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

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.

Spis treści

Praca z obrazami 4

W tym rozdziale:

Jednym z najciekawszych zastosowań JavaScriptu jest ożywianie 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!

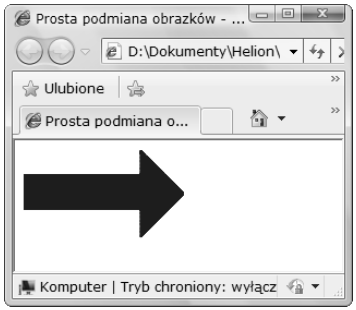
Tabela 4.1. *Podstawy HTML — obrazy*

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.

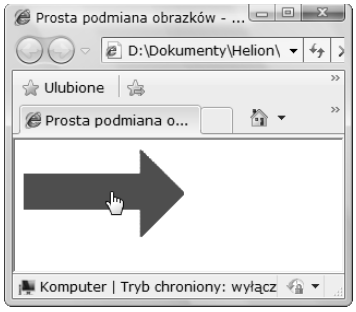
Skrypt 4.1. *Najprostszy sposób utworzenia animowanego przycisku menu w znaczniku łącza*

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'"><img
src="images/arrow_off.gif" id="arrow"
alt="arrow"></a>
</body>
</html>
```



Rysunek 4.1. *Pierwszy obrazek przed wskazaniem go myszą*



Rysunek 4.2. *Po wskazaniu obrazka myszą 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:

1. `<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`.

   ```
   <img src="images/arrow_off.gif" id="arrow"
   ➥alt="strzałka">
   ```

   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.

Skrypt 4.3. *Jedynym kodem JavaScript na tej stronie HTML jest wskazanie zewnętrznego pliku .js*

Skrypt

```
<!DOCTYPE html>
<html>
<head>
   <title>Lepsza podmiana obrazków</title>
   <script src="script02.js"></script>
   <link rel="stylesheet"
href="script01.css">
</head>
<body>
   <a href="next1.html"><img src=
   ➥"images/button1_off.gif" alt="button1"
   ➥id="button1"></a>  
   <a href="next2.html"><img src=
   ➥"images/button2_off.gif" alt="button2"
   ➥id="button2"></a>
</body>
</html>
```

Skrypt 4.4. *Oto lepszy sposób tworzenia podmienianych obrazków. Jest dużo elastyczniejszy od sposobu ze skryptu 4.2*

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

Lepsza technika podmiany obrazków

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.

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.).

Oto skuteczniejsza podmiana obrazków:

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

   Ten znacznik pochodzi ze skryptu 4.3, czyli ze strony HTML. Atrybut `src to` informacja dla przeglądarki, gdzie może znaleźć plik *.js*, w którym zapisany jest kod JavaScript.

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

   <a href="next2.html"><img src=
   ➥"images/button2_off.gif" alt="button2"
   ➥id="button2"></a>
   ```

   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 `<img>` 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.

3. `window.onload = rolloverInit;`

   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 niezaładowanej 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 łączami. 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.

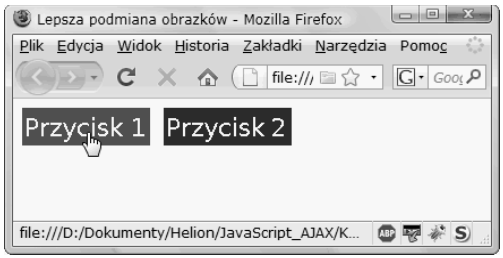
W tym miejscu informujemy przeglądarkę, co powinna zrobić w momencie, gdy użytkownik przesunie wskaźnik myszy poza obrazek. W takiej sytuacji chcemy, żeby przywrócona została początkowa wersja obrazka, zapisana w zmiennej `outImage`.

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

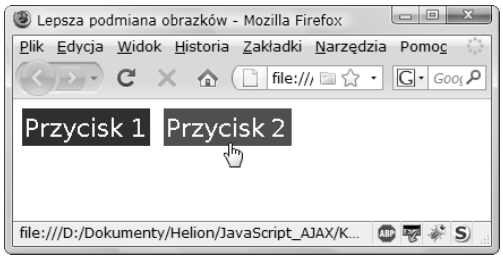
W pierwszym wierszu tworzymy nowy obiekt obrazu, który będzie zawierać wersję obrazka wyświetlaną po wskazaniu go myszą. Drugi wiersz definiuje źródło obrazka dla obiektu `overImage`. Nazwa pliku budowana jest na bieżąco przez złożenie nazwy katalogu `images/`, identyfikatora obrazka podstawowego (pamiętamy, 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).



