



**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?

## » Idź do

- Spis treści
- Przykładowy rozdział
- Skorowidz

## » Katalog książek

- Katalog online
- Zamów drukowany katalog

## » Twój koszyk

- Dodaj do koszyka

## » Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

## » Czytelnia

- Fragmenty książek online

## » Kontakt

Helion SA  
ul. Kościuszki 1c  
44-100 Gliwice  
tel. 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
© Helion 1991–2011

## 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 ożywić 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 tworzeniem menu rozwijanego
- Ramki nakładane galerii
- Zarządzanie formularzami
- Dane tabel dynamicznych
- Efekty zdarzeń myszy
- Modalne okna dialogowe
- Niestandardowe dodatki biblioteki jQuery

# Spis treści

<b>CZĘŚĆ I</b>	<b>BIBLIOTEKA JQUERY I JĘZYK JAVASCRIPT</b>	
	<b>— WPROWADZENIE</b>	<b>15</b>
<b>Rozdział 1</b>	<b>Biblioteka jQuery — wprowadzenie</b>	<b>17</b>
	Poznanie bibliotek języka JavaScript	18
	Korzyści wynikające z użycia biblioteki języka JavaScript	
	w porównaniu z podejściem tradycyjnym	18
	Główni gracze w branży bibliotek	19
	Korzyści oferowane przez bibliotekę jQuery	22
<b>Rozdział 2</b>	<b>Biblioteka jQuery — pierwsze kroki</b>	<b>31</b>
	Konfigurowanie środowiska programistycznego	32
	Zastosowanie rozszerzenia Firebug w przeglądarce Firefox	33
	Pobieranie biblioteki jQuery	39
	Dołączanie biblioteki jQuery do strony internetowej	42
	Opakowanie biblioteki jQuery	44
	Uruchamianie kodu poza programem obsługi zdarzenia document.ready()	46
	Zapobieganie konfliktom z innymi bibliotekami	47
	Użycie kodu JavaScript razem z biblioteką jQuery	47
<b>CZĘŚĆ II</b>	<b>BIBLIOTEKA JQUERY — PODSTAWY</b>	<b>49</b>
<b>Rozdział 3</b>	<b>Używanie selektorów, filtrów i stylów CSS: fundamenty biblioteki jQuery</b>	<b>51</b>
	Praca z elementami modelu DOM przy użyciu selektorów jQuery	52
	Wybieranie elementów strony przy użyciu selektorów CSS	53
	Filtrowanie elementów modelu DOM przy użyciu filtrów selektorów	
	jQuery	65
	Zastosowanie definicji filtrów podstawowych	66
	Tworzenie tabel z naprzemiennie rozjaśnianymi wierszami przy użyciu filtrów even i odd	67
	Użycie stylu dla pierwszej i ostatniej pozycji listy lub kolekcji elementów	68
	Filtrowanie elementów zawierających konkretny element	70
	Filtrowanie elementów, które nie zawierają żadnego elementu ani tekstu	71
	Filtrowanie elementów zawierających tekst	72
	Wybieranie elementów w modelu DOM według ich atrybutów	74
	Wybieranie odnośników zawierających adres konkretnej witryny internetowej	74
	Wybieranie wszystkich elementów zakończonych konkretnym słowem	76
	Modyfikowanie kodu HTML i CSS przy użyciu biblioteki jQuery	77
	Dodawanie, usuwanie, klonowanie i zastępowanie elementów i treści modelu DOM	77
	Praca ze stylami CSS i biblioteką jQuery	82

<b>Rozdział 4</b>	<b>Praca ze zdarzeniami</b>	<b>85</b>
	Zdarzenia biblioteki jQuery	86
	Praca ze zdarzeniami dokumentów i okien	87
	Wykrywanie całkowitego załadowania modelu DOM przy użyciu zdarzenia ready()	87
	Wstępne ładowanie obrazów przy użyciu zdarzenia load()	88
	Wyświetlanie alertu w momencie opuszczania strony przez użytkownika	90
	Wyświetlanie pomocniczego obrazu przy użyciu zdarzenia błędu	92
	Delegowanie zdarzeń — wprowadzenie	92
	Użycie metody bind do dowiązania programu obsługi zdarzenia do elementu	93
	Użycie metody live do dowiązania programu obsługi zdarzenia do elementu	95
	Użycie metody delegate do dowiązania programu obsługi zdarzenia do elementu	96
	Przechwytywanie zdarzeń myszy	97
	Dodawanie treści do strony i usuwanie jej przez kliknięcie przycisku myszy	97
	Działanie zdarzenia double-click	100
	Tworzenie podpowiedzi wyświetlającej treść po wystąpieniu zdarzenia hover	101
	Tworzenie podstawowej funkcji dodawania do koszyka przy użyciu zdarzeń mousedown i mouseup	106
	Tworzenie efektu podmieniania dla przycisku z obrazami	109
	Przechwytywanie zdarzeń formularza	111
	Dodawanie ramki do pola formularza w momencie aktywowania tego pola przez użytkownika	112
	Wyświetlanie komunikatu po opuszczeniu przez użytkownika pola danych wejściowych	112
	Przechwytywanie zdarzeń klawiatury	113
<b>Rozdział 5</b>	<b>Zżywianie witryny internetowej przy użyciu efektów</b>	<b>117</b>
	Poznanwanie możliwości efektów jQuery	118
	Użycie efektów pokazywania i ukrywania	119
	Konfigurowanie komunikatu wyświetlanego jednokrotnie w witrynie przy użyciu metody show i informacji cookie	121
	Przełączanie się między efektami show i hide	124
	Przesuwanie elementów w górę i w dół	125
	Wyświetlanie alternatywnych opcji wyszukiwania przy użyciu metody slideToggle	126
	Znikanie elementów	128
	Tworzenie prostej galerii obrazów przy użyciu przejścia z efektem znikania	129
	Zastosowanie opóźnienia w celu utworzenia zsynchronizowanej animacji	133
	Łańcuchowe łączenie wielu efektów	135
	Tworzenie paska kanału informacyjnego przy użyciu wielu efektów	136
	Tworzenie zaawansowanych animacji	140
	Tworzenie galerii obrazów z nagłówkami tekstowymi przy użyciu zaawansowanych animacji	140
	Dodatkowe efekty przenikania oferowane przez dodatek Easing biblioteki jQuery	149

<b>CZĘŚĆ III ZASTOSOWANIE BIBLIOTEKI JQUERY W WITRYNIE INTERNETOWEJ</b>	<b>151</b>
<b>Rozdział 6 Usprawnianie nawigacji: menu, karty i harmonijki</b>	<b>153</b>
Ustawianie wszystkich odnośników na stronie w celu otwierania nowego okna	154
Ustawianie aktywnej pozycji w menu nawigacyjnym	155
Tworzenie prostego menu rozwijanego	157
Dodawanie zaawansowanych efektów do podstawowego menu rozwijanego za pomocą metody animate	163
Tworzenie menu harmonijkowego	165
Tworzenie treści z kartami	172
<b>Rozdział 7 Tworzenie interaktywnych i ekscytujących tabel</b>	<b>181</b>
Określanie stylów dla danych w tabelach przy użyciu kodu CSS	182
Dodawanie naprzemiennego kolorowania wierszy przy użyciu filtrów	183
Użycie dla wierszy prostego efektu hover	185
Użycie dla wierszy zaawansowanego efektu hover	186
Przetwarzanie danych w tabelach	187
Dodawanie komunikatu po pierwszym/ostatnim wierszu tabeli	189
Usuwanie wiersza przy użyciu selektora filtru	191
Dodawanie wiersza po wierszu na podstawie jego wartości indeksu	192
Usuwanie wiersza na podstawie jego wartości indeksu	192
Dodawanie komunikatu po wierszach z określoną treścią	192
Usuwanie wiersza na podstawie jego treści	193
Konfigurowanie paginacji tabeli przy użyciu biblioteki jQuery	194
Tworzenie zaawansowanych tabel przy użyciu dodatków biblioteki jQuery	200
Sortowanie wierszy przy użyciu dodatku tablesorter	200
Zmiana domyślnej kolejności sortowania	203
Tworzenie atrakcyjnych wykresów z danymi tabelarycznymi przy użyciu dodatku Visualize	204
Tworzenie wykresu słupkowego	205
<b>Rozdział 8 Tworzenie zaawansowanych formularzy przy użyciu biblioteki jQuery</b>	<b>209</b>
Aktywowanie pola danych wejściowych po załadowaniu strony	210
Wyłączanie i włączanie elementów formularza	211
Wyróżnianie bieżących pól formularza	212
Określanie tekstu domyślnego pól danych wejściowych	214
Ograniczanie liczby znaków w polach danych wejściowych	217
Tworzenie odnośnika pola wyboru Zaznacz wszystkie	219
Uzyskiwanie wartości pola danych wejściowych	221
Pobieranie wartości opcji wyboru	223
Dodawanie do formularza prostego mechanizmu sprawdzania poprawności adresu e-mail	224
Kopiowanie zawartości jednego pola do drugiego	229

