Helion

SMASHING MAGAZINE

Jake Rutter

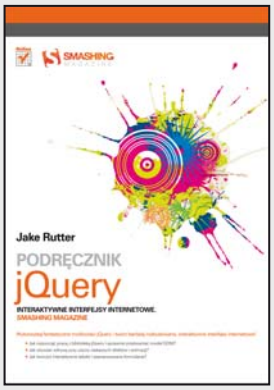
PODRĘCZNIK jQuery

INTERAKTYWNE INTERFEJSY INTERNETOWE.
SMASHING MAGAZINE

Wykorzystaj fantastyczne możliwości jQuery i twórz bardziej rozbudowane, interaktywne interfejsy internetowe!

- Jak rozpocząć pracę z biblioteką jQuery i sprawnie przetwarzać model DOM?
- Jak ożywiać witrynę przy użyciu ciekawych efektów i animacji?
- Jak tworzyć interaktywne tabele i zaawansowane formularze?

Podrecznik jQuery. Interaktywne interfejsy internetowe. Smashing Magazine



Autor: Jake Rutter
Tłumaczenie: Piotr Pilch
ISBN: 978-83-246-3316-6
Tytuł oryginału: Smashing jQuery (Smashing Magazine Book Series)
Format: 168×237, stron: 360

Wykorzystaj fantastyczne możliwości jQuery i twórz bardziej rozbudowane, interaktywne interfejsy internetowe!

- Jak rozpocząć pracę z biblioteką jQuery i sprawnie przetwarzać model DOM?
- Jak ożywiać witrynę przy użyciu ciekawych efektów i animacji?
- Jak tworzyć interaktywne tabele i zaawansowane formularze?

Stworzona w 2006 roku biblioteka jQuery miała być wybawieniem dla wielu programistów, którzy wcześniej nie mieli alternatywy – byli zmuszeni do korzystania ze skomplikowanych bibliotek języka JavaScript. I choć nie oferowała żadnych nowych funkcji, dzięki swej przejrzystej i prostej składni miała sprawić, by trudne do zrozumienia i utworzenia interfejsy API JavaScriptu stały się wreszcie szeroko dostępne. Twórcy stron nie rozczarowali się! Biblioteka jQuery spełniła pokładane w niej oczekiwania ? korzystanie z niej znacząco skróciło czas pisania kodu oraz umożliwiło projektantom i programistom szybkie tworzenie komponentów interaktywnych zgodnych ze wszystkimi najważniejszymi przeglądarkami.

Jak zatem łatwo tworzyć bogate w możliwości interfejsy internetowe, integrując strukturę biblioteki jQuery z witryną internetową przy minimalnej znajomości języka JavaScript? Oto znakomita książka, napisana z myślą o wszystkich projektantach i programistach stron internetowych, którzy chcą szybko rozpocząć pracę z biblioteką jQuery. Pierwsza część książki dokładnie omawia bibliotekę jQuery, korzyści płynące z jej użycia oraz strategię progresywnego rozszerzania.

Wnikliwie przedstawia też sposób instalowania i przygotowywania biblioteki jQuery do natychmiastowego użycia. W drugiej części podręcznika krok po kroku omówiono korzystanie z selektorów oraz pracę ze zdarzeniami i efektami ? wszystko po to, aby zapewnić Ci solidne podwaliny pod tworzenie własnej witryny i komponentów interfejsu użytkownika. Kolejne części publikacji koncentrują się na wykorzystaniu biblioteki jQuery do usprawnienia sprawdzania poprawności formularzy, tworzeniu dodatków oraz pracy z aplikacjami mobilnymi jQuery.

W książce omówiono m.in. następujące zagadnienia:

- Podstawy biblioteki jQuery
- Żądania Ajax oraz zdarzenia i efekty
- Przetwarzanie modelu DOM z kursami poświęconymi takim czynnościom jak tworzeniemenu rozwijanego
- Ramki nakładane galerii
- Zarządzanie formularzami
- Dane tabel dynamicznych
- Efekty zdarzeń myszy
- Modalne okna dialogowe
- Niestandardowe dodatki biblioteki jQuery

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

Spis treści

5 OŻYWIANIE WITRYNY INTERNETOWEJ PRZY UŻYCIU EFEKTÓW

W OSTATNICH LATACH efekty JavaScript dojrzały do branży, w której światem efektów na stronach internetowych rządziła technologia Adobe Flash. Witryny internetowe z pokazami slajdów, animowanymi menu lub animacjami przypominającymi wideo, które były tworzone wyłącznie w technologii Flash, obecnie często są wykonywane w języku JavaScript w celu zwiększenia ich zgodności z różnymi przeglądarkami i urządzeniami przenośnymi. Ten wzrost popularności efektów JavaScript stał się głównym powodem wykorzystania przez projektantów i programistów interfejsu API efektów w bibliotece jQuery.

W celu zapewnienia niezawodnych rozwiązań biblioteka jQuery korzysta z macierzystych efektów JavaScript, które mogą być Ci już znane. Rozwiązania te można łatwo zintegrować z dowolną witryną internetową. Ponieważ efekty są pisane przy użyciu biblioteki jQuery, ich konfigurowanie jest wyjątkowo proste. Dzięki temu cieszą się one popularnością wśród projektantów i programistów witryn internetowych.

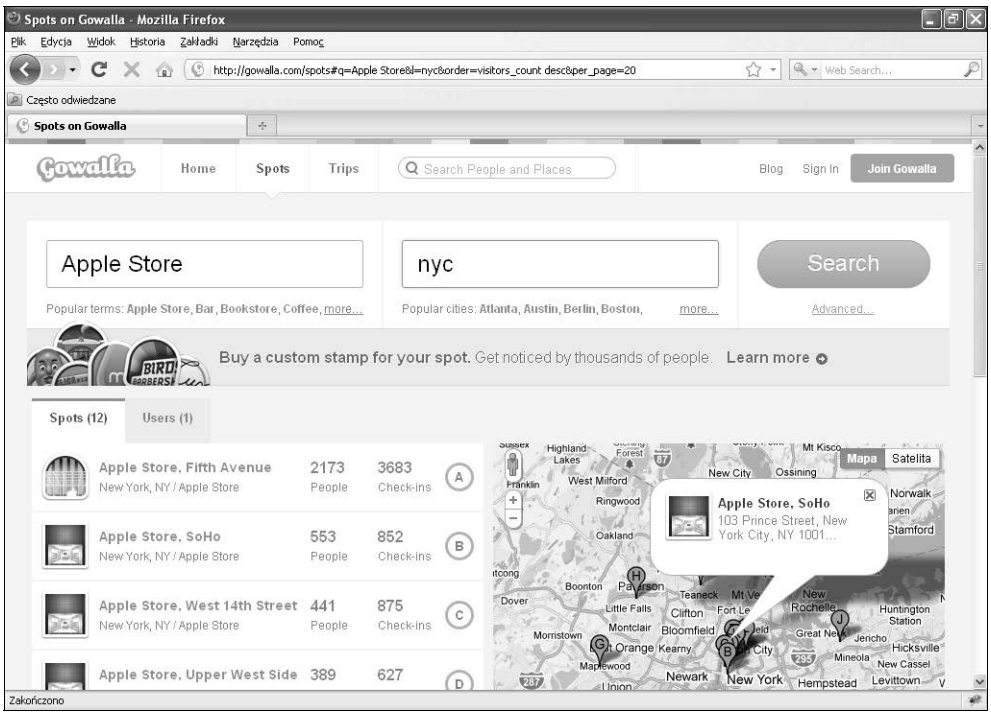
W rozdziale tym dokonam przeglądu efektów dostępnych za pośrednictwem interfejsu API biblioteki jQuery, omówię poszczególne efekty i ich działanie, a następnie zaprezentuję kilka rzeczywistych scenariuszy.

POZNAWANIE MOŻLIWOŚCI EFEKTÓW JQUERY

Zadaniem projektantów i programistów interfejsów użytkownika witryn internetowych jest zapewnienie ich użyteczności. Często uwzględnia to przenoszenie elementów na ekran lub poza jego obręb w celu zmieszczenia na jednej stronie większej ilości treści. Użytkownicy domagają się i oczekują natychmiastowej satysfakcji. Nie będą czekać na to, aż zakończy się ładowanie niezgrabnych witryn internetowych z mnóstwem stron.

Facebook, najpopularniejszy portal społecznościowy z liczbą użytkowników przekraczającą ponad 500 milionów, oferuje bardzo interaktywny i zabawny interfejs oparty na języku JavaScript. Jeśli zalogujesz się na stronie portalu Facebook, będziesz mógł porozmawiać z przyjaciółmi i sprawdzić kanał informacyjny znajomego bez konieczności żądania załadowania nowej strony. Tego rodzaju obsługę użytkowników uzyskano przy użyciu efektów JavaScript takich jak wyświetlanie oraz ukrywanie i animacje. Witryny internetowe takie jak Facebook ustawiają wysoko poprzeczkę w kwestii tego, czego użytkownicy mogą oczekiwać, gdy chodzi o komfort obsługi stron internetowych.

Coraz większą popularność zdobywają aplikacje oparte na geolokacji. Wiele spośród tych witryn internetowych wykorzystuje technologie interfejsowe typu Google Maps bazujące na języku JavaScript. Na rysunku 5.1 pokazano witrynę Gowalla, czyli portal społecznościowy wykorzystujący geolokację.



Rysunek 5.1. Witryna internetowa Gowalla będąca portalem społecznościowym bazującym na geolokacji (© 2010 Gowalla Incorporated)

Biblioteka jQuery zapewnia proste efekty takie jak pokazywanie, ukrywanie, zsuwanie i znikanie. W tabeli 5.1 wyszczególniono podstawowe efekty tworzone w podobny sposób, a także mające te same parametry opcjonalne, które mogą być przekazywane metodom.

Tabela 5.1. Podstawowe efekty jQuery

Nazwa efektu	Funkcja
`show()`	Pokazuje element.
`hide()`	Ukrywa element.
`toggle()`	Powoduje przełączenie między efektami pokazywania i ukrywania przy użyciu zdarzenia `click`.
`slideDown()`	Powoduje zsuwanie elementu w dół.
`slideUp()`	Powoduje przesuwanie elementu w górę.
`slideToggle()`	Powoduje przełączenie między efektami przesuwania elementu w górę i w dół.
`fadeIn()`	Powoduje zmniejszanie przezroczystości elementu.
`fadeOut()`	Powoduje zwiększanie przezroczystości elementu.
`fadeTo()`	Powoduje uzyskanie określonej nieprzezroczystości elementu.

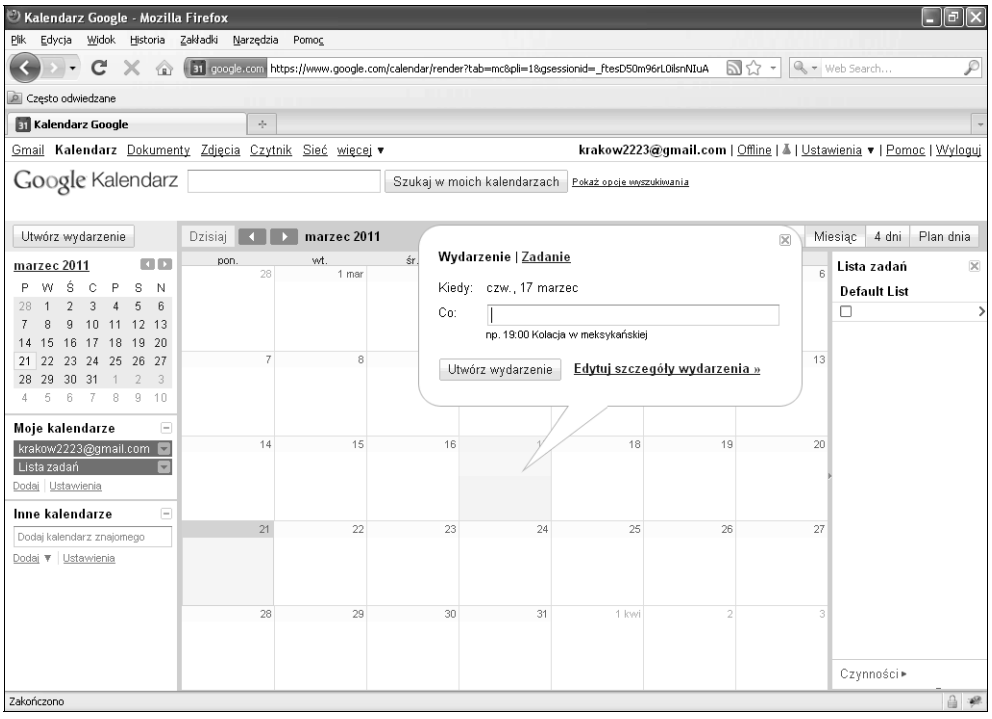
UŻYCIE EFEKTÓW POKAZYWANIA I UKRYWANIA

W przypadku biblioteki jQuery pokazywanie i ukrywanie elementów to proste efekty. Choć w poprzednich rozdziałach przedstawiłem ich przykłady, zwykle są one stosowane w połączeniu ze zdarzeniem `click`. Efekty te są powszechnie używane w internecie. Na rysunku 5.2 przedstawiono aplikację Google Kalendarz, która korzysta z efektów `show` i `hide` w celu wyświetlenia okienka wydarzenia.

