterystycznych dla DOM używanej przeglądarki). (Należy zauważyć, że funkcja `makeAreas()` nie działa w przeglądarkach IE 5 dla komputerów Macintosh). Przesuwając wskaźnik myszy w obszarze obrazu po wygenerowaniu nowych obiektów elementów `AREA`, możemy zauważyć, że wyświetlane są nowe adresy URL. Warto także zwrócić uwagę na pojawienie się czterech nowych obszarów; komunikaty o nich są wyświetlane na pasku stanu przeglądarki (patrz rysunek 6.2).

Listing 22.7. *Dynamiczne modyfikowanie elementów AREA*

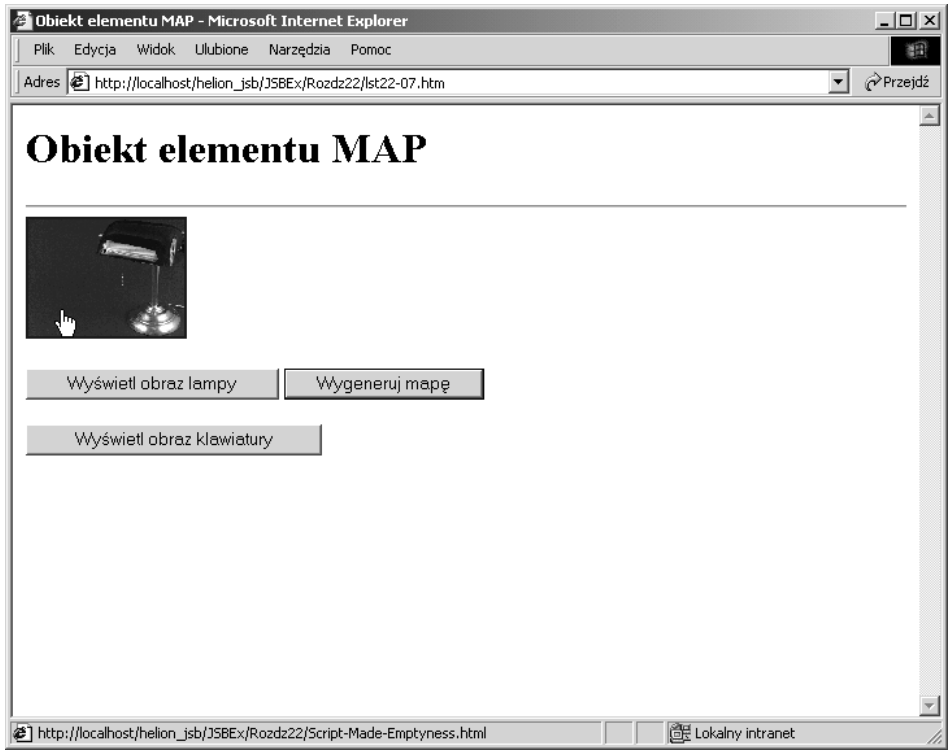
```
<HTML>
<HEAD>
<TITLE>Obiekt elementu MAP</TITLE>
<SCRIPT LANGUAGE="JavaScript">
// dynamicznie generujemy elementy AREA
function makeAreas() {
    document.myIMG.src = "desk3.gif"
    // tworzymy obiekty elementów area
    var area1 = document.createElement("AREA")
```

```
    area1.href = "Script-Made-Shade.html"
    area1.shape = "polygon"
    area1.coords = "52,28,108,35,119,29,119,8,63,0,52,28"
    var area2 = document.createElement("AREA")
    area2.href = "Script-Made-Base.html"
    area2.shape = "rect"
    area2.coords = "75,65,117,87"
    var area3 = document.createElement("AREA")
    area3.href = "Script-Made-Chain.html"
    area3.shape = "polygon"
    area3.coords = "68,51,73,51,69,32,68,51"
    var area4 = document.createElement("AREA")
    area4.href = "Script-Made-Emptyness.html"
    area4.shape = "rect"
    area4.coords = "0,0,50,120"
    // wstawiamy nowe elementy do węzłów podrzędnych elementu MAP
    if (document.all) {
        // kod dla IE4+
        document.all.lamp_map.areas.length = 0
        document.all.lamp_map.areas[0] = area1
        document.all.lamp_map.areas[1] = area2
        document.all.lamp_map.areas[2] = area3
        document.all.lamp_map.areas[3] = area4
    } else if (document.getElementById) {
        // NN6 jest zgodny z modelem węzłów W3C
        var mapObj = document.getElementById("lamp_map")
        while (mapObj.childNodes.length) {
            mapObj.removeChild(mapObj.firstChild)
        }
        mapObj.appendChild(area1)
        mapObj.appendChild(area2)
        mapObj.appendChild(area3)
        mapObj.appendChild(area4)
        // rozwiązanie problemu z wyświetlaniem w NN6
        document.myIMG.style.display = "inline"
    }
}

function changeToKeyboard() {
    document.myIMG.src = "cpu2.gif"
    document.myIMG.useMap = "#keyboardMap"
}

function changeToLamp() {
    document.myIMG.src = "desk3.gif"
    document.myIMG.useMap = "#lampMap"
}
</SCRIPT>
</HEAD>
<BODY>
<H1>Obiekt elementu MAP</H1>
<HR>
<IMG NAME="myIMG" SRC="cpu2.gif" WIDTH=120 HEIGHT=90 USEMAP="#keyboardMap">
<MAP NAME="keyboardMap">
<AREA HREF="AlpaKeys.htm" SHAPE="rect" COORDS="0,0,26,42">
<AREA HREF="ArrowKeys.htm" SHAPE="polygon"
➥COORDS="48,89,57,77,69,82,77,70,89,78,84,89,48,89">
<AREA HREF="PageKeys.htm" SHAPE="circle" COORDS="104,51,14">
```

```
</MAP>
<MAP NAME="lampMap" ID="lamp_map">
<AREA HREF="Shade.htm" SHAPE="polygon" COORDS="52,28,108,35,119,29,119,8,63,0,52,28">
<AREA HREF="Base.htm" SHAPE="rect" COORDS="75,65,117,87">
<AREA HREF="Chain.htm" SHAPE="polygon" COORDS="68,51,73,51,69,32,68,51">
</MAP>
<FORM>
<P><INPUT TYPE="button" VALUE="Wyświetl obraz lampy" onClick="changeToLamp()">
<INPUT TYPE="button" VALUE="Wygeneruj mapę" onClick="makeAreas()"></P>
<P>
<INPUT TYPE="button" VALUE="Wyświetl obraz klawiatury"
↳onClick="changeToKeyboard()"></P>
</FORM>
</BODY>
</HTML>
```



Rysunek 6.2. *Skrypt wygenerował specjalną mapę odnośników dostosowaną do wyświetlanego obrazu*