Rozszerzanie formularzy przy użyciu dodatków	232
Zastosowanie dodatku qTip w witrynie internetowej	233
Tworzenie prostego pola formularza dodatku qTip przy użyciu atrybutu title	234
Użycie dodatku Validate biblioteki jQuery	
do sprawdzania poprawności formularzy	235
Użycie prostego sprawdzania poprawności dla formularza kontaktowego	237
Dodawanie zaawansowanych reguł sprawdzania poprawności i komunikatów do formularza kontaktowego	240

## **CZĘŚĆ IV POZNAWANIE ZAAWANSOWANYCH MOŻLIWOŚCI BIBLIOTEKI JQUERY** **245**

<b>Rozdział 9 Praca z danymi dynamicznymi i technologią Ajax</b>	<b>247</b>
Poznanie technologii Ajax	248
Ładowanie treści dynamicznej strony internetowej	250
Ładowanie całej treści	250
Obsługa błędów w przypadku braku ładowanej treści	252
Ładowanie sekcji treści	254
Wysyłanie formularzy przy użyciu żądań GET i POST	256
Użycie żądania POST do wysyłania formularzy kontaktowych bez ponownego ładowania strony	258
Praca z danymi XML	262
Analiza składniowa wewnętrznych danych XML i tworzenie kodu HTML	264
Praca z danymi JSON	267
Pobieranie wewnętrznych danych JSON i tworzenie kodu HTML	269
Tworzenie widżetu użytkownika w witrynie Delicious z zastosowaniem odbierania danych JSONP z żądań API	271
Tworzenie widżetu najważniejszych przeglądów witryny Yelp przy użyciu kodu JSONP za pośrednictwem interfejsu API witryny Yelp	278
Proces uzyskiwania klucza interfejsu API witryny Yelp	279
Użycie interfejsu API witryny Yelp do wyświetlania przeglądów na podstawie numerów telefonów	282
<b>Rozdział 10 Tworzenie i używanie dodatków biblioteki jQuery</b>	<b>289</b>
Czym są dodatki?	290
Zastosowanie dodatku biblioteki jQuery we własnej witrynie internetowej	291
Zastosowanie biblioteki jQuery UI we własnej witrynie internetowej	292
Pobieranie biblioteki jQuery UI	293
Dodawanie biblioteki jQuery UI do własnej witryny	293
Zasady działania widżetów biblioteki jQuery UI	294
Dostosowywanie projektu biblioteki jQuery UI	295
Tworzenie kompozycji interfejsu użytkownika za pomocą aplikacji ThemeRoller	297
Korzystanie z kompozycji biblioteki jQuery UI	299
Uwzględnienie funkcji biblioteki jQuery UI we własnej witrynie internetowej	300

Wykorzystanie popularnych dodatków biblioteki jQuery	308
we własnej witrynie internetowej	308
Używanie biblioteki jQuery Tools	309
Fancybox	313
Tworzenie pierwszego własnego dodatku biblioteki jQuery	315
Przygotowywanie planu dodatku	316
Struktura dodatku	316
Ustawianie opcji dodatku	317
Tworzenie dodatku	318
Dystrybuowanie dodatku biblioteki jQuery	324
Przygotowanie pakietu dodatku biblioteki jQuery do dystrybucji	325
Zamieszczanie dodatku w witrynach internetowych	325
<b>Rozdział 11 Programowanie przy użyciu biblioteki jQuery dla mobilnych aplikacji internetowych</b>	<b>327</b>
Tworzenie mobilnej aplikacji internetowej przy użyciu biblioteki jQuery	328
Przeglądarki mobilne	329
CSS3	330
HTML5	331
Przygotowanie się do rozpoczęcia projektowania mobilnej aplikacji internetowej	332
Korzystanie z mobilnej przeglądarki Apple iPhone Safari	333
Korzystanie z przeglądarki Google Android	334
Wyświetlanie treści na podstawie tego, z jakiego smartfonu korzysta użytkownik	336
Tworzenie mobilnych witryn i aplikacji internetowych za pomocą biblioteki jQuery	337
Ogólny przegląd dodatku jQuery Mobile	337
Mobilne struktury programistyczne	337
Korzystanie ze struktury Appcelerator Titanium Mobile	338
Korzystanie z dodatku jQTouch	339
<b>Rozdział 12 Wyszukiwanie zasobów dotyczących biblioteki jQuery</b>	<b>341</b>
Obserwowany wzrost popularności biblioteki jQuery	342
Korzystanie z witryny internetowej biblioteki jQuery	343
Praca z dokumentacją interfejsu API	344
Znajdowanie kursów dotyczących biblioteki jQuery	345
Udział w spotkaniu lub konferencji dotyczącej biblioteki jQuery	345
Umieszczanie błędów w sekcji Bug Tracker	348
Uczestniczenie w forum poświęconym bibliotece jQuery	348
Inne zasoby dotyczące projektowania i programowania witryn internetowych	349
<b>Skorowidz</b>	<b>351</b>

## 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.



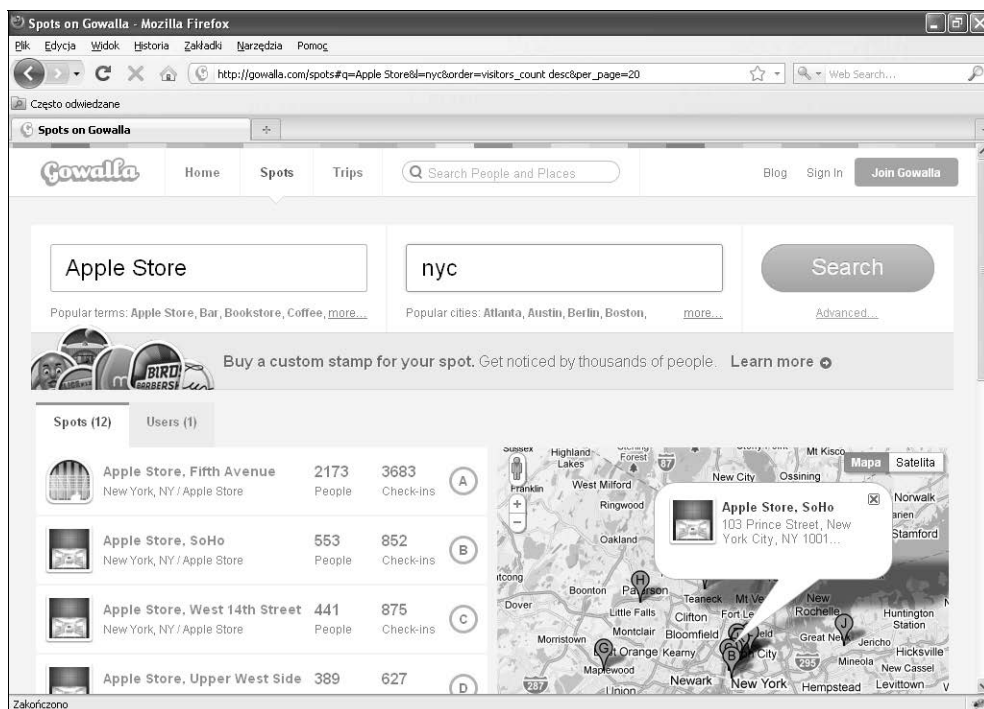
## POZNAWANIE MOŻLIWOŚCI EFEKTÓW JQUERY

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.

118



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

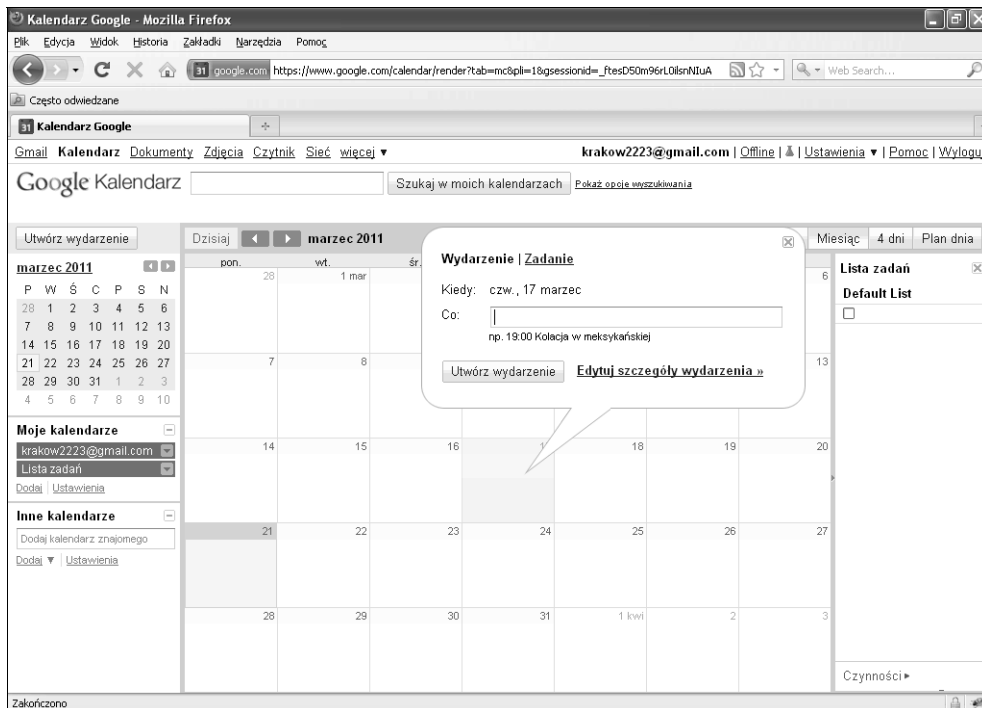
## UŻYCIE EFEKTÓW POKAZYWANIA I UKRYWANIA

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 powiązaniem zdarzeniem `click`. Po kliknięciu odnośnika zostanie pokazany element z klasą `recipe`. Jest to efekt `show` w swojej najprostszej postaci.