Efekt `show` lub `hide` jest dołączany do selektora. W efektach tych mogą być przekazywane dwa parametry opcjonalne. Parametr `duration` określa czas odtwarzania animacji, który możesz ustawić przy użyciu słów kluczowych `fast` lub `slow`, jak również wyrazić w milisekundach (600, 200, 700 itd.). Parametr `callback` umożliwia połączenie z funkcją, która jest wykonywana po zakończeniu działania efektu `show`.

```
$(selector).show(duration, callback)
```

W poniższym przykładzie zaprezentowano odnośnik z powiązanym zdarzeniem `click`. Po kliknięciu odnośnika zostanie pokazany element z klasą `recipe`. Jest to efekt `show` w swojej najprostszej postaci.



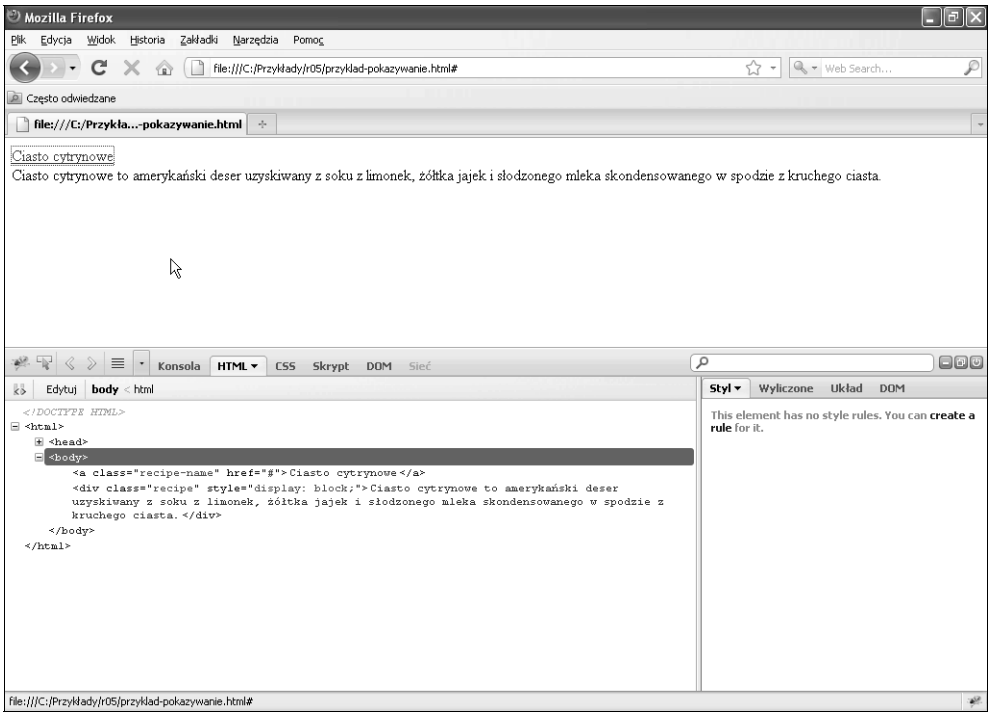
Rysunek 5.2. Aplikacja Google Kalendarz używa efektów show i hide do wyświetlenia okienka wydarzenia (reprodukcja z 2010 © Google)

```
<style>
.recipe {display:none;}
</style>
$('.recipe-name').bind('click', function() {
  $('.recipe').show();
});
<a href="#" class="recipe-name">Ciasto cytrynowe</a>
<div class="recipe">Ciasto cytrynowe to amerykański deser uzyskiwany z soku z limonek,
↪żółtka jajek i słodzonego mleka skondensowanego w spodzie z kruchego ciasta.</div>
```

Biblioteka jQuery umożliwia pokazanie treści elementu `div` przez dodanie stylu wstawianego `display:block` do wybranego elementu. Na rysunku 5.3 przedstawiono wynik wykonania w przeglądarce kodu z poprzedniego przykładu.

Zdarzenie `hide` działa dokładnie tak jak zdarzenie `show` z tą różnicą, że ukrywa wybrany element. Styl wstawiany zmieniono na styl `display:none`.

```
$('.recipe-name').bind('click', function() {
  $('.recipe').hide();
});
<a href="#" class="recipe-name">Ciasto cytrynowe</a>
<div class="recipe">Ciasto cytrynowe to amerykański deser uzyskiwany z soku z
limonek, żółtka jajek i słodzonego mleka skondensowanego w spodzie z kruchego
ciasta.</div>
```



Rysunek 5.3. Wynik wykonania w przeglądarce kodu z poprzedniego przykładu

Aby w większym stopniu kontrolować szybkość efektu `show`, wystarczy przekazać metodzie `show` określone słowo kluczowe (`fast`/`slow`) lub liczbę milisekund.

```
$('.recipe').show('slow');
```

Metodzie `show` możesz również przekazać funkcję `callback`, która jest wywoływana po zakończeniu przetwarzania efektu.

KONFIGUROWANIE KOMUNIKATU WYŚWIETLANEGO JEDNOKROTNIE W WITRYNIE PRZY UŻYCIU METODY SHOW I INFORMACJI COOKIE

Załóżmy sytuację, w której możesz wyświetlić użytkownikom specjalną ofertę lub komunikat, lecz chcesz je pokazać tylko raz. Często widywałem komunikaty z interfejsem WWW podobne do wyświetlanego w witrynie narzędzia Basecamp, które służy do zarządzania projektami. W tym przypadku jest wyświetlany komunikat logowania informujący użytkownika o nowej funkcji. Aby zapobiec ponownemu wyświetlaniu tego komunikatu dla tego samego komputera i konta użytkownika, możesz użyć metody `show` w połączeniu z funkcją zwrotną, która pozostawia informację *cookie* na komputerze użytkownika.

Na początek utwórz komunikat i przygotuj informację *cookie*.

1. Utwórz kod HTML dla komunikatu, który ma zostać pokazany, i dołącz odnośnik umożliwiający użytkownikowi ukrycie tej informacji. Umieść na stronie odnośnik, który pozwoli wyświetlić ukryty komunikat.

```
<a href="#" class="special-offer">Wyświetl tę ofertę specjalną!</a>
<div id="message">
  Oferta specjalna dla członków! 50% zniżki przy pierwszym zakupie.<br/>
  <a href="#" class="hide">Ukryj ten komunikat</a>
</div>
```

2. Utwórz zdarzenie `click` dla odnośnika oferty specjalnej umożliwiającego wyświetlenie komunikatu oraz funkcję `callback` `hideMessage`, która wkrótce zostanie zdefiniowana. Wewnątrz zdarzenia `click` dodaj instrukcję selektora dla elementu komunikatu z dołączonym do niego efektem `show`. Nie ma potrzeby przekazywania metodzie `show` parametru `duration`. Niezbędne jest jednak dodanie funkcji `callback`, która ma zostać wykonana po zakończeniu działania metody `show`. W tym przypadku jest to funkcja `hideMessage`.

```
$('.special-offer').bind('click', function(){
  $('#message').show(hideMessage);
});
```


3. Utwórz kolejne zdarzenie `click` dla odnośnika ukrywającego komunikat i ustaw funkcję `callback` `hideMessage`.

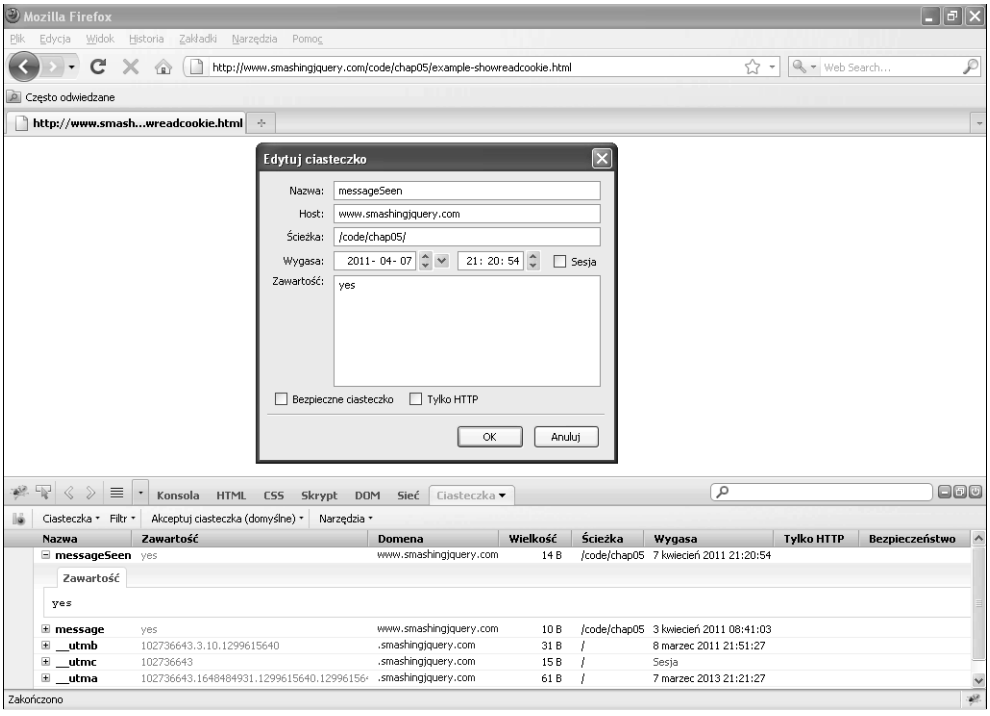
```
$('.hide').bind('click', function(){
    $('#message').hide(hideMessage);
  });
```

4. Następnie ustaw funkcję `hideMessage()`, która po wyświetleniu przez użytkownika komunikatu pozostawia na jego komputerze informację *cookie*.

```
function hideMessage() {
}
```

5. Utwórz informację *cookie* o nazwie `hideCookie` i ustaw dla niej 30-dniowy okres ważności, począwszy od dnia bieżącego. Data jest ustawiana za pomocą obiektu `date` języka JavaScript. Jest to dobry przykład mieszania macierzystych funkcji JavaScript z kodem jQuery. Na rysunku 5.4 zaprezentowano zrzut ekranu wykonany w przeglądarce Firefox informacji *cookie* z jej ustawionymi wartościami.

```
function hideMessage() {
  var expirDate=new Date();
  expirDate.setDate(expirDate.getDate()+30);
  document.cookie = "name=hideCookie;expires="+expirDate.toUTCString();
}
```



Rysunek 5.4. Informacja cookie z jej ustawionymi wartościami w przeglądarce Firefox