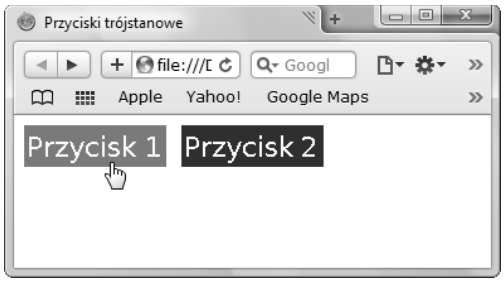
Rysunek 4.3. *Na jednej stronie można mieć wiele podmienianych obrazków*



Rysunek 4.4. *Wskazujemy drugi obrazek*

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 4.5. *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 jakiejkolwiek modyfikacji kodu HTML*

Skrypt

```
<!DOCTYPE html>
<html>
<head>
   <title>Przyciski trójstanowe</title>
   <script src="script03.js"></script>
   <link rel="stylesheet"
href="script01.css">
</head>
<body>
   <a href="next1.html"><img
src="images/button1_off.gif" alt="button1"
id="button1"></a>  
   <a href="next2.html"><img
src="images/button2_off.gif" alt="button2"
id="button2"></a>
</body>
</html>
```

Tworzenie przycisków trójstanowych

Przycisk trójstanowy 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

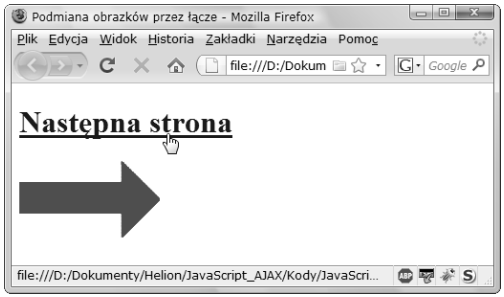
```
window.onload = rolloverInit;

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 podmienianie 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>
   <img src="images/arrow_off.gif"
   ↪id="arrowImg" alt="arrow">
</body>
</html>
```

Podmiana obrazków poprzez łącze

We wcześniejszych przykładach użytkownik powodował zamianę 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;
}
```

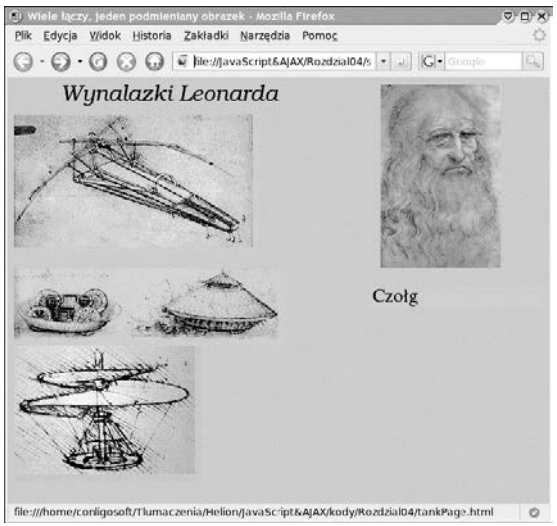
   W momencie wywołania zdarzenia `mouseover` lub `mouseout` obserwujemy działanie nieco inne od prezentowanego w poprzednich przykładach. Tym razem modyfikowana jest właściwość `this.imgToChange.src`, a nie właściwość `this.src`, tak jak to było robione poprzednio.

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 łączami 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 któregokolwiek 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">
      <img src="images/DaVinci.jpg"
width="144" height="219" alt="DaVinci">
      <img src="images/bg.gif"
id="captionField" alt="Pole tekstowe">
   </div>
   <div id="inventionDiv">
      <img src="images/leoText.gif"
id="heading" alt="Wynalazki Leonarda">
      <a href="flyPage.html"
class="captionField" id="flyer"><img
src="images/flyer.gif" width="293"
height="165" alt="Maszyna latająca"
id="flyerImg"></a>
      <a href="tankPage.html"
class="captionField" id="tank"><img
src="images/tank.gif" width="325"
height="92" alt="Czołg" id="tankImg"></a>
      <a href="heliPage.html"
class="captionField" id="helicopter"><img
src="images/helicopter.gif" width="224"
height="160" alt="Helikopter"
id="helicopterImg"></a>
   </div>
</body>
</html>
```