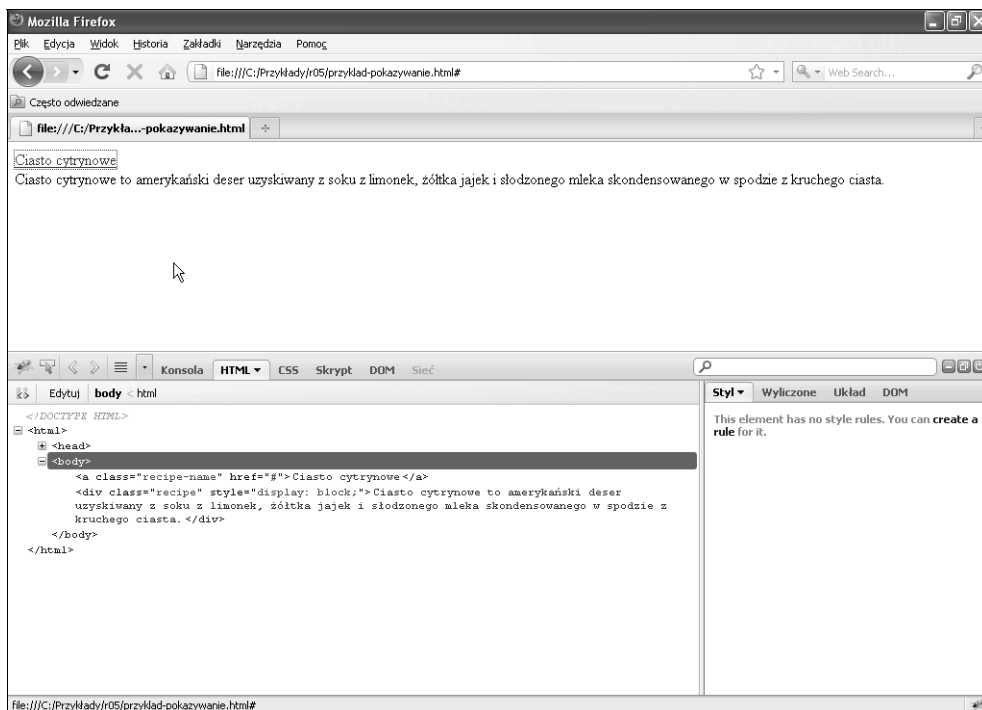
Rysunek 5.2. Aplikacja Google Kalendarz używa efektów show i hide do wyświetlenia okienka wydarzenia (reprodukcja z 2010 © Google)

```
<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`

Żałuję 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 zwrótną, 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`.

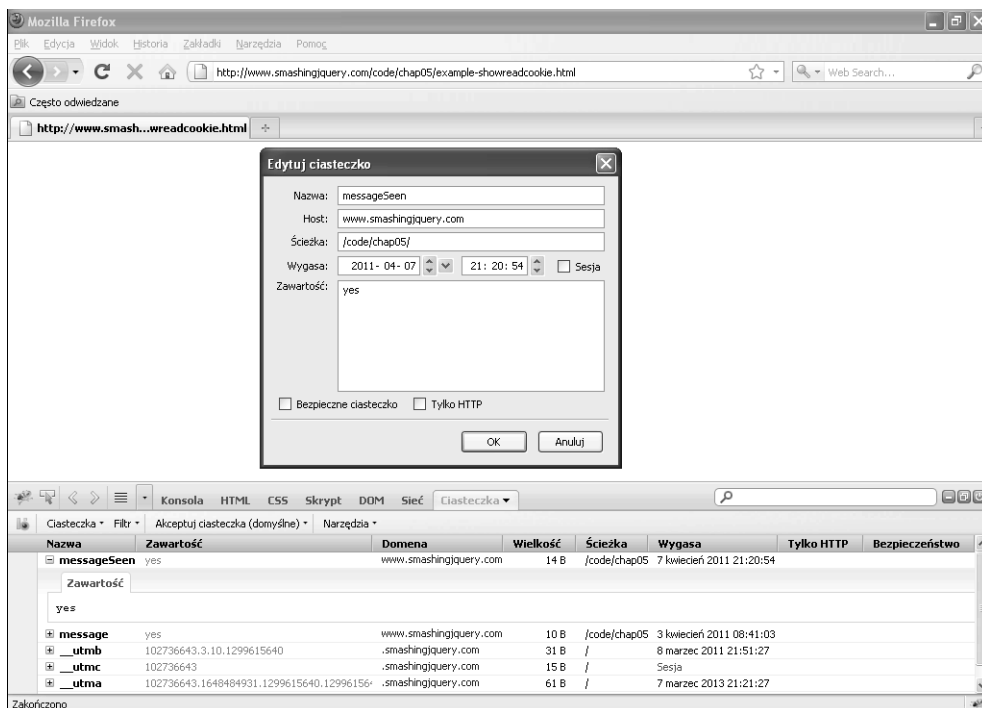
```
$('.hide').bind('click', function(){
  $('#message').hide(hideMessage);
});
```

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

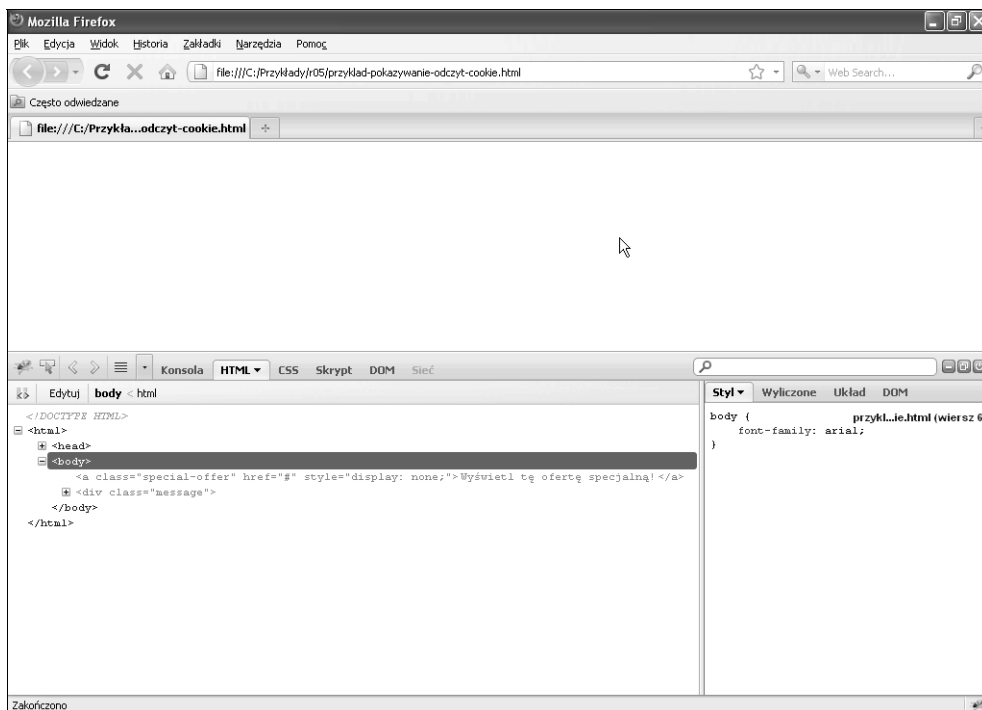
Na tym etapie powinno być możliwe wyświetlenie i ukrycie komunikatu oraz ustawienie informacji *cookie*. A co z użytkownikami, którzy już wyświetlili komunikat? Jeśli jutro powrócą do witryny, nie mogą ponownie ujrzeć tej informacji. Aby to osiągnąć, utwórz kolejną funkcję, która będzie uruchamiana przy 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

124

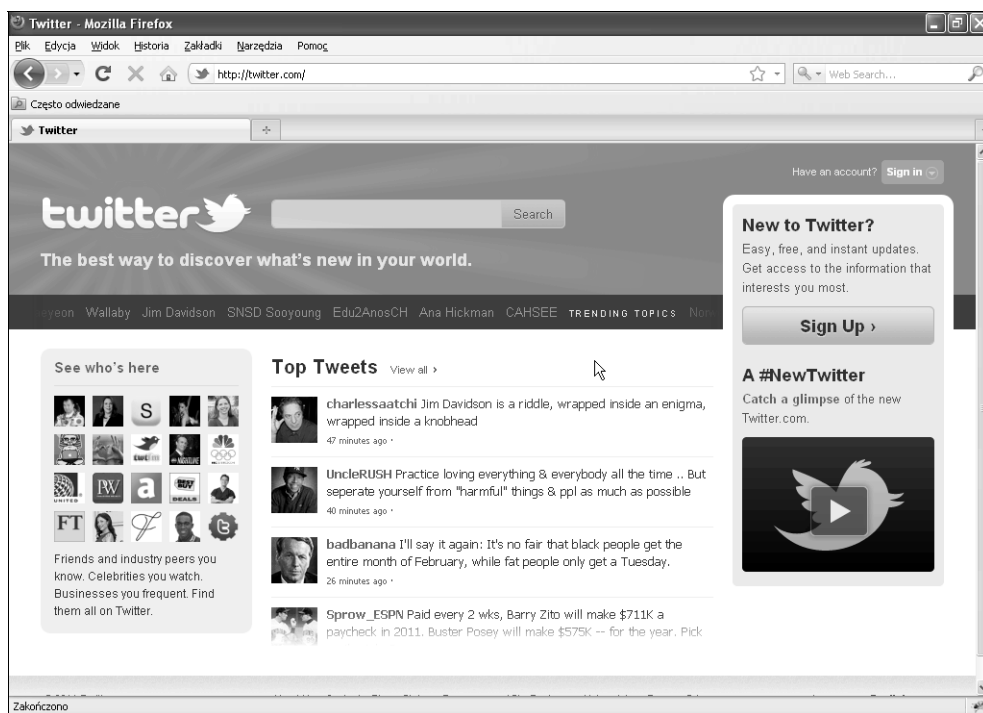
### PRZEŁĄCZANIE SIĘ MIĘDZY EFEKTAMI SHOW I HIDE

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 spódzie 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 częściej jest stosowane przesuwanie 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](http://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 SLIDETOOGLE

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 rozwinęty. Jest to użyteczna funkcja, ponieważ umożliwia rozszerzenie zakresu wyszukiwania bez opuszczania bieżącej strony. Funkcję tę z łatwością możesz dodać przy użyciu metody `slideDown` biblioteki jQuery.

126



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`. Jediną 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 `slideDown`. 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).

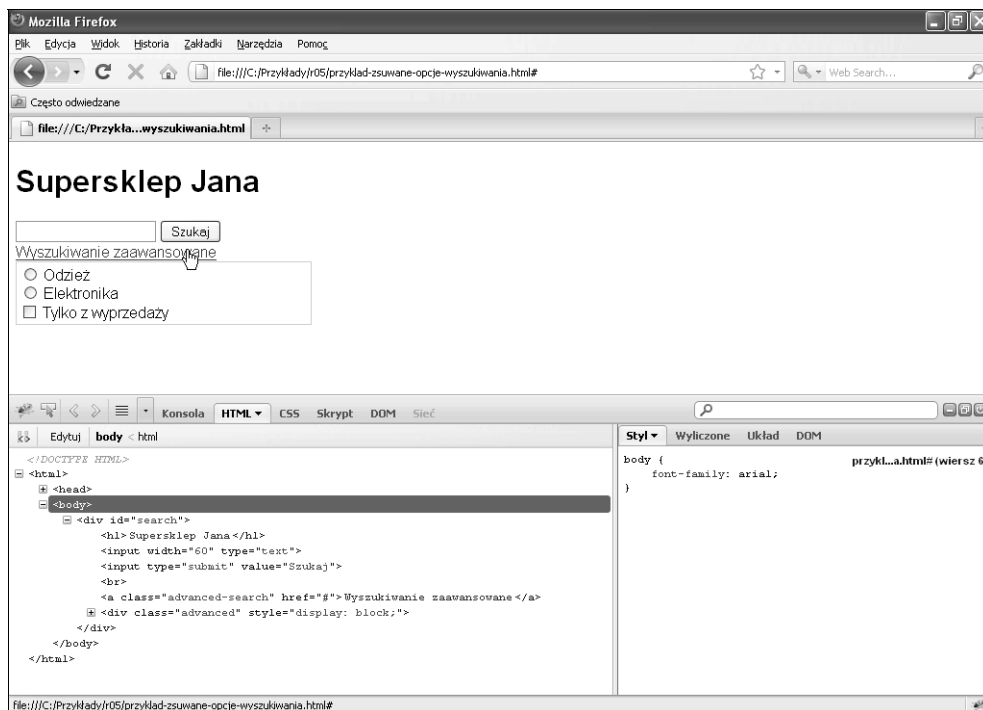
1. Utwórz kod HTML dla pola wyszukiwania i opcji wyszukiwania zaawansowanego.