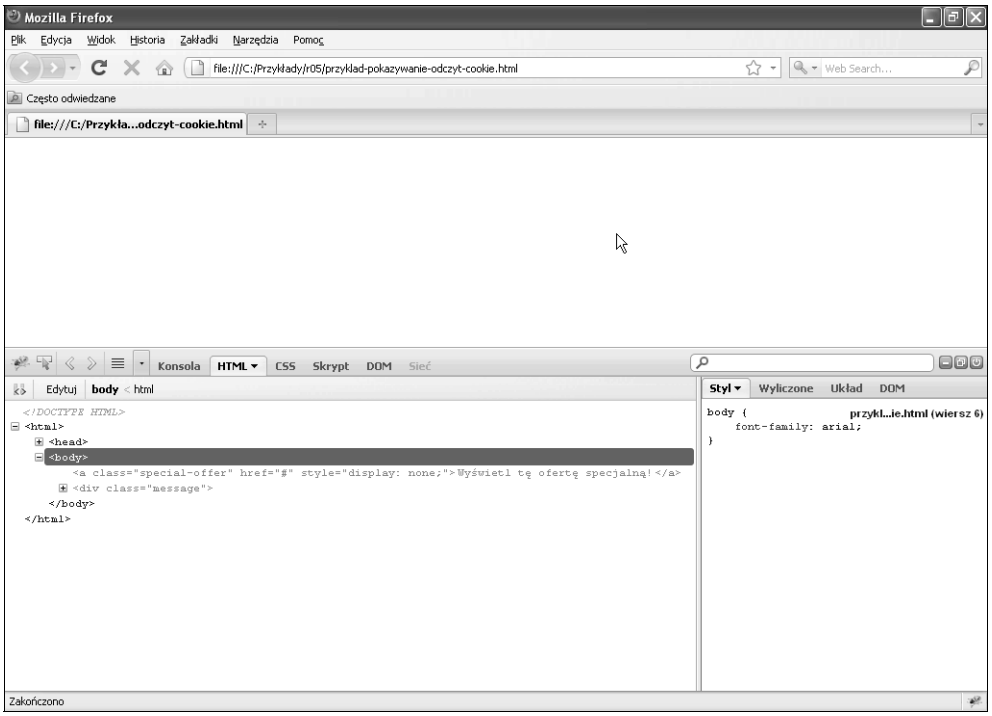
Na tym etapie powinno być możliwe wyświetlenie i ukrycie komunikatu oraz ustawienie informacji *cookie*. A co z użytkownikami, którzy już wyświetlili komunikat? Jeśli jutro powrócą do witryny, nie mogą ponownie ujrzeć tej informacji. Aby to osiągnąć, utwórz kolejną funkcję, która będzie uruchamiana przy użyciu zdarzenia `load` w celu ukrycia komunikatu, gdy zostanie znaleziona informacja *cookie* (rysunek 5.5).

1. Dodaj zmienną przypisaną informacji *cookie*. Za pomocą obiektu JavaScript `cookie` możesz uzyskać tę informację. Umożliwia to instrukcja `document.cookie`.

   ```
   var messageCookie = document.cookie;
   ```

2. Utwórz instrukcję warunkową sprawdzającą, czy zmienna `messageCookie` ma wartość, a następnie ukrywającą odnośnik oferty specjalnej. W przeciwnym razie instrukcja nie wykona żadnego działania.

   ```
   if (messageCookie) {
      // Jeśli istnieje informacja cookie komunikatu, ukryj odnośnik oferty specjalnej
      $('.special-offer').hide();
    }
    else {
      // Nie wykonuj żadnego działania
    }
   ```



Rysunek 5.5. Po kliknięciu odnośnika są ukrywane elementy komunikatu i oferty specjalnej

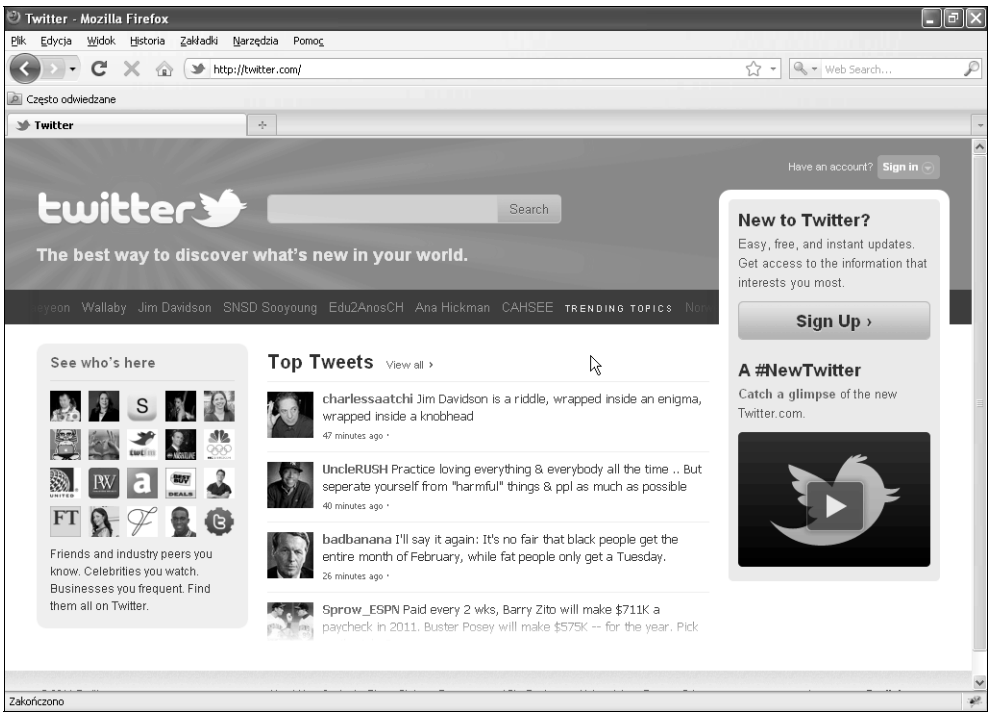
PRZEŁĄCZANIE SIĘ MIĘDZY EFEKTAMI SHOW I HIDE

Może wystąpić sytuacja, w której konieczne będzie przełączenie się między efektami `show` i `hide`. Biblioteka jQuery oferuje ciekawe rozwiązanie w postaci metody `toggle()`. Metoda ta dowiązuje program obsługi zdarzeń do zdarzenia `click` i umożliwia przełączenie się między efektami `show` i `hide` na podstawie bieżącej widoczności elementu. W przypadku poniższego przykładu istotną częścią jest ustawienie dla elementu `recipe` wartości `none` właściwości stylów CSS — przełączanie działa niezależnie od niej.

```
<style>
.recipe {display:none;}
</style>
$('.recipe-name').toggle(
 function() {
  $('.recipe').show();
 },
 function() {
  $('.recipe').hide();
 }
);
<a href="#" class="recipe-name">Ciasto cytrynowe</a>
<div class="recipe">Ciasto cytrynowe to amerykański deser uzyskiwany z soku z limonek,
➥żółtka jajek i słodzonego mleka skondensowanego w spodzie z kruchego ciasta.</div>
```

PRZESUWANIE ELEMENTÓW W GÓRĘ I W DÓŁ

Z efektem przesuwania możesz się spotkać w wielu witrynach internetowych. Najczęściej jest on obecny w galeriach obrazów, które przesuwają się w obręb widoku lub poza niego. Ponadto wraz z ostatnim dynamicznym rozwojem rozwiązań umożliwiających rozmowę w czasie rzeczywistym w portalach Facebook i Twitter coraz częstsze jest stosowanie efektów przesuwania w górę i w dół w odniesieniu do działania ostatnio wykonanego na stronie. Na rysunku 5.6 przedstawiono sposób zintegrowania ze stroną główną portalu Twitter (*http://www.twitter.com*) efektu zsuwania w dół, który jest używany w momencie publikowania nowych wiadomości. Każdorazowo po pojawieniu się nowej wiadomości jest ona przesuwana od góry, powodując przemieszczenie w dół wiadomości aktualnie wyświetlanych na stronie. Ostatecznie kolejne wiadomości znikają z widocznego obszaru strony.



Rysunek 5.6. Ze stroną główną portalu Twitter zintegrowano efekt zsuwania, który jest stosowany w momencie publikowania nowych wiadomości (© 2010 Twitter; www.twitter.com)

Metody `slideDown` i `slideUp` są konfigurowane dokładnie w taki sam sposób co metody `show` i `hide`. Dodając metodę do selektora, możesz przekazać dwa parametry opcjonalne (`duration` i `callback`). Dla osób rozpoczynających przygodę z biblioteką jQuery nazwy metod często mogą być niejasne. Metoda `slideDown` powoduje wyświetlanie elementów, a metoda `slideUp` ich ukrywanie.

```
$('.message').slideDown();
```

WYŚWIETLANIE ALTERNATYWNYCH OPCJI WYSZUKIWANIA PRZY UŻYCIU METODY SLIDETOGGLE

Funkcja wyszukiwania stanowi integralną część witryn internetowych. Całkiem możliwe, że właśnie z tego powodu firmę Google uznaje się obecnie za tę, która odniosła największy sukces. Każdy użytkownik oczekuje możliwości wyszukiwania i łatwego znajdowania tego, czego szuka w witrynie. Utworzenie lepszego interfejsu użytkownika zapewniającego możliwość natychmiastowego znalezienia wszystkiego, co oferuje witryna, jest znaczącym wyznacznikiem rozwoju. Firma Mozilla może się pochwalić dużą społecznością programistów, którzy tworzą dodatki dla przeglądarki Firefox. Rysunek 5.7 prezentuje przykład jej paska wyszukiwania z opcjami zaawansowanymi. Jeśli klikniesz te opcje, zawierający je pasek wyszukiwania zostanie rozwinięty. Jest to użyteczna funkcja, ponieważ umożliwia rozszerzenie zakresu wyszukiwania bez opuszczania bieżącej strony. Funkcję tę z łatwością możesz dodać przy użyciu metody `slideToggle` biblioteki jQuery.

Rysunek 5.7. Witryna internetowa dodatków przeglądarki Firefox z zastosowanym efektem zsuwania opcji wyszukiwania zaawansowanego

W przypadku pokazywania i ukrywania elementów metoda `slideToggle` bardzo przypomina metodę `toggle`. Jedyną różnicą jest to, że metoda `slideToggle` nie jest już dowiązywana do zdarzenia `click`. Jeśli element został już pokazany, a metoda `slideToggle` wywołana, stosowany jest efekt `slideUp`. Odwrotnie wygląda to w przypadku ukrywania elementu.

```
$('.message').slideToggle();
```

Poniżej prezentuję, jak utworzyć menu wyszukiwania zaawansowanego zsuwane przy użyciu metody `slideToggle()` (podobnie jak w przykładowej witrynie internetowej przedstawionej na rysunku 5.7).