Skrypt 4.10. *Ten skrypt pozwala podmieniać jeden obrazek poprzez wiele łączy*

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 łączy*

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">
      <img src="images/DaVinci.jpg" width="144" height="219" alt="DaVinci">
      <img src="images/bg.gif" id="captionField" alt="Pole tekstowe">
   </div>
   <div id="inventionDiv">
      <img src="images/leoText.gif" id="heading" alt="Wynalazki Leonarda">
      <a href="flyPage.html" class="captionField" id="flyer"><img src="images/flyer.gif"
width="293" height="165" alt="Maszyna latająca" id="flyerImg"></a>
      <a href="tankPage.html" class="captionField" id="tank"><img src="images/tank.gif"
width="325" height="92" alt="Czołg" id="tankImg"></a>
      <a href="heliPage.html" class="captionField" id="helicopter"><img
src="images/helicopter.gif" width="224" height="160" alt="Helikopter" id="helicopterImg"></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 = rollOut;
  thisLink.onmouseover = rollOver;

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

function rollOut() {
  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">
      <img src="images/reading1.gif"
      ↪id="adBanner" alt="Reklama">
   </div>
</body>
</html>
```

Skrypt 4.15. *Kod JavaScript służy do cyklicznego podmieniania banerów*

Skrypt

```
window.onload = rotate;

var thisAd = 0;

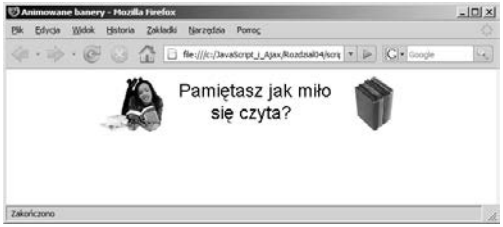
function rotate() {
   var adImages = new Array("images/
   ↪reading1.
   ↪gif","images/reading2.gif","images/
   ↪reading3.gif");

   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 podmiany 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 wyznaczanej 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 miałoby 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 obiadki" 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 łącza 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 łącza. 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 łączami</title>
   <script src="script08.js"></script>
   <link rel="stylesheet" href="script01.css">
</head>
<body>

      <a href="linkPage.html"><img
src="images/banner1.gif" id="adBanner"
alt="reklama"></a>

</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`.

Prezentacje

Prezentacje, czyli tak zwane *slide shows* (dosł. pokazy slajdów), wyświetlają obrazki i umożliwiają użytkownikowi kierowanie ich przewijaniem do przodu lub do tyłu. Kod JavaScriptu zapewnia niezbędną interaktywną kontrolę nad tymi działaniami. W skrypcie 4.18 przedstawiony został kod HTML, a w skrypcie 4.19 kod JavaScript strony tworzącej taki właśnie pokaz slajdów.

W skrypcie tym powstaje zawijający się pokaz slajdów, co oznacza, że po dotarciu do ostatniego zdjęcia wyświetlane jest pierwsze i vice versa. Działanie skryptu można podziwiać na rysunku 4.14.

Skrypt 4.18. *Kod HTML strony z pokazem slajdów*

Skrypt

```
<!DOCTYPE html>
<html>
<head>
   <title>Pokaz slajdów</title>
   <script src="script09.js"></script>
   <link rel="stylesheet" href="script01.css">
</head>
<body>
   <div class="centered">
      <h1>Witajcie, wspaniałe roboty!</h1>
      <img src="images/robot1.jpg"
      ↪id="myPicture" width="200"
      ↪height="400" alt="Slideshow">
      <h2><a href="previous.html"
      ↪id="prevLink">&lt;&lt;
      ↪Poprzedni</a>  
      ↪<a href="next.html" id="nextLink">
      ↪Następny &gt;&gt;</a></h2>
   </div>
</body>
</html>
```

Skrypt 4.19. *Skrypt tworzący pokaz slajdów, które użytkownik może przeglądać, klikając odpowiednie łącza*

Skrypt

```
window.onload = initLinks;

var myPix = new Array("images/robot1.jpg", "images/robot2.jpg", "images/robot3.jpg");
var thisPic = 0;

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

function processPrevious() {
  if (thisPic == 0) {
    thisPic = myPix.length;
  }
  thisPic--;
  document.getElementById("myPicture").src = myPix[thisPic];
  return false;
}

function processNext() {
  thisPic++;
  if (thisPic == myPix.length) {
    thisPic = 0;
  }
  document.getElementById("myPicture").src = myPix[thisPic];
  return false;
}
```

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 łącza 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>
   <img src="images/spacer.gif" width="305"
   ➥height="312" id="myPicture" alt="obrazek">
</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 wybieranego 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 rety!).

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 malutki 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">
      <img src="images/spacer.gif"
      ➥id="adBanner" alt="Reklama">
   </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 podmiany 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.

Skorowidz

S

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

Q

R

S

Ś

T

U

W

X

Z

Ź

Ż

PO PROSTU JavaScript

Wydanie VIII

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 JavaScripcie
- możliwości i ograniczenia tego języka
- najlepsze sposoby korzystania z technologii AJAX

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-4271-7

9 788324 642717

Cena 69,00 zł