```
<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 JavaScript 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 zmniejszenie. Efekt `fadeOut` umożliwia określenie poziomu nieprzezroczystości, który ma zostać osiągnięty przez wybrany element.

```
$(selector).fadeOut(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 = [
  "anse1_adams1.jpg",
  "anse1_adams2.jpg",
  "anse1_adams3.jpg",
  "anse1_adams4.jpg",
  "anse1_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/anse1_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('</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.

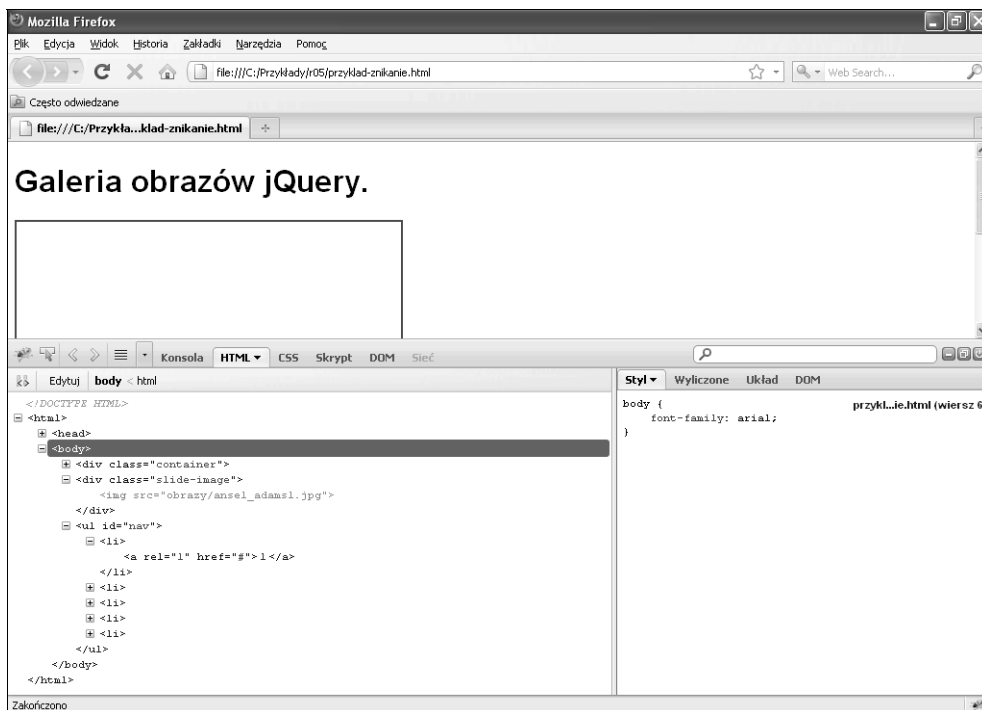
```
for(i=0; i < slideLength; i++){
}
```

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` umożliwi wybranie wstawianego obrazu. Na rysunku 5.10 przedstawiono pętlę przetwarzaną w przeglądarce z wynikiem w postaci kodu HTML, którego wyświetlenie umożliwi 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

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

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

- 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 wyzwolenia zdarzenia `click`, a nie dołączania.

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

- 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('');
});
```

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 klikniętego.

```
$('#nav li a').bind('click', function(){
  var numSlide = $(this).attr('rel');
  $('.slide-image').html('');
  $('#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('');
  $('.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 ujżenia komunikatu.