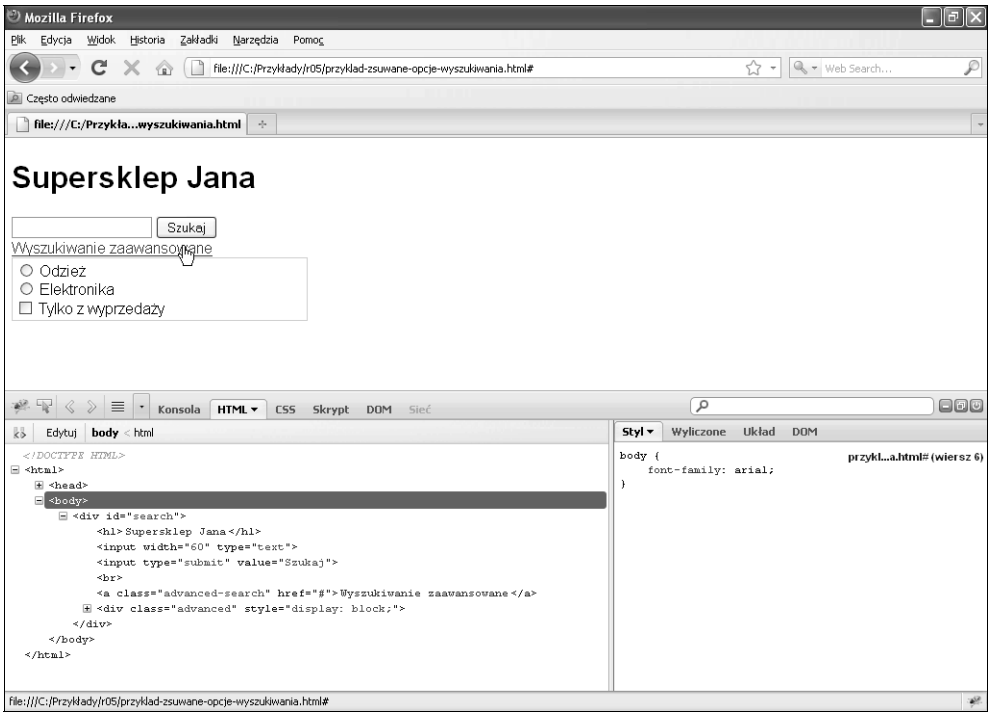
1. Utwórz kod HTML dla pola wyszukiwania i opcji wyszukiwania zaawansowanego.

```
<style>
body {font-family:arial;}
.advanced {display:none;
padding:3px;
border:1px solid #ccc;
width:300px;
}
</style>
<div id="search">
  <h1>Supersklep Jana</h1>
  <input type="text" width="60" />
  <input type="submit" value="search"/><br>
  <a href="#" class="advanced-search">Wyszukiwanie zaawansowane</a>
    <div class="advanced">
      <input type="radio" name="category"/> Odzież<br/>
      <input type="radio" name="category"/> Elektronika<br/>
      <input type="checkbox" name="sale"/> Tylko z wyprzedaży<br/>
    </div>
  </div>
```

2. Utwórz instrukcję selektora dla elementu `advanced-search` i dowiąż do niego program obsługi zdarzenia `click`. Wewnątrz programu obsługi zdarzenia skonfiguruj selektor elementu `advanced` i dodaj do niego metodę `slideToggle`.

```
$('.advanced-search').bind('click',function(){
    $('.advanced').slideToggle();
       });
```

Każdorazowo po kliknięciu przycisku opcji wyszukiwania zaawansowanego element tych opcji jest rozwijany lub zwijany (zależnie od jego stanu w chwili ładowania strony). W tym przypadku element wyszukiwania zaawansowanego jest ukrywany podczas ładowania strony za pomocą stylu CSS. Na rysunku 5.8 przedstawiono dane wyjściowe w oknie przeglądarki Firefox i rozszerzenia Firebug. Podobnie do efektów `hide` i `show` element jest wyświetlany przez kod jQuery poprzez dodanie do niego stylu wstawianego `display:block`.



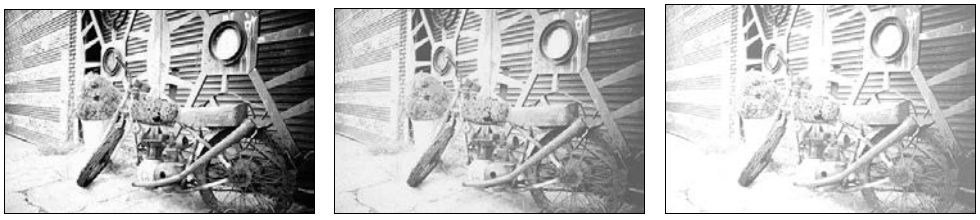
Rysunek 5.8. Dane wyjściowe w oknie przeglądarki Firefox i rozszerzenia Firebug — podobnie do efektów hide i show element jest wyświetlany przez kod jQuery poprzez dodanie do tego elementu stylu wstawianego display:block

ZNIKANIE ELEMENTÓW

Efekty znikania i pojawiania się mogą wzbogacić elementy witryny internetowej o kolejny wymiar interaktywności. Najczęściej efekt znikania jest stosowany w galeriach obrazów lub pokazach slajdów, w przypadku których jeden obraz pojawia się, gdy inny znika. Jeszcze kilka lat temu wydawało się, że jedynym sposobem uzyskania takiego efektu jest użycie technologii Flash, która umożliwia utworzenie animowanego pokazu slajdów, lub skorzystanie z zaawansowanych możliwości języka JavaScipt wymagających napisania wielu wierszy kodu.

Biblioteka jQuery pozwoliła na użycie własnych efektów JavaScript bez konieczności bezpośredniej interakcji z trudnym do opanowania interfejsem API tego języka. Dzięki efektom udostępnionym przez jQuery ten sam animowany pokaz slajdów może być obecnie implementowany w języku JavaScript. Zastosowanie pokazów slajdów opartych na tym języku zamiast na technologii Flash zapewnia korzyść związaną z optymalizacją wyszukiwarki. Wynika to stąd, że nie wszystkie wyszukiwarki mają możliwość indeksowania treści znalezionej w plikach Flash.

Kluczową właściwością CSS używaną w połączeniu z efektem znikania elementów na stronie internetowej jest właściwość `opacity` (rysunek 5.9). Pobiera ona wartość z zakresów `0` – `100` lub `0.0` – `1.0` i jest stosowana w metodach `fadeIn` i `fadeOut`.



Rysunek 5.9. Przykład procesu znikania obrazu w przypadku użycia właściwości opacity

Efekt `fadeIn` jest tworzony podobnie do efektu `show`. Dostępne są dwa parametry opcjonalne, które mogą zostać przekazane: `duration` i `callback`. Parametr `duration` określa czas odtwarzania animacji, który możesz ustawić przy użyciu słów kluczowych `fast` lub `slow`, jak również wyrazić w milisekundach (600, 200, 700 itd.). Parametr `callback` umożliwia połączenie z funkcją wykonywaną po zakończeniu działania efektu `fadeIn`.

```
$(selector).fadeIn(duration, callback)
```

Efekt `fadeOut` różni się od efektu `fadeIn` tylko tym, że zamiast zwiększania nieprzezroczystości elementu powoduje jej zmniejszanie. Efekt `fadeTo` umożliwia określenie poziomu nieprzezroczystości, który ma zostać osiągnięty przez wybrany element.

```
$(selector).fadeIn(duration, opacity, callback)
```


TWORZENIE PROSTEJ GALERII OBRAZÓW PRZY UŻYCIU PRZEJŚCIA Z EFEKTEM ZNIKANIA

Aby zademonstrować sposób użycia znikania w witrynie internetowej, w tym podrozdziale omówię proces tworzenia prostej galerii obrazów. Zawiera ona pięć obrazów i listę numerów, które można kliknąć w celu zmiany obrazu. W momencie zmiany obrazów bieżący obraz znika, a nowy pojawia się. Podzielę proces pisania skryptu na wiele kroków, aby umożliwić prześledzenie stopniowego tworzenia skryptów coraz bardziej dynamicznych.

1. Najpierw utwórz prosty kod HTML. Dodaję cały kod HTML niezbędny do obsługi pokazu slajdów za pośrednictwem kodu jQuery. Dzięki temu skrypt ma duże możliwości przenoszenia, a ponadto jest prosty do skonfigurowania.

   ```
   <div class="container">
     <h1>Galeria obrazów jQuery.</h1>
   </div>
   ```

2. Następnie skonfiguruj arkusz stylów, aby zapewnić poprawne rozmieszczenie galerii obrazów na stronie.

   ```
   body {
   font-family:arial;
   }
   ```

```
ul#nav {
list-style-type:none;
margin:10px 0 10px;
padding:0;}

ul#nav li {
float:left;
width:30px;}

ul#nav li a {text-decoration:none;
background:#05609A;
color:#fff;
padding:5px;}

ul#nav li a.active {
background:#B4F114;
}

.slide-image {width:400px;
height:300px;
border:2px solid #05609A;
overflow:hidden;
}

.slide-image img {
display:none;
}
```


3. Utwórz tablicę do przechowywania wszystkich obrazów i przypisz ją zmiennej `slideArray`. Tablica ta określa liczbę odnośników nawigacyjnych do utworzenia (jest ona zależna od liczby znajdujących się w tablicy obrazów). W dowolnym momencie możesz zmniejszyć lub zwiększyć liczbę obrazów, a skrypt zostanie automatycznie dostosowany.

```
var slideArray = [
"ansel_adams1.jpg",
"ansel_adams2.jpg",
"ansel_adams3.jpg",
"ansel_adams4.jpg",
"ansel_adams5.jpg"
]
```

Utwórz również zmienną o nazwie `imgDir` do przechowywania wartości ścieżki względnej lub bezwzględnej folderu, w którym znajdują się obrazy do pokazu slajdów. Zmienna ta jest dołączana w dalszej części procesu podczas konfigurowania obrazów w pokazie.

```
var imgDir = 'obrazy/ansel_adams';
```

4. Dodaj element `slide-image` do drzewa DOM wewnątrz elementu `container`. W tym wstawionym elemencie są przechowywane wszystkie obrazy dodawane do drzewa DOM.

```
$('.container').append('<div class="slide-image"></div>');
```

5. Po utworzeniu elementu `slide-image` konieczne jest dodanie obrazu, który zostanie wyświetlony w elemencie po załadowaniu strony.

```
$('.slide-image').html('<img src="obrazy/'+slideArray[0]+'"/>');
```

6. Po elemencie `slide-image` w drzewie DOM umieść element `nav` listy nieuporządkowanej. Element ten zawiera wszystkie odnośniki do obrazów w pokazie slajdów.

```
$('.slide-image').after('<ul id="nav"></ul>');
```

7. Przy użyciu właściwości `length` określ liczbę pozycji tablicy `slideArray` i przypisz tę wartość zmiennej `lengthArray`.

```
var slideLength = slideArray.length;
```

8. Utwórz pętlę `for` przetwarzającą wszystkie pozycje zmiennej `slideArray`, używając zmiennej `slideLength` do ograniczenia do pięciu liczby wykonań pętli.

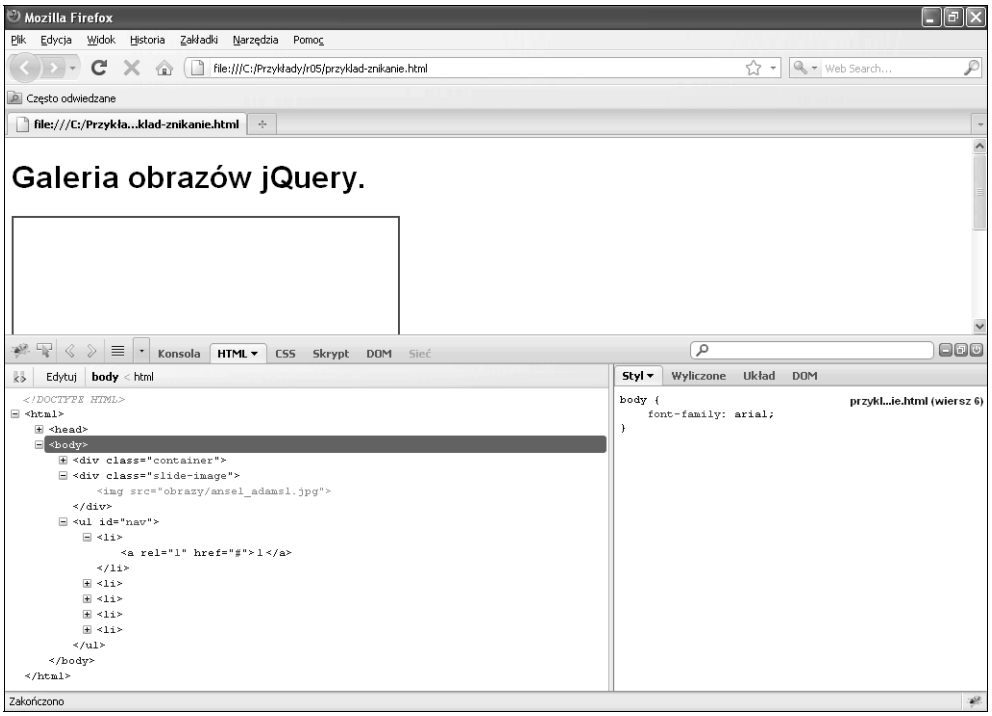
```
for(i=0; i < slideLength; i++){
}
```