```
<a href="#" class="show-tip">Dowiedz się czegoś o brzoskwiach</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 wspomniano 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').fadeOut(900);
$('.tool-tip').delay(10000);
$('.tool-tip').fadeIn(900);
```

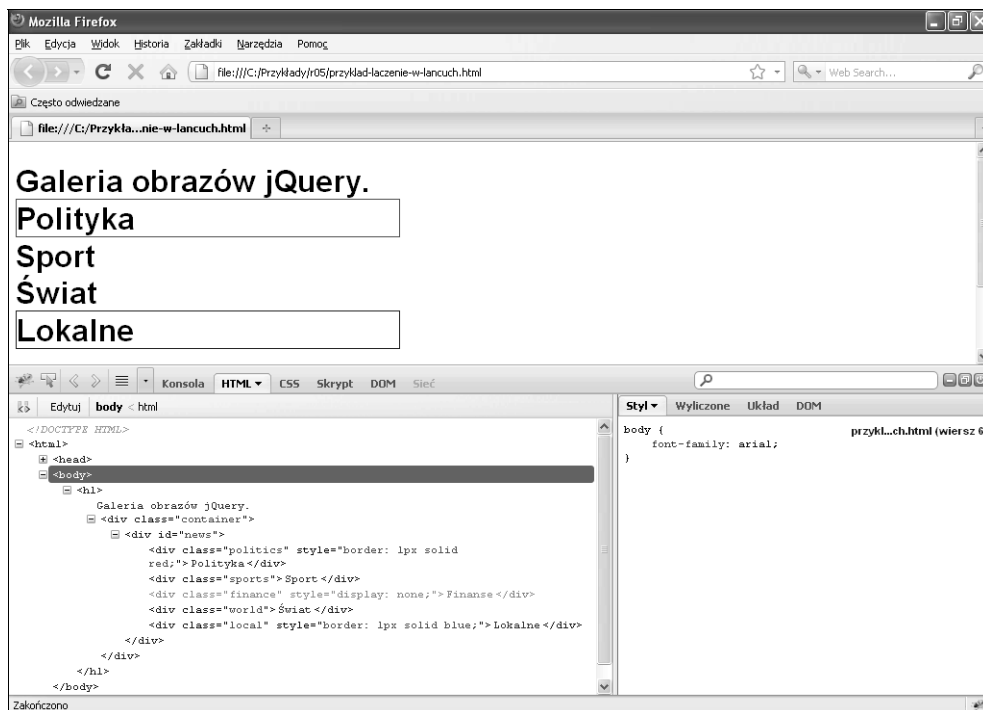
Choć powyższa sekwencja instrukcji jQuery daje ten sam wynik co instrukcja z zastosowanym łączeniem w łańcuch, pierwsze rozwiązanie pozwala zaoszczędzić miejsce i zapewnia większą przejrzystość kodu. Rozważmy scenariusz, w którym musisz wybrać trzy różne elementy `li`, używając ich odpowiednich nazw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



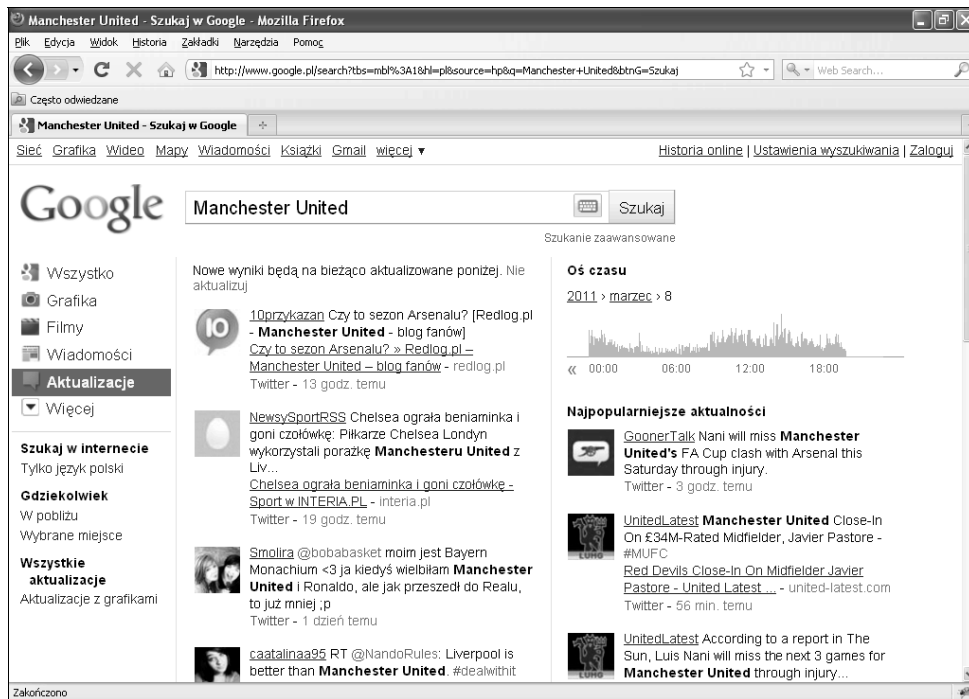
Rysunek 5.11. Końcowy wynik połączenia w łańcuch wielu metod z poprzedniego przykładowego kodu

```
$('#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 kupić 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"
];
```

- 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;
```

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

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

- 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>');
}
```

- 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();
}
```

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

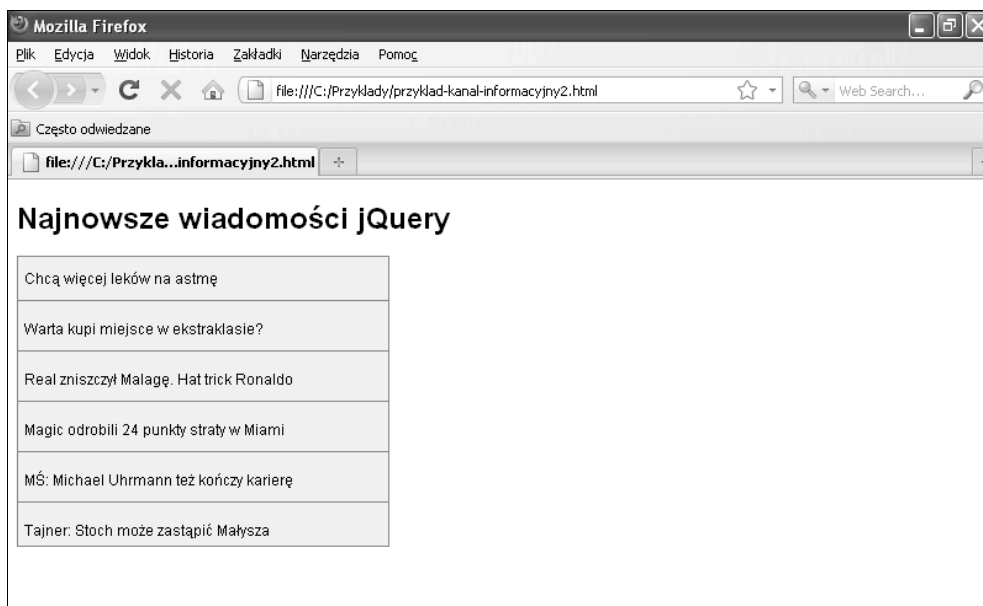
```
function slideHeadline() {
$('#news-feed li:last').clone().prependTo('#news-feed').css('display','none');
$('#news-feed li:first').slideDown();
}
```

- 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
<code>opacity</code>	0.5
<code>top</code>	10 px
<code>height</code>	100 px
<code>width</code>	200 px
<code>margin</code>	10 px
<code>padding</code>	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.

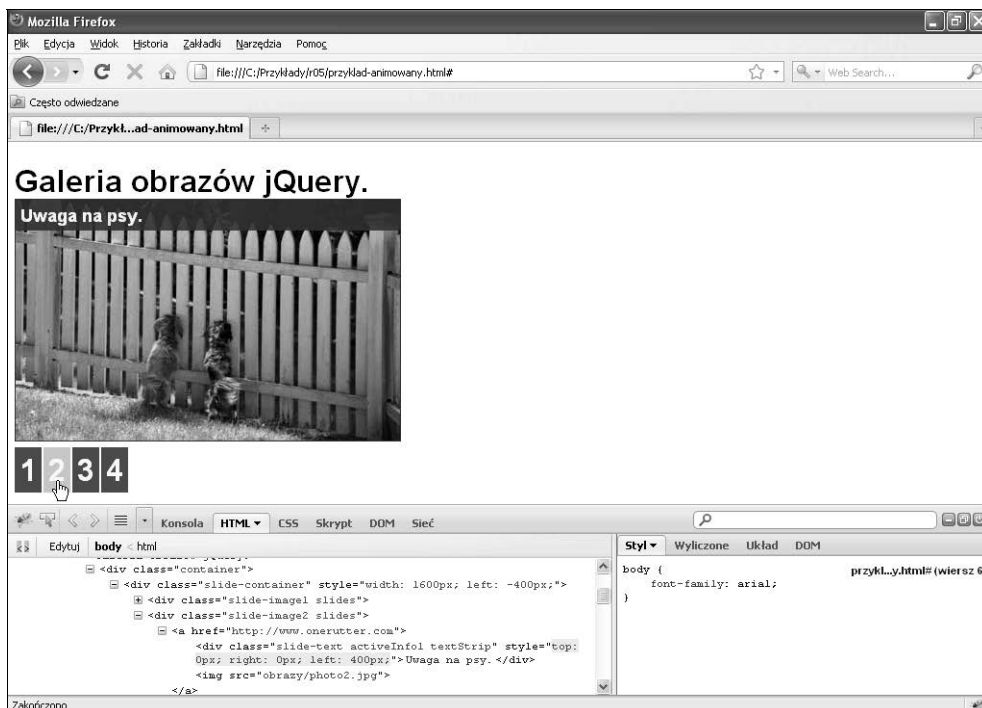
W 2006 r. utworzyłem swój pierwszy animowany pokaz slajdów oparty na technologii Flash 8 XML. Była to naprawdę zgrabna i niewielka aplikacja. Gdy sformatowany plik XML był przekazywany do pliku Flash, przetwarzał on kod XML, dodawał slajdy i stosował dla nich animację, umożliwiając użytkownikowi kliknięcie 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 czyszcząca 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 jakiegokolwiek 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 += '</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 += '</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 zwrótnym 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');

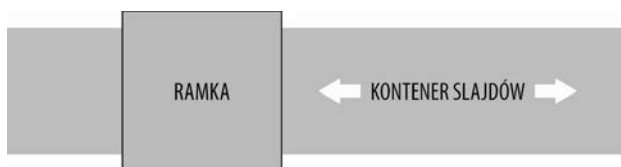
  $(".slide-text").css({
    'top':'-100px',
    'right':'0px'
  });

  $(".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 pocziwy 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(){
  $('.'+slideImage+'.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(){
      $('.'+slideText+'.slide-text').delay(5000).animate({
        top: -100,
      }, 1000);
    });
});
```

# Skorowidz

- #, 57
- #copy-fields (element danych wejściowych), 231
- #sidebar, 57
- \$, 44, 45, 47, 52, 76
- \*, 54, 74
- .addClass (metoda), 83
- .after (metoda), 189
- .animate (metoda), 149, 163
- .append (metoda), 79
- .attr (metoda), 154
- .before (metoda), 80
- .bind (metoda), 93, 97, 132
- .blur (zdarzenie), 215
- .clone (metoda), 82
- .container (właściwość), 142
- .content (klasa), 70
- .content-container (element), 174
- .css (metoda), 53, 83
- .delay (metoda), 134
- .delegate (metoda), 96
- .error (klasa), 71
- .error (zdarzenie), 92
- .extend (metoda), 317
- .fadeIn (efekt), 119, 129
- .fadeOut (efekt), 119
- .fadeTo (efekt), 119
- .focus (zdarzenie), 210, 212, 215
- .hasClass (metoda), 83
- .hide (efekt), 119
- .hover (zdarzenie), 110, 162
- .html (metoda), 78
- .last (klasa), 68
- .live (metoda), 95
- .load (metoda), 250, 253
- .prepend (metoda), 79
- .ready (zdarzenie), 45, 87
- .remove (metoda), 81, 191
- .removeClass (metoda), 83, 84
- .show (efekt), 119, 122
- .slideDown (efekt), 119
- .slideToggle (metoda), 127
- .slideToggle (efekt), 119
- .slideUp (efekt), 119
- .text (metoda), 79
- .toggle (efekt), 119, 124
- .toggleClass (metoda), 84
- .toggleClass(metoda), 83
- .validate (metoda), 242
- :animated (filtr), 66
- :contains (filtr), 72, 192
- :contains ('to jest mój tekst') (filtr), 66
- :empty (filtr), 66, 71
- :eq (filtr), 192
- :eq(index) (filtr), 66
- :even (filtr), 66, 67
- :filtr, 65
- :first (filtr), 68, 168, 189
- :first-child (filtr), 66
- :gt(index) (filtr), 66
- :has(p) (filtr), 66
- :header (filtr), 66
- :hidden (filtr), 66
- :last (filtr), 68, 190
- :last-child (filtr), 66
- :lt(index) (filtr), 66
- :not (filtr), 66
- :nth-child(filtr), 66
- :odd (filtr), 66, 67
- :only-child (filtr), 66
- :parent (filtr), 66
- :visible (filtr), 66
- „pstryknięcie” elementu do wybranego miejsca, 149
- <a> (znacznik), 74
- >, 61
- 200 (komunikat), 253
- 301 (komunikat), 253
- 302 (komunikat), 253
- 400 (komunikat), 253
- 401 (komunikat), 253
- 403 (komunikat), 253
- 404 (komunikat), 253
- 500 (komunikat), 253

**A**

Accordion (komponent), 302  
 Accordion (widżet), 293  
 accordion-content (klasa), 166  
 accordion-header (klasa), 166  
 adres  
   e-mail (poprawność), 224, 226, 240  
   URL (poprawność), 240  
 Ajax, 248  
   biblioteka oferująca funkcjonalność technologii, 20  
   Facebook, 249  
   Gmail, 249  
   technologia, 248  
 Ajax (żądanie)  
   obsługa wielu przeglądarek  
     JavaScript, 28  
     jQuery, 28  
 aktualnie używane pole formularza, 212  
 aktywacja pola  
   formularza, 210  
   tekstowego, 111  
   za pomocą klawisza Tab pola tekstowego  
     zawartego wewnątrz elementu lub  
     zaznaczenia tego pola, 111  
 aktywna pozycja menu, 155, 157  
 alias  
   biblioteki jQuery, 52  
   własny, 47  
 analiza składniowa treści HTML na stronie, 264  
 Android, 334  
   emulator, 333  
 animacja, 133  
   niestandardowa, 140  
   obrazu, 141  
 animate, 177  
 anonimowa funkcja, 94  
 anulowanie zaznaczenia wszystkich pól wyboru  
   na stronie internetowej, 219  
 Apache (serwer), 32, 337  
 API, 17  
   klucz interfejsu, 279  
   Phone Search, 284  
   Review Search, 283  
   Yelp, 279  
 aplikacja  
   macierzysta, 329  
   mobilna, 45  
 Appcelerator Titanium Mobile, 338

append, 82  
 Apple  
   iPhone/iPad, 328  
   Safari, 333  
     emulator, 332  
   Safari 5, 329  
 Application Programming Interface (API), 17  
 arkusz stylów CSS  
   resetujący, 160  
 Asynchronous JavaScript and XML, 248  
 atrybuty  
   składnia, 74  
   znacznika, 74  
 Autocomplete (komponent), 293, 304, 306  
 automatyczne uzupełnianie, 304

**B**

background (właściwość CSS), 183  
 Bad request, 253  
 Bank of America (witryna z jQuery ), 24  
 BBC (witryna z jQuery ), 24  
 Berkeley Software Distribution (BSD), 20  
 biblioteka interfejsu użytkownika, 292, 309  
 biblioteka  
   JavaScript  
     Dojo, 21  
     główne, 19  
     MooTools, 21  
     Prototype, 20  
     Scriptaculous, 20  
     YUI, 20  
     zalety, 18  
   jQuery  
     dołączanie, 42, 43  
     metoda, 53  
     opakowanie, 44  
     wersje, 41  
   jQuery Tools, 309  
   open source, 19  
 blur (zdarzenie), 111, 112  
 błąd  
   serwera, 253  
   żądania HTTP (zdarzenie), 87  
 Border  
   Image, 331  
   Radius, 331  
 Box Shadow, 331  
 brak autoryzacji, 253  
 BSD, 20

Bug Tracker, 348  
 Button (widżet), 293

## C

Cascading Style Sheets 3, 330  
 change (zdarzenie formularza), 111  
 check (klasa), 219  
 cienie na elementach tekstowych, 331  
 class (znacznik), 74  
 click, 97
 

- Dojo, 21
- jQuery, 21
- MooTools, 21
- Prototype, 20
- YUI, 20

 click (zdarzenie), 98, 226  
 Coda (edytor kodu), 32  
 CSS
 

- arkusz resetujący, 160
- selektor, 53

 CSS3, 66, 330, 334
 

- funkcje rozszerzone, 28
- lista nowych funkcji i opcji, 331

 cssAsc, 200  
 cssDesc, 200  
 cssHeader, 200  
 cyfry poprawność, 240  
 czasu ładowania witryny skracanie, 40  
 czcionki osadzone (CSS3), 28

## D

data (poprawność) 240  
 Datepicker (komponent), 293, 306, 308  
 dblclick (zdarzenie), 97  
 debugger JavaScript, 33, 38, 39  
 delegowanie zdarzeń, 92  
 Delicious (witryna), 273  
 Dell (witryna z jQuery), 24  
 dezaktywowania pola lub obszaru tekstowego, 111  
 Dialog (widżet), 293  
 div, 57  
 doctype, 42  
 Document Object Model (DOM), 18  
 document.ready, 44, 46  
 dodanie
 

- ramki do każdego elementu, 54
- wielu metod do tej samej instrukcji, 135

dodatek, 290, 308
 

- biblioteki jQuery (własny), 315
- dokumentacja, 324
- dołączenie do własnej witryny internetowej, 291
- dystrybuowanie, 324, 325
- działanie w przeglądarkach, 324
- licencja, 324
- minifikacja, 325
- plan, 316
- sprawdzenie w witrynie internetowej, 323
- struktura, 316
- tworzenie, 318
- ustawienia opcji, 317

 dodawanie
 

- danych do tabeli, 187
- klasy, 83
- treści, 193
- wiersza na podstawie indeksu, 192

 Dojo, 21  
 dołączanie
 

- biblioteki jQuery, 42, 43
- treści z oddzielnego pliku HTML, 250

 DOM, 18, 52
 

- rozszerzenie, 18

 DOM scripting, 95  
 domyślna kolejność sortowania, 203  
 domyślny tekst, 214  
 double-click (zdarzenie), 100  
 dowiązanie programu obsługi zdarzenia
 

- .bind, 97
- .delegate, 96
- .live, 95

 Draggable (interakcja), 293  
 Dreamweaver (edytor kodu), 32  
 Droppable (interakcja), 293  
 dwukrotne kliknięcie przycisku myszy, 97  
 dymek informacyjny, 101  
 dynamiczne,
 

- dodawanie treści do strony, 189
- karty, 300
- treści, 251

## E

Easing (dodatek biblioteki jQuery), 149  
 EditPlus (edytor kodu), 32  
 edycja
 

- kodu HTML, 37
- zawartości pola, 111

 edytor kodu, 32

efekty, 119  
 podmieniania, 109  
 przenikania, 149  
 tekstowe (CSS3), 28  
 znikania/pojawiania się elementów  
 witryny, 128

element

- HTML
  - klonowanie, 82
  - usuwanie, 81
- modelu DOM, 52
- o wielu klasach, 60