9. Ponieważ w tablicy znajduje się pięć pozycji, zwrócona wartość to 5. Dodaję zmienną o nazwie `slideText` i przypisuję jej wartość `1 + i` (indeks). Zapewnia to, że tekst odnośnika rozpoczyna się od wartości 1 i kończy na wartości 5, zamiast rozpoczynać się od wartości 0 i kończyć na wartości 4.

```
for(i=0; i <= slideLength; i++){
  var slideText = i + 1;
}
```

10. Następnie, używając elementu `nav` dodanego w kroku 5. do drzewa DOM, dołączam element `li` dla każdego indeksu w tablicy. Wewnątrz każdego elementu `li` umieszczam znacznik `anchor` z atrybutem `rel` o ustawionej wartości zmiennej `slideText`. Dodatkowo wstawiam zmienną `slideText` jako tekst odnośnika. Atrybut `rel` umożliwia wybranie wstawianego obrazu. Na rysunku 5.10 przedstawiono pętlę przetwarzaną w przeglądarce z wynikiem w postaci kodu HTML, którego wyświetlenie umożliwia okno rozszerzenia Firebug.

```
for(i=0; i < slideLength; i++){
  var slideText = i + 1;
$('#nav').append('<li><a href="#"
rel="'+slideText+'">'+slideText+'</a></li>');
}
```



Rysunek 5.10. Dane wyjściowe pętli for w oknie przeglądarki oraz otwarte okno rozszerzenia Firebug z utworzonym kodem HTML

11. Po utworzeniu listy nawigacyjnej konieczne jest skonfigurowanie zdarzenia `click` dla każdego elementu odnośnika. Aby zdefiniować podstawowe zdarzenie `click`, użyj metody `bind` do dowiązania instrukcji selektora elementu `nav` do funkcji programu obsługi zdarzeń.

```
$('#nav li a').bind('click', function(){
});
```

12. Dodaj zmienną o nazwie `numSlide` w celu przechowywania wartości atrybutu `rel`. Wartość ta będzie ustawiana tylko w momencie kliknięcia znacznika odnośnika z numerem slajdu, czyli wyzwalania zdarzenia `click`, a nie dowiązywania.

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
});
```

13. Wybierz element `slide-image` i wstaw znacznik obrazu z dołączonymi zmiennymi `imgDir` i `numSlide`. Po kliknięciu odnośnika do strony spowoduje to dodanie poprawnej nazwy obrazu i ścieżki serwerowej.

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
  $('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
});
```

14. Zawsze dodatkową zaletą jest znajomość położenia w obszarze nawigacji po kliknięciu odnośnika. Do zdarzenia `click` dodaję więc dwie kolejne instrukcje. Pierwsza z nich usuwa wszystkie instancje aktywnej klasy ze znaczników odnośników na liście nawigacyjnej, które wcześniej zawierały klasę. Druga instrukcja dodaje aktywną klasę tylko do znacznika odnośnika kliknięty.

```
$('#nav li a').bind('click', function(){
var numSlide = $(this).attr('rel');
$('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
$('#nav li a').removeClass('active');
$(this).addClass('active');
});
```

Skoro już wiesz, jak utworzyć prosty pokaz slajdów, zademonstruję sposób dodawania efektów znikania w celu autentycznego ożywienia go. Zastosowanie efektu znikania obrazu wymaga tylko jednego wiersza.

15. Wybierz znacznik obrazu potomny względem elementu `slide-image` i dodaj efekt `fadeIn()`.

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
  $('.slide-image').html('<img src="'+imgDir + numSlide+'.jpg"/>');
  $('.slide-image img').fadeIn();
  $('#nav li a').removeClass('active');
  $(this).addClass('active');
});
```

16. Aby zapewnić, że pierwszy slajd zostanie wczytany podczas ładowania strony, utwórz instrukcję selektora wybierającą pierwszy znacznik odnośnika w obszarze nawigacyjnym i przeprowadź symulację kliknięcia tego odnośnika.

```
$('#nav li a').eq(0).click();
```


ZASTOSOWANIE OPÓŹNIENIA W CELU UTWORZENIA ZSYNCHRONIZOWANEJ ANIMACJI

Ponieważ animacje to zazwyczaj seria zdarzeń występujących w określonym przedziale czasu, podstawowym wymogiem dla uzyskania zsynchronizowanej animacji jest możliwość opóźniania elementów. Biblioteka jQuery oferuje możliwość dodania opóźnienia do animacji przy użyciu metody `delay`.

Metoda `delay` została niedawno dodana do wersji 1.4 biblioteki, aby umożliwić zastosowanie opóźnienia dla metod następujących w dalszej kolejności, które są dołączane przez łączenie w łańcuch. Metoda ta może być użyta tylko w przypadku efektów biblioteki jQuery. Jeśli szukasz bardziej elastycznej funkcji czasowej, zwróć uwagę na macierzystą funkcję `setTimeout` języka JavaScript.

Jeżeli komunikat ma zostać wyświetlony użytkownikom, a następnie ukryty po upływie określonego czasu, metoda `delay` okaże się idealnym rozwiązaniem. W poniższym przykładzie chcę wyświetlić komunikat w momencie zatrzymania przez użytkownika kursora myszy na odnośniku. Jeśli użytkownik przesunie kursor poza obręb odnośnika, po upływie 10 sekund komunikat ma zniknąć.

1. Utwórz kod HTML dla elementu podpowiedzi z komunikatem i odnośnika wyświetlającego tę podpowiedź, na którym użytkownik zatrzymuje kursor myszy w celu ujrzenia komunikatu.

```
<a href="#" class="show-tip">Dowiedz się czegoś o brzoskwiniach</a>
      <div class="tool-tip">
      Choć nazwa botaniczna Prunus persica wskazuje na perskie pochodzenie
      ↪brzoskwini, w rzeczywistości wywodzi się ona z Chin, w których była
      ↪uprawiana od początków istnienia kultury chińskiej. Brzoskwinie,
      ↪o których w kronikach wspominano już 1000 lat p.n.e., były ulubionymi
      ↪owocami królów i cesarzy. W ostatnim czasie historia uprawy brzoskwiń
      ↪w Chinach została znacznie wydłużona na podstawie kilku oryginalnych
      ↪manuskryptów datowanych na 1100 lat p.n.e.
      </div>
```

2. Następnie utwórz zdarzenie `hover`, które dokonuje przełączenia między zdarzeniami `mouseenter` i `mouseleave`. W pierwszej instrukcji (zdarzenie `mouseenter`) wybierz komunikat podpowiedzi i spowoduj jego pojawienie się po upływie 900 milisekund.

```
$('.show-tip').hover(
  function(){
  $('.tool-tip').fadeIn(900);
  },
  function() {
  });
```

3. W drugiej instrukcji (zdarzenie `mouseleave`) wybierz komunikat podpowiedzi, lecz tym razem opóźnij jego wyświetlenie o 10 000 milisekund (10 sekund), a następnie spowoduj jego zniknięcie po upływie 900 milisekund.

```
$('.show-tip').hover(
  function(){
    $('.tool-tip').fadeIn(900);
},
  function() {
    $('.tool-tip').delay(10000).fadeOut(900);
  });
```

ŁAŃCUCHOWE ŁĄCZENIE WIELU EFEKTÓW

Na tym etapie naprawdę dobrze powinno być znane zagadnienie łączenia w łańcuch w przypadku biblioteki jQuery. Umożliwia ono dodanie wielu metod do tej samej instrukcji. Jest to pomocne w ograniczeniu ilości kodu i zwiększeniu wydajności skryptów.

W poniższym przykładzie używam łączenia w łańcuch w celu zilustrowania dodawania wielu metod do instrukcji selektora. Najpierw podpowiedź jest ukrywana, a następnie element pojawia się w ciągu 900 milisekund. Po wystąpieniu jednosekundowego opóźnienia podpowiedź znika w ciągu 900 milisekund.

```
$('.tooltip').hide().fadeIn(900).delay(10000).fadeOut(900);
```

Gdyby tę samą instrukcję zapisano bez łączenia w łańcuch, wymagałoby to trzech osobnych instrukcji w trzech oddzielnych wierszach.

```
$('.tool-tip').fadeIn(900);
$('.tool-tip').delay(10000);
$('.tool-tip').fadeOut(900);
```

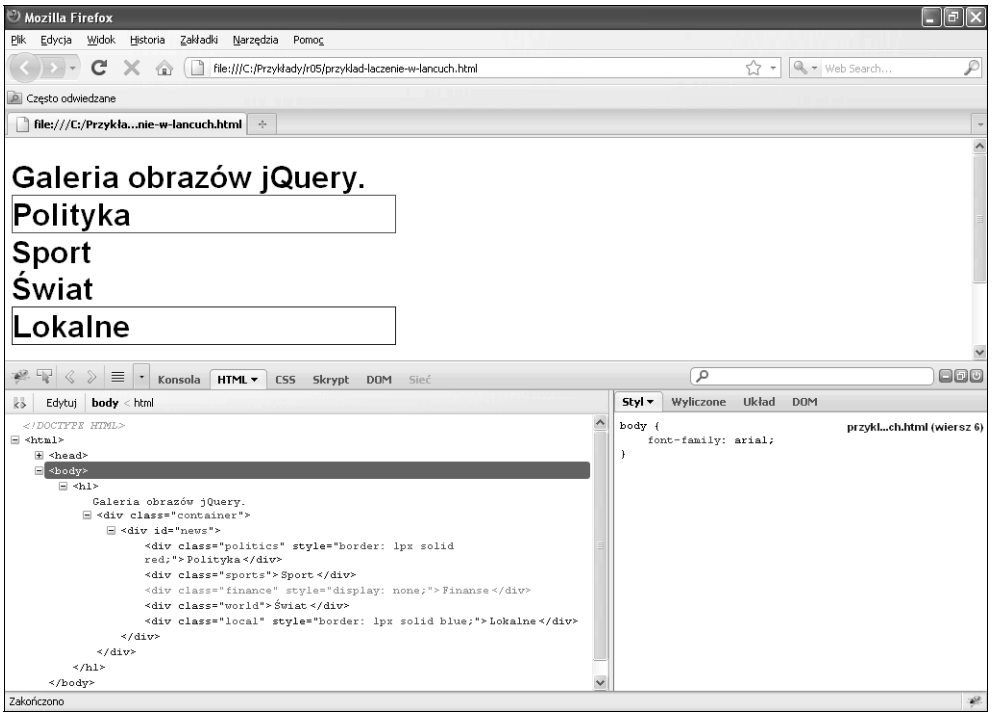
Choć powyższa sekwencja instrukcji jQuery daje ten sam wynik co instrukcja z zastosowanym łączeniem w łańcuch, pierwsze rozwiązanie pozwala zaoszczędzić miejsce i zapewnia większą przejrzystość kodu. Rozważmy scenariusz, w którym musisz wybrać trzy różne elementy `li`, używając ich odpowiednich nazw `id`, oraz dla każdego elementu zastosować inny styl.


```
<ul id="news">
    <li id="politics">Polityka</li>
    <li id="sports">Sport</li>
    <li id="finance">Finanse</li>
    <li id="world">Świat</li>
    <li id="local">Lokalne</li>
</ul>
```

Operację tę możesz wykonać na dwa sposoby. Pierwszy polega na utworzeniu trzech instrukcji, z których każda wybiera element i stosuje dla niego styl CSS.

```
$('#politics').css('border','1px solid red');
$('#finance').css('display','none');
$('#local').css('border','1px solid green');
```

Drugi sposób wykonania operacji sprowadza się do połączenia w łańcuch wszystkich elementów i metod w ramach jednej instrukcji z wykorzystaniem metody `end()`. Metoda ta określa koniec bieżących metod i umożliwia ponowne rozpoczęcie za nią łańcucha nowych metod, nie powodując ich nakładania się na metody znajdujące się przed nią. Na rysunku 5.11 przedstawiono wynik połączenia w łańcuch wielu metod z poprzedniego przykładowego kodu.



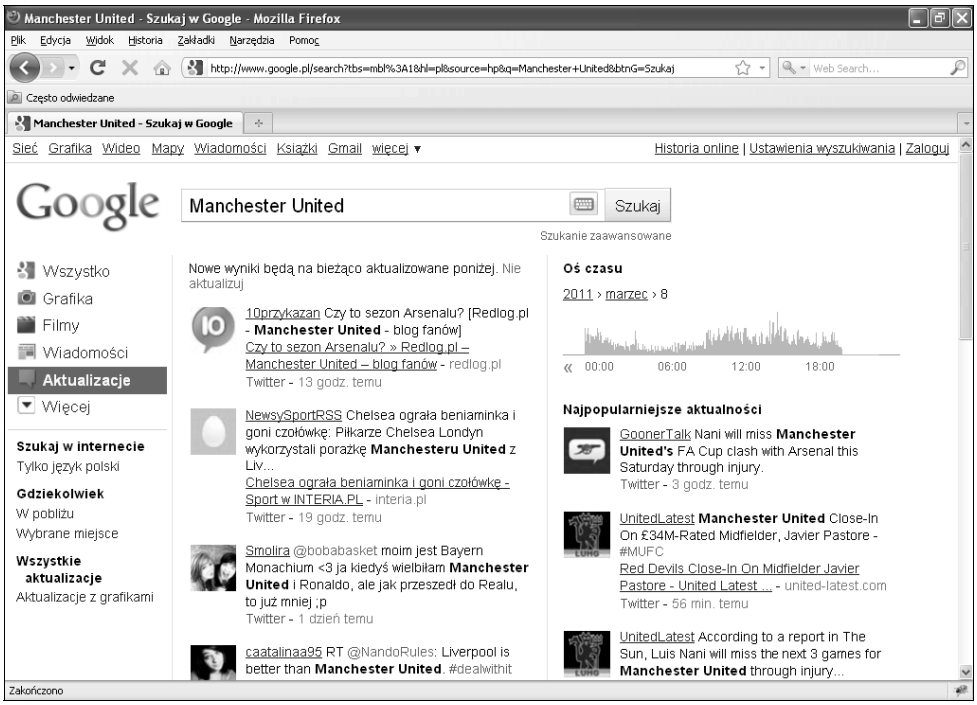
Rysunek 5.11. Końcowy wynik połączenia w łańcuch wielu metod z poprzedniego przykładowego kodu

```
$('#news').find('#politics').css('border','1px solid red').end().find('#finance').
hide().end().find('#local').css('border','1px solid blue');
```

TWORZENIE PASKA KANAŁU INFORMACYJNEGO PRZY UŻYCIU WIELU EFEKTÓW

Wraz ze zwiększonym zapotrzebowaniem na możliwość zdobycia szerszej grupy odbiorców poprzez dostarczanie informacji za pośrednictwem kanałów informacyjnych RSS (*Really Simple Syndication*) coraz częściej są stosowane paski informacyjne (ang. *tickers*) i widżety. Paski informacyjne mogą mieć postać zarówno ramki z 10 artykułami prasowymi odświeżanymi przy każdym przeładowaniu strony, jak i bardziej zaawansowanego rozwiązania, które w czasie rzeczywistym wyświetla aktualizacje, tak jak to jest w przypadku wyszukiwarki Google Realtime (rysunek 5.12).

Poniżej omówię proces tworzenia prostego paska kanału informacyjnego, który pobiera treść statyczną przy użyciu wielu efektów znikania i zsuwania pozycji w widoczny obszar. Choć po wprowadzeniu kilku modyfikacji pasek ten może zostać rozszerzony o kanały informacyjne RSS czasu rzeczywistego, na początek wystarczające powinno być wykonanie następujących kroków:



Rysunek 5.12. Wyszukiwarka Google Realtime, w której w momencie opublikowania pojawia się nowa treść z portalu Twitter (reprodukcja z 2010 © Google)

1. Pierwszym krokiem jest utworzenie podstawowej struktury HTML. Celem jest zapewnienie jak największej dynamiczności paska informacyjnego, tak aby mógł zostać umieszczony na dowolnej stronie bez konieczności jej modyfikowania. W tym przypadku utworzyłem stronę z elementem H1, którego użyję w instrukcji selektora w celu wstawienia paska informacyjnego bezpośrednio za tym elementem.

```
<body>
  <h1>Najnowsze informacje dotyczące biblioteki jQuery</h1>
</body>
```

2. W tablicy o nazwie `newsArray` utworzyłem nagłówki. Jest to treść statyczna cyklicznie wyświetlana w obrębie paska informacyjnego.

```
var newsArray = [
"Lekarze zadowoleni z postępów Kubicy",
"Małysz do mistrza: Świetna robota, gratuluję",
"Koniec konfliktu Polonii ze Smudą",
"Chcą więcej leków na astmę",
"Warta kupi miejsce w ekstraklasie?",
"Real zniszczył Malagę. Hat trick Ronaldo",
"Magic odrobili 24 punkty straty w Miami",
"MŚ: Michael Uhrmann też kończy karierę",
"Tajner: Stoch może zastąpić Małysza",
"Małysz: Skoki to całe moje życie"
];
```

3. Utwórz dwie zmienne. Zmienna `newsLength` określa długość tablicy `newsArray`. Zmienna `newsInterval` przechowuje wartość liczbową wyrażoną w milisekundach, która określa częstotliwość pobierania nowego nagłówka i wstawiania go do paska informacyjnego.

```
var newsLength = newsArray.length;
var newsInterval = 2000;
```

4. Wybierz element `H1` i wstaw za nim elementy listy nieuporządkowanej `news-feed`.

```
$('h1').after('<ul id="news-feed"></ul>');
```

5. Utwórz pętlę `for` w celu wykonywania jej dla wszystkich nagłówków w tablicy `newsArray`. Dla każdego nagłówka dodaj pozycję listy nieuporządkowanej `news-feed` z nagłówkiem umieszczonym między dwoma znacznikami `li`.

```
for(i=0; i < newsLength; i++){
$('#news-feed').append('<li>'+newsArray[i]+'</li>');
}
```

6. Następnie utwórz funkcję o nazwie `slideHeadline()`, która zawiera wszystkie efekty umożliwiające działanie paska informacyjnego. Pierwsza instrukcja wewnątrz funkcji wybiera ostatnią pozycję listy nieuporządkowanej `news-feed`, klonuje ją i przy użyciu metody `prepend` ponownie dodaje do listy, lecz na jej początek.


```
function slideHeadline() {
$('#news-feed li:last').clone().prependTo('#news-feed').hide();
}
```

7. Druga instrukcja dodana do funkcji wybiera pierwszą pozycję listy (sklonowany element, który został właśnie utworzony w kanale informacyjnym) i stosuje dla niej efekt `slideDown`.

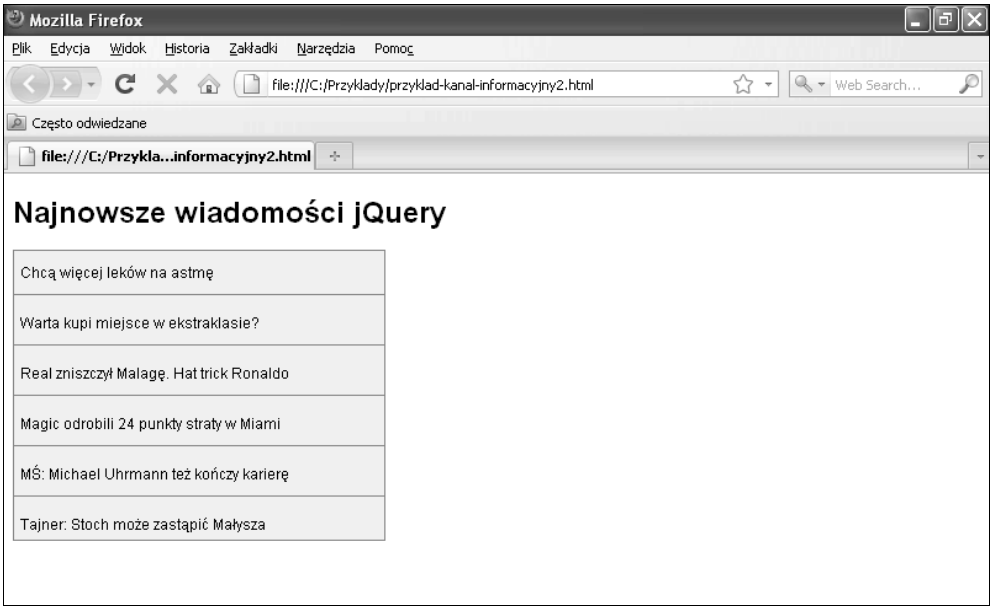
```
function slideHeadline() {
$('#news-feed li:last').clone().prependTo('#news-
feed').css('display','none');
$('#news-feed li:first').slideDown();
        }
```

8. Pierwsza pozycja listy nie tylko ma zostać przesunięta w czasie wynoszącym 500 milisekund, ale też ma pojawić się w czasie równym 1000 milisekund po umieszczeniu jej na swoim miejscu. W tym celu w tej samej instrukcji zawierającej metodę `slideDown` łańcuchowo wstawiam metodę `fadeIn`.

```
function slideHeadline() {
$('#news-feed li:last').clone().prependTo('#news-
feed').css('display','none');
$('#news-feed li:first').fadeIn(1000).slideDown(500);
}
```

9. Ostatnia instrukcja dodawana do funkcji `slideHeadline` usuwa ostatnią pozycję listy. Poniższe trzy instrukcje następują kolejno po sobie. Dzięki temu jest symulowany ładnie prezentujący się efekt pojawiania się i zsuwania (rysunek 5.13).

```
function slideHeadline() {
$('#news-feed li:last').clone().prependTo('#news-
feed').css('display','none');
$('#news-feed li:first').fadeIn(1000).slideDown(500);
$('#news-feed li:last').remove();
}
```



Rysunek 5.13. Demonstracja pojawiania się górnego elementu, a następnie jego zsuwania się w obręb widocznego obszaru przy jednoczesnym usunięciu dolnego elementu z pola widzenia

10. Poniższy ostatni wiersz kodu JavaScript jest prawdopodobnie najważniejszy. Muszę zdefiniować macierzystą funkcję JavaScript `setInterval`, aby wykonać funkcję `slideHeadline` po upływie 2000 milisekund (`newsInterval`). Funkcja ta ciągle wykonuje pętlę dla paska informacyjnego. Bez niej jego działanie nie byłoby możliwe.