- witryny
  - pojawianie się, 128
  - znikanie, 128
- wyboru daty, 307

eliminacja migotania podczas ładowania  
 obrazu, 167

e-mail (sprawdzanie poprawności), 224, 226,  
 238, 240

emulator

- Android Chrome, 333
- Apple Safari, 332

error (zdarzenie), 87

eXtensible Markup Language, 262

**F**

Facebook

- Ajax, 249
- interfejs, 118

Fancybox (dodatek), 313, 314

filtrowanie, 65

- adres witryny internetowej, 74
- danych zawartych w tabeli, 187
- elementy
- puste, 71
  - zakończone konkretnym łańcuchem  
 tekstowym, 76
  - zawierające konkretny element, 70
- pozycja
  - ostatnia, 68
  - pierwsza, 68
- tekst zawarty w elemencie, 72

Firebug, 33, 37

- instalowanie, 34
- konsola, 37
- włączanie, 35

Firefox, 33

focus (zdarzenie formularza), 111, 112

focusin (zdarzenia formularza), 111

focusout (zdarzenia formularza), 111

font-family, 55

Forbidden, 253

format JSON, 267

formularz

- aktywacja pola, 210
- kontaktowy, 237, 241
  - wysyłanie, 258
- sprawdzanie poprawności, 113, 235
- włączanie elementów, 211
- wyłączanie elementów, 211
- wyróżnienie aktualnie używanego pola, 212

funkcja anonimowa, 94

**G**

galeria

- animowanych obrazów, 141
- obrazów, 129

GET (żądanie), 256

getElementById, 57

getElementsByClassName, 58

getElementsByName, 55

Gmail Ajax, 249

Google (witryna z jQuery), 24

Google Android, 328, 329

- emulator, 333

Google CDN, 293

grupa docelowa aplikacji, 328

**H**

harmonijka skrypt, 171

harmonijka (widżet), 302

harmonijkowe menu, 165

has (filtr), 70

height, 164

Hewitt Joe, 33

hide, 119

hiperłącza otwierane w nowym oknie, 154

hover, 177, 185, 187

href (znacznik), 74

hreflang (znacznik), 74

HTML

- analiza składniowa treści na stronie, 264
- edycja kodu, 37
- inspekcja kodu, 37
- klonowanie elementów, 82
- usuwanie elementów, 81

HTML5, 331, 334

**I**

- id (znacznik), 74
- identyczności elementów sprawdzanie, 240
- include, 43
- inspekcja kodu HTML, 37
- instrukcji selektora jQuery, 53
- interakcja, 292
- interaktywny wykres, 204
- interfejs API, 17
  - klucz, 279
- Internet Explorer 6.0+
  - zgodność, 27

**J**

- JavaScript
  - debugger, 33, 38, 39
  - testowanie kodu, 38
  - zalety biblioteki, 18
- JavaScript Object Notation, 267
- jednokrotne lub wielokrotne naciśnięcie
  - klawisza, 113
- język XML, 262
- jQTouch (dodatek), 339
- jQuery, 52
  - błędy, 348
  - cechy podstawowe, 23
  - Conference, 347
  - dokumentacja, 344
  - dołączanie
    - biblioteki, 42, 43
    - do strony, 24
  - forum, 348
  - historia, 22
  - korzyści, 22
  - kursy, 345
  - metoda, 53
  - Mobile (dodatek), 337
  - opakowanie, 44
  - pobieranie, 39
  - popularność, 342
  - powody utworzenia, 17
  - spotkania i konferencje, 345
  - testowanie kodu, 38
  - Tools, 309
  - udostępniana przez Google, 43
  - UI, 292, 293
  - wersje biblioteki, 41
  - witryna internetowa, 343
  - zastosowanie, 23

- jquery.plugins.js, 291
- JSON, 267
  - kod, 267
  - ładowanie danych za pośrednictwem żądania
    - Ajax, 269
  - narzędzie sprawdzające poprawność kodu, 268
- JSON with Padding, 275
- JSONLint, 268
- JSONP, 275

**K**

- kanal informacyjny, 275
  - RSS, 136
- karty, 311
  - dynamiczne, 300
  - nawigacja, 172
  - rozmieszczenie na stronie, 175
- keydown (zdarzenia klawiatury), 113
- keypress (zdarzenia klawiatury), 113, 217
- keyup (zdarzenia klawiatury), 113
- klasa
  - book, 60
  - dodawanie, 83
  - inactive, 60
  - przełączanie, 84
  - usuwanie, 84
- kliknięcie
  - i zwolnienie przycisku myszy, 97
  - przycisku myszy, 97
- klonowanie elementów HTML, 82
- klucz interfejsu API, 279
- kod
  - dodanie na
    - końcu wybranego elementu, 79
    - początku wybranego elementu, 79
  - działanie we wszystkich przeglądarkach, 42
  - html pobieranie, 78
  - JSON, 267
  - JSONP, 275
  - kolejność ładowania, 42
  - przyspieszenie pisania, 18
  - resetujący CSS, 167
  - zwięzłość, 27
- kody odpowiedzi serwera, 253
- kolejność
  - ładowania kodu, 42
  - sortowania, 203
- kolorowanie wierszy tabeli, 183

kolory  
 CSS3, 28  
 inne niż RGB, 331  
 kompozycja niestandardowa, 299  
 komunikat o błędzie, 38  
 konfigurowanie środowiska programistycznego, 32  
 konflikty z bibliotekami JavaScript, 47  
 kopiowanie zawartości pól formularza, 229  
 koszyk zakupów, 107

**L**

licencja  
 BSD, 20  
 liczba  
 dziesiętna (poprawność), 240  
 liczba znaków w polu danych wejściowych  
 (ograniczenie), 217  
 lightbox, 290  
 literał funkcji, 94  
 load (zdarzenie), 87  
 lokalne środowisko programistyczne, 32

**Ł**

ładowanie  
 niektórych sekcji treści z zewnętrznego pliku  
 HTML, 254  
 obrazów  
 eliminacja migotania, 167  
 wstępne, 88  
 łączenie metod w łańcuch, 27

**M**

Major League Baseball (witryna z jQuery ), 24  
 MAMP (środowisko programistyczne), 32  
 mechanizm selektorów Sizzle, 52  
 menu  
 harmonijkowe, 165, 302  
 nawigacyjne (pozycja aktywna), 155  
 pozycja aktywna, 157  
 rozwijane, 157  
 zmiana nieprzezroczystości, 163  
 metoda jQuery, 53  
 migotanie podczas ładowania obrazu, 167  
 minifikacja, 325  
 mobilna  
 aplikacja internetowa, 328  
 projektowanie, 332

przeglądarka, 329  
 struktura programistyczna, 337  
 Mod\_rewrite, 336  
 model DOM, 18, 52  
 rozszerzenie, 18  
 MooTools, 21  
 mousedown, 97, 107  
 mouseenter, 97, 104, 176, 186, 187  
 mouseleave, 97, 104, 176, 186, 187  
 mousemove, 97  
 mouseout, 97, 104  
 mouseover, 97, 104  
 mouseup, 97, 107  
 Moved  
 permanently, 253  
 temporarily, 253  
 Mozilla Firefox 2+  
 zgodność, 27  
 Mozilla Firefox Mobile, 329  
 Multiple Backgrounds, 331  
 mysz, 97

**N**

naciśnięcie klawisza, 113  
 nadrzędno-podrzędny selektor, 61  
 naprzemienne  
 kolorowanie wierszy tabeli, 183  
 rozjaśnianie wierszy, 67  
 narożniki zaokrąglone, 28, 331  
 nawigacja (karty), 172  
 NBC (witryna z jQuery ), 24  
 Netflix (witryna z jQuery ), 24  
 nie znaleziono, 253  
 niedozwolone, 253  
 niepoprawne żądanie, 253  
 nieprzezroczystość elementu, 119  
 noConflict, 47  
 Not found, 253  
 numeru karty kredytowej poprawność, 240

**O**

obrazy  
 animowane, 141  
 galeria, 129  
 ładowanie wstępne, 88  
 tła zaawansowane (CSS3), 28  
 znikanie, 129

obsługa  
 błędów, 252  
 wielu przeglądarek, 28  
 odebranie błędu z żądania HTTP (zdarzenie), 87  
 odpowiedzi serwera (kody), 253  
 ograniczenie liczby znaków w polu danych wejściowych, 217  
 okna wyskakujące, 29  
 onMouseDown, 107  
 onMouseup, 107  
 opacity, 164  
 opakowanie biblioteki jQuery, 44, 45  
 opcja wyboru pobierania wartości, 223  
 open source, 19  
 Opera 10 Mobile, 329  
 Opera 9.0+ (zgodność), 27  
 operacja przepisywania na serwerze, 336  
 optymalizacja wielkość pliku biblioteki jQuery, 45  
 opuszczanie strony przez użytkownika, 90  
 ostatnia pozycja (filtrowanie), 68  
 otwieranie strony w nowym oknie, 30

## P

p, 61  
 paginacja, 194  
 wiersze tabeli, 195  
 wynik, 199  
 parent-child (właściwość CSS), 61  
 paski informacyjne, 136  
 pierwsza pozycja (filtrowanie), 68  
 pisania kodu przyspieszenie, 18  
 pobieranie  
 biblioteki jQuery z pamięci podręcznej, 41  
 jQuery, 39  
 kodu html, 78  
 wartości opcji wyboru, 223  
 podmienianie (efekt), 109  