```
setInterval(slideHeadline, newsInterval);
```

Funkcja `setInterval` to macierzysta funkcja zegara języka JavaScript, która pozwala na uruchomienie określonej funkcji po upływie ustalonego czasu. Choć istnieje podobna funkcja zegara języka JavaScript o nazwie `setTimeout`, różni się ona od funkcji `setInterval`, która wykonuje pętlę do momentu zakończenia jej przez użytkownika, przede wszystkim tym, że jest wykonywana tylko raz. Zakończenie działania tych funkcji zegara umożliwiają dwie funkcje: `clearTimeout()` i `clearInterval()`. Do obsługi interfejsów API efektów w bibliotece jQuery są używane metody `setTimeout` i `setInterval`.

TWORZENIE ZAAWANSOWANYCH ANIMACJI

Biblioteka jQuery oferuje metodę `animate`, która pozwala tworzyć niestandardowe animacje. Zamiast łączenia w łańcuch metod `fade`, `slide` i `show`, które mają dość ograniczony zasięg, skorzystaj z metody `animate` umożliwiającej zastosowanie dowolnej właściwości CSS kontrolowanej przez wartość liczbową. W tabeli 5.2 wyszczególniono właściwości CSS, które mogą być użyte w przypadku metody `animate`.

Tabela 5.2. Typowe właściwości CSS, które mogą zostać zastosowane w przypadku metody animate

Właściwość CSS	Przykładowa wartość
`opacity`	`0.5`
`top`	`10 px`
`height`	`100 px`
`width`	`200 px`
`margin`	`10 px`
`padding`	`15 px`

TWORZENIE GALERII OBRAZÓW Z NAGŁÓWKAMI TEKSTOWYMI PRZY UŻYCIU ZAAWANSOWANYCH ANIMACJI

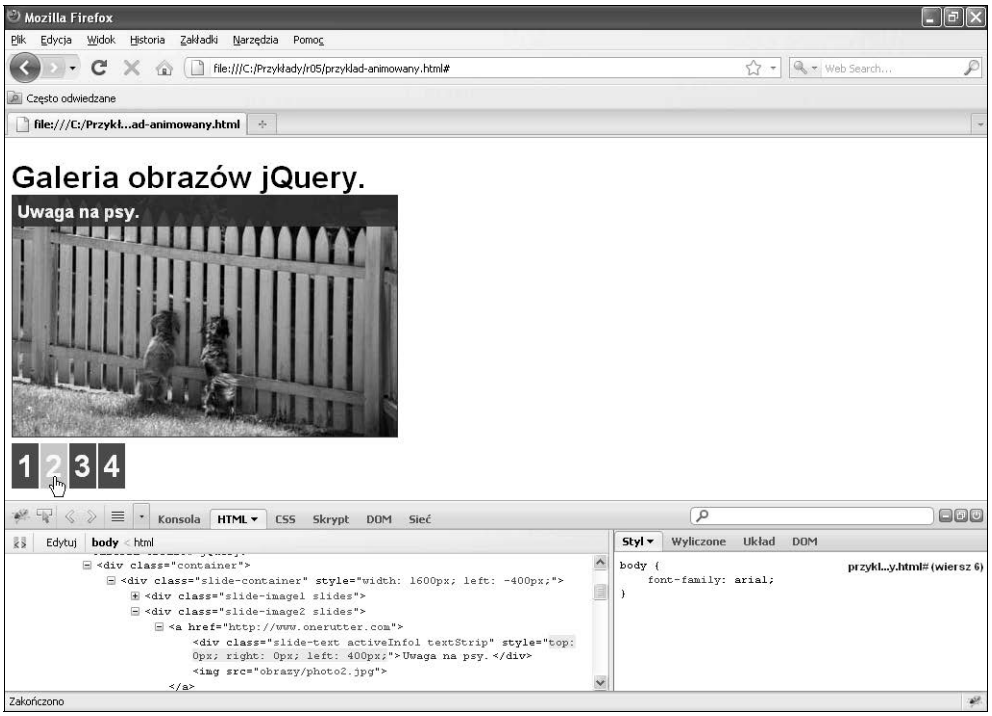
Jeszcze około dwóch lat temu korzystałem z technologii Flash i języka ActionScript. Postanowiłem wtedy zmienić styl pracy, rozpoczynając przygodę z językiem JavaScript i biblioteką jQuery. Zmiana ta wynikała z uzyskania większych możliwości kontrolowania modelu DOM przy użyciu języka JavaScript, a także z braku obsługi technologii Flash i języka ActionScript w przypadku urządzeń przenośnych takich jak iPhone. Choć technologia Flash sprawdza się podczas tworzenia zaawansowanych animacji, język JavaScript od dawna obsługuje galerie obrazów. Podstawową korzyścią wynikającą z opanowania języka ActionScript jest w moim przypadku jego podobieństwo do języka JavaScript pod względem sposobu obsługi zdarzeń i efektów.

Opanowanie podstawowych zagadnień związanych z dowolnym językiem programowania stanowi ogromną korzyść przy podejmowaniu próby nauki innego języka. Większość pojęć jest taka sama. Zmianie ulega jedynie składnia. Zarówno język ActionScript 3, jak i język JavaScript są oparte na języku ECMAScript. Z tego powodu mają one wiele podobieństw. Więcej niż raz podczas pracy z językiem JavaScript miałem wrażenie déjà vu, ponieważ jego składnia i konwencje są zbliżone do tych z języka ActionScript.

W 2006 r. utworzyłem swój pierwszy animowany pokaz slajdów oparty na technologii Flash 8 XML. Była to naprawdę zgrabna i niewielka aplikacja. Gdy sformatowany plik XML był przekazywany do pliku Flash, przetwarzał on kod XML, dodawał slajdy i stosował dla nich animację, umożliwiając użytkownikowi kliknięcie każdego slajdu i kierując go do odnośnika (rysunek 5.14). Byłem bardzo dumny z tego skryptu, do tego stopnia, że opublikowałem go na moim blogu (*http://onerutter.com*). Skrypt został pobrany ponad 13 tys. razy. Plik Flash o wielkości 5 kB zawierał 136 wierszy kodu, który był dość zwarty i robił w tamtym czasie wrażenie. Sytuacja uległa zmianie, gdy zacząłem pracować z biblioteką jQuery i uświadomiłem sobie, o ile mniej wierszy kodu mogę napisać!

Rysunek 5.14. Galeria Flash 8 XML, którą utworzyłem w 2006 r.

Przedstawiony poniżej kurs tworzenia galerii animowanych obrazów pozwala uzyskać galerię podobną do zbudowanej przeze mnie w 2006 r. przy użyciu technologii Flash. Różnica jest jednak taka, że tym razem galeria nie bazuje na kodzie XML i wymaga tylko około 85 wierszy kodu. W razie potrzeby możesz ją rozbudować przez dodanie obsługi kanałów informacyjnych XML — decyzja należy do Ciebie. Na rysunku 5.15 zilustrowano działanie pokazu slajdów.



Rysunek 5.15. Sposób działania pokazu slajdów

1. Jedyny kod HTML wymagany na stronie to element o nazwie `container`. W przypadku tego elementu niezbędny kod HTML w całości jest przetwarzany przez skrypt obsługujący model DOM.

```
<div class="container"></div>
```

2. Następnie konieczne jest zdefiniowanie arkusza stylów dla galerii obrazów. Jeśli postanowisz użyć obrazów o różnej wielkości, musisz zmodyfikować właściwość `.container` w celu uwzględnienia nowego rozmiaru. Utworzono klasę `.text-strip`, aby wyświetlić tekst u góry obrazów w momencie modyfikowania ich.

```
body {font-family:arial;}

ul#nav {
list-style-type:none;
margin:10px 0 10px;
padding:0;}

ul#nav li {
float:left;
width:30px;}

ul#nav li a {text-decoration:none;
background:#05609A;
color:#fff;
```

```
padding:5px;}

ul#nav li a.active {
background:#B4F114;}

.container {position:relative;
height:250px;
width:400px;
border:1px solid #333;
overflow:hidden;}

.slide-container {
position: absolute;
top: 0;
left: 0;}

.slides {
float:right;}

.slide-text {
display:none;
font-size:18px;}
img {border:0;}

.textStrip {top:0px;
display:block;
position:absolute;
left:-400px;
padding:5px;
background:#333333;
opacity:.9;
color:#ffffff;
width:100%;
}
```


3. Pierwszym krokiem jest utworzenie trzech tablic. Pierwsza tablica o nazwie `slideArray` służy do przechowywania plików obrazów, które mają zostać wyświetlone jako slajdy podczas pokazu. Druga tablica o nazwie `textArray` przechowuje nagłówki pojawiające się w przypadku każdego slajdu. W trzeciej tablicy o nazwie `urlArray` są przechowywane adresy URL, które mają zostać zastosowane dla każdego ze slajdów. Tablice te mogą przechowywać taką liczbę pozycji, jaka ma zostać wyświetlona w ramach pokazu.

```
var slideArray = ["photo1.jpg","photo2.jpg","photo3.jpg","photo4.jpg"];
var textArray = ["Zardzewiała rzecz.","Uwaga na psy.","Zlew z roślinami.",
➥"Miejski kowboj."];
var urlArray = ["http://www.google.com", "http://www.onerutter.com", "http://
www.flickr.com", "http://www.facebook.com"];
```

4. Dołącz element `slide-container` do elementu kontenera strony. Jak sugeruje nazwa, element `slide-container` przechowuje wszystkie slajdy.

```
$('.container').append('<div class="slide-container" />');
```

5. Po elemencie `slide-container` wstaw listę nieuporządkowaną o nazwie `nav`. Element listy zawiera odnośniki nawigacyjne służące do kontrolowania slajdów pokazu, a ponadto dodano klasę o nazwie `clearfix`, która za listą nawigacyjną ustawia właściwość czyszczącą CSS.

```
$('.container').after('<ul id="nav" class="clearfix"></ul>');
```

6. Utwórz pętlę `for` ograniczoną przez wartość zmiennej `slideArray.length`. Ponieważ tablica `slideArray` zawiera cztery wartości, 3 to największa wartość indeksu. Wewnątrz pętli `for` utwórz zmienną `slideNum` o wartości `i+1`. Jest to konieczne, gdyż pierwszą wartością `i` będzie 0, a pierwszy obraz w tablicy zawiera w nazwie pliku wartość 1. Jeśli zmienna ta nie zostanie utworzona, nazwa pliku obrazu nie zostanie poprawnie dopasowana.

```
for(i=0; i < slideArray.length; i++){
var slideNum = i + 1;
}
```


7. Pierwsza instrukcja umieszczona wewnątrz pętli `for` dołącza pozycję listy z wartością zmiennej `slideNum`.

```
for(i=0; i < slideArray.length; i++){
var slideText = i + 1;
$('#nav').append('<li><a href="#"
rel="'+slideNum+'">'+slideNum+'</a></li>');
}
```

8. Następnie dodaj zmienną o nazwie `slideInfo` przechowującą kod HTML, który musi zostać dodany do strony w celu wyświetlenia poszczególnych slajdów. Kod HTML obejmuje wiele wierszy, tymczasem w przypadku wystąpienia jakiejkolwiek dodatkowej spacji po którymkolwiek wierszu kod JavaScript nie zadziała. Aby temu zapobiec, utwórz wiele wierszy i połącz je za pomocą operatora `+=`. Dzięki temu kod będzie bardziej przejrzysty i łatwiejszy do odczytania.

   W pierwszym wierszu jest dodawany element o nazwie `slide-image` i używana jest zmienna `slideNum` w celu wstawienia unikalnej liczby dla każdego slajdu. Drugi wiersz powoduje dodanie elementu o nazwie `slide-text`, który jest ujęty w tekst nagłówka pobieranego ze zmiennej `textArray`. W trzecim wierszu przy użyciu indeksu z pętli dodawany jest obraz przechowywany w tablicy `slideArray`.

```
for(i=0; i < slideArray.length; i++){
var slideNum = i + 1;
$('#nav').append('<li><a href="#"
rel="'+slideNum+'">'+slideNum+'</a></li>');
var slideInfo = '<div class="slide-image'+slideNum+' slides">';
```

```
slideInfo += '<div class="slide-text">'+textArray[i]+'</div>';
slideInfo += '<img src="obrazy/'+slideArray[i]+'"/></div>';
}
```

9. Po przygotowaniu kodu HTML dodaj go do elementu `slide-container`.

```
for(i=0; i < slideArray.length; i++){
var slideText = i + 1;
$('#nav').append('<li><a href="#"
rel="'+slideText+'">'+slideText+'</a></li>');
var slideInfo = '<div class="slide-image'+slideText+' slides">';
slideInfo += '<div class="slide-text
activeInfo'+[i]+'">'+textArray[i]+'</div>';
slideInfo += '<img src="obrazy/photo'+slideText+'.jpg"/></div>';
$('.slide-container').append(slideInfo);
}
```

10. Następnie dodaj trzy kolejne zmienne. Zmienna `slideTotal` przechowuje całkowitą liczbę slajdów uzyskanych z tablicy `slideArray.length` (cztery pozycje), zmienna `slideWidth` określa szerokość każdego slajdu, a wartość zmiennej `slideContainer` jest wynikiem pomnożenia szerokości przez wartość zmiennej `slideTotal`. Wartość zmiennej `slideContainer` wynosi więc 1600.

```
var slideTotal = slideArray.length;
slideWidth = 400;
var slideContainer = slideWidth * slideTotal;
```

11. Za pomocą metody `css` ustaw szerokość elementu `slide-container` będącą nową wartością przechowywaną w zmiennej `slideContainer`, którą utworzono w poprzednim kroku.

```
$(".slide-container").css({'width' : slideContainer});
```

12. Utwórz zdarzenie `click`, wybierając znacznik kotwicy, który jest potomkiem znacznika `li` pozycji listy nieuporządkowanej `#nav`.

```
$('#nav li a').bind('click', function(){
});
```

13. Wyróżnianie pozycji aktywnej polega na tym, że po kliknięciu odnośnik pozostaje aktywny, aby umożliwić zidentyfikowanie bieżącego slajdu. W celu zastosowania wyróżniania pozycji aktywnej konieczne jest dodanie dwóch instrukcji. Pierwsza z nich usuwa aktywną klasę ze wszystkich elementów. Druga instrukcja przy użyciu słowa kluczowego `this` dodaje aktywną klasę do elementu, który został kliknięty.

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
  $(this).addClass('active');
});
```

14. Następna instrukcja dodana do programu obsługi zdarzenia `click` resetuje położenie elementu `slide-text` dla kolejnego slajdu przy użyciu metody `css` (jeśli element `slide-text` nie zakończył animacji).

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
  });
});
```

15. Poprzednia instrukcja resetuje położenie elementu `slide-text`, z kolei dwie następne powodują zakończenie animacji i wyczyszczenie kolejki. Metody `stop()` i `clearQueue()` uniemożliwiają zastosowanie jakichkolwiek dalszych efektów. Ma to na celu zapewnienie, że z kolejki zostaną usunięte wszystkie pozostałe efekty. Dodanie efektów do kolejki animacji bez jej wyczyszczenia może spowodować niezamierzone rezultaty.

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
  });
$(".slide-text").stop();
$(".slide-text").clearQueue();
});
```


16. Utwórz trzy kolejne zmienne. Pierwsza o nazwie `active` przechowuje pomniejszoną o 1 wartość atrybutu znacznika `rel` bieżącej aktywnej pozycji listy `nav`. Druga zmienna, `slideNum`, przechowuje tę samą wartość co poprzednia, lecz bez odjęcia liczby 1. Trzecia zmienna o nazwie `slidePos` ma wartość równą wartości zmiennej `active` pomnożonej przez wartość zmiennej `slideWidth`.

```
$('#nav li a').bind('click', function(){
  $('#nav li a').removeClass('active');
  $(this).addClass('active');
  $(".slide-text").css({
    'top':'-100px',
    'right':'0px'
  });
  $(".slide-text").clearQueue();
  $(".slide-text").stop();

  var active = $('#nav li a.active').attr("rel") - 1;
  var slidePos = active * slideWidth;
  var slideNum = $('#nav li a.active').attr("rel");
});
```

17. Jeśli w przypadku użycia zmiennych utworzonych w poprzednim kroku wartość zmiennej `active` jest równa 2, a zmiennej `slideWidth` wynosi 400, wartość zmiennej `slidePos` jest równa 800. Jest to kluczowa zmienna służąca do przesuwania slajdów w lewo poprzez dowiązanie metody `animate` do elementu `slide-container` (widoczne w poniższym kodzie). Metodzie tej jest przekazywany również parametr `duration` o wartości 1000 i funkcja parametru `callback`.

```
$('#nav li a').bind('click', function(){
$('#nav li a').removeClass('active');
$(this).addClass('active');
$(".slide-text").css({
  'top':'-100px',
  'right':'0px'
});
$(".slide-text").clearQueue();
$(".slide-text").stop();

var active = $('#nav li a.active').attr("rel") - 1;
var slidePos = active * slideWidth;
var slideNum = $('#nav li a.active').attr("rel");

$(".slide-container").animate({
  left: -slidePos,
  },1000, function(){
});
});
```


18. W pierwszej instrukcji `animate` wewnątrz funkcji parametru `callback` zdefiniuj animację dla elementu `slide-text`. Utwórz selektor dla klasy unikalnego elementu `slide-text` i za pomocą metody `css` skonfiguruj style niezbędne do wyświetlenia elementu `slide-text` nad bieżącym obrazem.

```
$('#nav li a').bind('click', function(){
  $('#nav li a').removeClass('active');
  $(this).addClass('active');

  $(".slide-text").css({
  'top':'-100px',
  'right':'0px',
});

  $(".slide-text").stop();
  $(".slide-text").clearQueue();

  var active = $('#nav li a.active').attr("rel") - 1;
  var slidePos = active * slideWidth;
  var slideNum = $('#nav li a.active').attr("rel");
```

```
  $(".slides-container").animate({
  left: -slidePos,
  },1000, function(){
    $('.slide-text').addClass('textStrip'});
});
```

19. W ostatnim wywołaniu zwrotnym dodaj kolejną metodę `animate` w celu przesunięcia elementu `slide-text` na bieżący slajd na okres jednej sekundy, a następnie przemieszczenia tekstu poza obręb slajdu. Na rysunku 5.16 przedstawiono końcowy wynik wykonania tego skryptu w przeglądarce.

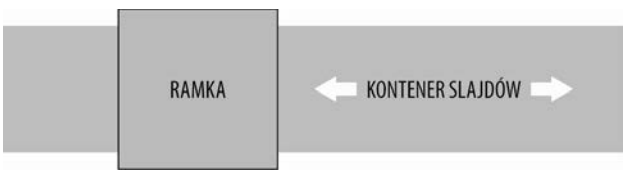
```
$('#nav li a').bind('click', function(){
  $('#nav li a').removeClass('active');
  $(this).addClass('active');

  $(".slide-text").css({
  'top':'-100px',
  'right':'0px'
  });

  $(".slide-text").stop();
  $(".slide-text").clearQueue();

  var active = $('#nav li a.active').attr("rel") - 1;
  var slidePos = active * slideWidth;
  var slideNum = $('#nav li a.active').attr("rel");

  $(".slide-container").animate({
  left: -slidePos
  },1000, function(){
    $('.slide-image'+slideNum+' .slide-text').addClass('textStrip').animate({
        top:0,
        left:slidePos,
        right:0
    }, 1000, function(){
      $('.slide-text').delay(5000).animate({
      top:-100
      }, 1000);
       });
   });
});
```

Rysunek 5.16. Końcowy wynik wykonania skryptu w przeglądarce

DODATKOWE EFEKTY PRZENIKANIA OFEROWANE PRZEZ DODATEK EASING BIBLIOTEKI JQUERY

Dodatek Easing biblioteki jQuery udostępniany przez firmę GSGD (*http://gsgd.co.uk/sandbox/jquery/easing/*) umożliwia dodanie do własnej witryny internetowej 30 różnego typu efektów przenikania. Choć metoda `animate` już sama zawiera dwa takie efekty — `swing` i `linear` — są one bardzo ograniczone. Aby mieć możliwość tworzenia bardziej realistycznych efektów przenikania takich jak efekt odbijania i efekt elastyczności, najlepiej skorzystać z tego dodatku. Przenikanie kontroluje animację, zwiększając lub zmniejszając szybkość jej pojawiania się we właściwym miejscu (często objawia się to „pstryknięciem" animacji do wybranego miejsca) i czyniąc ją bardziej realistyczną.

```
.animate(duration, easing, callback);
```

Biblioteka jQuery może być rozszerzana za pomocą dodatków. Jak już wiesz, możesz napisać własne funkcje jQuery. Możesz też utworzyć własne dodatki jQuery, aby zastosować kod ponownie w przypadku określonego zadania lub udostępnić go innym użytkownikom należącym do społeczności *open source* biblioteki. Więcej informacji na temat dodatków jQuery znajdziesz w rozdziale 11.

Oto kilka przykładowych efektów wchodzących w skład dodatku Easing:

- `easeOutBounce`,
- `easeInBounce`,
- `easeInElastic`,
- `easeInCubic`.

Efekty `easeOutBounce` i `easeInBounce` stosują połączenie przenikania i odbijania w celu uzyskania symulacji odbijania obiektu na ekranie. Efekt `easeInElastic` powoduje „pstryknięcie" elementu do wybranego miejsca na podobieństwo zachowania taśmy elastycznej. Choć efekt `easeInCubic` przypomina prosty poczciwy efekt przenikania, jest znacznie wolniejszy. Witryna internetowa dodatku Easing oferuje przykłady każdego z 30 efektów wchodzących w jego skład.

Po pobraniu dodatku Easing i dołączeniu go do strony bezpośrednio po pliku biblioteki jQuery możesz zacząć z niego korzystać. W celu demonstracji do poprzedniego przykładu dodam efekt przenikania.

Poniższy fragment kodu uzyskaj z wcześniejszego przykładu. Jeśli dla parametru efektu przenikania ustawisz efekt `easeOutBounce`, animacja użyje tego efektu zawartego w niestandardowym dodatku Easing. To takie proste!

```
$(".slide-container").animate({
  left: -slidePos,
  },'easeOutBounce',1000, function(){
    $('.slide-image'+slideNum+' .slide-text').css(
      {'display':'block',
      'position':'absolute',
      'top':'0px',
      'left':'-400px',
      'padding':'5px',
      'background':'#333333',
      'opacity':'.9',
      'color':'#ffffff',
      'width':'100%'
        }).animate({
        top:0,
        left:slidePos,
        right:0,
         },'easeOutBounce',1000, function(){
           $('.slide-text').delay(5000).animate({
           top:-100,
           }, 1000);
       });
   });
```

Skorowidz

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

Q

R

S

Ś

T

U

V

W

X

Y

Z

Ż

Stworzona w 2006 roku biblioteka jQuery miała być wybawieniem dla wielu programistów, którzy wcześniej nie mieli alternatywy — byli zmuszeni do korzystania ze skomplikowanych bibliotek języka JavaScript. I choć nie oferowała żadnych nowych funkcji, dzięki swej przejrzystej i prostej składni miała sprawić, by trudne do zrozumienia i utworzenia interfejsy API JavaScriptu stały się wreszcie szeroko dostępne. Twórcy stron nie rozczarowali się! Biblioteka jQuery spełniła pokładane w niej oczekiwania — korzystanie z niej znacząco skróciło czas pisania kodu oraz umożliwiło projektantom i programistom szybkie tworzenie komponentów interaktywnych zgodnych ze wszystkimi najważniejszymi przeglądarkami.

Jak zatem łatwo tworzyć bogate w możliwości interfejsy internetowe, integrując strukturę biblioteki jQuery z witryną internetową przy minimalnej znajomości języka JavaScript? Oto znakomita książka, napisana z myślą o wszystkich projektantach i programistach stron internetowych, którzy chcą szybko rozpocząć pracę z biblioteką jQuery. Pierwsza część książki dokładnie omawia bibliotekę jQuery, korzyści płynące z jej użycia oraz strategię progresywnego rozszerzania. Wnikliwie przedstawia też sposób instalowania i przygotowywania biblioteki jQuery do natychmiastowego użycia. W drugiej części podręcznika krok po kroku omówiono korzystanie z selektorów oraz pracę ze zdarzeniami i efektami — wszystko po to, aby zapewnić Ci solidne podwaliny pod tworzenie własnej witryny i komponentów interfejsu użytkownika. Kolejne części publikacji koncentrują się na wykorzystaniu biblioteki jQuery do usprawnienia sprawdzania poprawności formularzy, tworzeniu dodatków oraz pracy z aplikacjami mobilnymi jQuery.

W książce omówiono m.in. następujące zagadnienia:

- Podstawy biblioteki jQuery
- Żądania Ajax
- Zdarzenia i efekty
- Przetwarzanie modelu DOM z kursami poświęconymi takim czynnościom, jak tworzenie menu rozwijanego
- Ramki nakładane galerii
- Zarządzanie formularzami
- Dane tabel dynamicznych
- Efekty zdarzeń myszy

Jake Rutter jest projektantem i programistą stron internetowych z ponaddziewięcioletnim doświadczeniem w zakresie projektowania i programowania interfejsów użytkownika, pracy z kodem HTML, CSS i JavaScript obecnym w witrynach i aplikacjach internetowych utworzonych za pomocą języków PHP, Ruby on Rails, ASP i Java. W wolnym czasie prowadzi blog (http://jarerutter.com), na którym publikuje kursy dotyczące technologii jQuery, PHP, Magento, WordPress, CSS i HTML.

ISBN 978-83-246-3316-6

Cena 59,00 zł

9 788324 633166