 podpowiedzi  
 treść, 234  
 wyświetlanie, 101  
 zaawansowane, 233  
 pojawianie się elementów witryny, 128  
 pokaz slajdów, 300, 301  
 pokazanie elementu, 119  
 pole  
 danych wejściowych (ograniczenie liczby znaków), 217  
 danych wejściowych, uzyskiwanie wartości, 221

formularza kopiowanie zawartości, 229  
 wprowadzono wyłącznie liczby, 238  
 wyboru  
 anulowanie zaznaczanie wszystkich, 219  
 zaznaczanie wszystkich, 219  
 wymagane, 238  
 poprawności  
 adresu e-mail, 238  
 formularzy sprawdzanie, 235  
 POST (żądanie), 257, 258  
 powodzenie, 253  
 precyzowanie wybieranych elementów, 65  
 processJSON (funkcja), 270  
 Progress Bar (widżet), 293  
 propagacja zdarzenia, 93  
 Prototype, 20, 47  
 przeciąganie, 107  
 przeglądarka  
 Apple iPhone Safari, 333  
 Google Chrome, 334  
 mobilna, 329  
 przejście  
 CSS3, 28  
 do nowej strony za pośrednictwem paska adresu lub odnośnika (zdarzenie), 87  
 przekazywanie metodzie argumentu (kodu HTML), 78  
 przełączanie  
 klasy, 84  
 między efektami  
 pokazywania i ukrywania, 119  
 przesuwania elementu w górę i w dół, 119  
 show i hide, 124  
 przenikanie, 149  
 i odbijanie, 149  
 przepisywanie na serwerze, 336  
 przesuwanie  
 elementu, 125  
 w dół, 119  
 w górę, 119  
 kursora myszy w obręb elementu, 97  
 przewijania zainicjowanie (zdarzenie), 87  
 przyspieszenie pisania kodu, 18  
 pseudoklasa, 66

## Q

qTip (dodatek), 233, 234



## R

ramka  
 1-pikselowa, 58  
 dodanie do każdego elementu, 54  
 korzystająca z obrazu, 331  
 nakładana, 290, 313  
 WordPress, 290  
 ready (zdarzenie), 87  
 Really Simple Syndication, 136  
 rel (znacznik), 74  
 reset (zdarzenia formularza), 111  
 resetujący  
 arkusz stylów CSS, 160  
 kod CSS, 167  
 Resig John, 22, 342  
 Resizable (interakcja), 293  
 resize (zdarzenie), 87  
 rozjaśnianie naprzemienne wierszy, 67  
 rozmiaru okna przeglądarki zmiana (zdarzenie), 87  
 rozmieszczenie kart na stronie, 175  
 rozszerzanie progresywne, 30  
 rozszerzenia pliku sprawdzanie, 240  
 RSS, 136  
 Ruby on Rails, 20

## S

Safari 3.0+  
 zgodność, 27  
 Scriptaculous, 20  
 scroll (zdarzenie), 87  
 select (element), 224  
 select (zdarzenia formularza), 111  
 Selectable (interakcja), 293  
 selektor, 45  
 CSS, 53  
 elementu, 56  
 identyfikatora, 57  
 jQuery, 53  
 nadrzędno-podrzędny, 61  
 potomny, 63  
 Sizzle, 52  
 znaku wieloznacznego, 54  
 Sencha Touch, 339  
 Server error, 253  
 serwer  
 kody odpowiedzi, 253  
 WWW, 32

show, 119  
 Sizzle, 52  
 skrypt harmonijki, 171  
 Slider (widżet), 293  
 Sortable (interakcja), 293  
 sortForce, 200  
 sortList, 200  
 sortList (parametr), 203  
 sortMultiSortKey, 200  
 sortowanie  
 tabeli, 200  
 zmiana kolejności domyślnej, 203  
 sprawdzanie poprawności  
 adresu e-mail, 224  
 formularzy, 235  
 sprite, 167  
 Stevenson Sam, 20  
 strona  
 dynamiczne dodawanie treści, 189  
 opuszczanie przez użytkownika, 90  
 otwieranie w nowym oknie, 30  
 submit (zdarzenia formularza), 111  
 Success, 253  
 symulacja odbijania obiektu na ekranie, 149

## Ś

środowisko programistyczne  
 konfigurowanie, 32  
 lokalne, 32

## T

tabela  
 dodawanie danych, 187  
 filtrowanie danych, 187  
 naprzemienne kolorowanie wierszy, 183  
 sortowanie, 200  
 style CSS, 182  
 usuwanie  
 danych, 187  
 wiersza, 191  
 tablesorter (dodatek), 200, 201, 203  
 Tabs, 293, 300, 301, 311  
 target (atrybut), 30  
 technologia Ajax, 248  
 tekst domyślny, 214  
 testowanie  
 kodu, 38  
 wyników pracy, 32

Text Shadow, 331  
 TextMate (edytor kodu), 32  
 ThemeRoller, 295, 297, 299  
 tickers, 136  
 title, 74, 234  
 treść  
   brak, 252  
   dodawanie, 193  
   dynamiczna, 251  
     dodawanie do strony, 189  
   HTML analiza składniowa na stronie, 264  
   ładowanie z zewnętrznego pliku HTML,  
     250, 254  
   rozmieszczenie na stronie, 175  
   zaawansowane podpowiedzi, 234  
   zmiana po kliknięciu myszą, 97  
 trwale przeniesione, 253  
 tymczasowo przeniesione, 253

## U

ukrycie elementu, 119  
 Ulubione książki (widżet), 265  
 umieszczenie kursora myszy w obrębie  
 elementu, 97  
 Unauthorized, 253  
 unload (zdarzenie), 87, 90  
 upuszczanie, 107  
 URL poprawność, 240  
 usuwanie  
   danych z tabeli, 187  
   elementów HTML, 81  
   klasy, 84  
   kursora myszy z elementu, 97  
   wiersza  
     na podstawie indeksu, 192  
     na podstawie treści, 193  
     tabeli, 191  
 uzyskiwanie wartości pola danych  
 wejściowych, 221

## V

Validate (dodatek), 235  
 Visualize (dodatek), 204

## W

w polu wprowadzono wyłącznie liczby, 238  
 WampServer (środowisko programistyczne), 32  
 wartości  
   opcji wyboru (pobieranie), 223  
   pola danych wejściowych uzyskiwanie, 221  
 wartość wymagana, 240  
 wersje biblioteki jQuery, 41  
 widżet, 136, 292, 294  
 wiele typów elementów (wybieranie), 64  
 wiersz  
   dodawanie na podstawie indeksu, 192  
   naprzemienne rozjaśnianie, 67  
 tabeli  
   naprzemienne kolorowanie, 183  
   usuwanie, 191  
   usuwanie na podstawie  
     indeksu, 192  
     treści, 193  
 window.load, 46  
 witryna z jQuery, 24  
 własny  
   alias, 47  
   dodatek biblioteki jQuery, 315  
 włączanie elementów formularza, 211  
 wstępne ładowanie obrazów, 88  
 wszystkie  
   elementy, 54  
   akapitu (wybieranie), 61  
   hiperłącza na stronie otwierane w nowym  
   oknie, 154  
 wybieranie  
   elementów  
     filtrowanie, 65  
     precyzowanie, 65  
     według klasy, 58  
     z wieloma klasami, 60  
     znajdujących się kilka poziomów niżej, 63  
   wielu typów elementów, 64  
   wszystkich elementów, 54  
 wydajność zwiększanie, 45  
 wykres  
   interaktywny, 204  
   kierunek analizowania danych, 205  
   kolor, 205

## wykres

- obszar wokół wykresów słupkowych, 205
  - położenie etykiet na wykresie kołowym, 205
  - słupkowy, 205
  - szerokość, 205
  - typ, 205
  - tytuł, 205
  - wysokość, 205
  - zmiana obszaru wokół wykresu kołowego, 205
- wyłączanie elementów formularza, 211
- wymagane pola, 238
- wypełnienie, 58
- wyrażenia regularne, 229
- wysyłania formularza, 111, 258
- wyszukiwanie
- opcje alternatywne, 126
  - sugerujące, 304
- wyświetlanie
- dymków informacyjnych, 101
  - jednorazowe, 121
  - podpowiedzi, 101
  - różnych obrazów tła, 331
  - treści w zależności od platformy użytkownika, 336
  - użytkownikowi opcji edycji, 186

**X**

- XHR, 248, 249
- właściwości żądania, 252
- XML, 262
- XML HTTP Request, 248

**Y**

- Yahoo! User Interface (YUI), 20
- Yelp 278
- API, 279
- YUI, 20
- obsługa przez przeglądarki, 20

**Z**

- zaawansowane podpowiedzi, 233
- zainicjowanie przewijania (zdarzenie), 87
- zalety jQuery, 25
- degradacja, 29
  - dokumentacja, 25

## kod, 26

- łączenie metod w łańcuchach, 27
  - Open Source, 25
  - rozszerzanie progresywne, 30
  - stosowania niekłopotliwego kodu JavaScript, 29
  - zgodność z CSS3, 28
  - zgodność z przeglądarkami, 27
- załadowanie
- dokumentu HTML (zdarzenie), 87
  - wszystkich zasobów (zdarzenie), 87
- zamknięcie okna przeglądarki, 87
- zaznaczanie
- tekstu wewnątrz elementu, 111
  - wszystkich pól wyboru na stronie internetowej, 219
- zdarzenia, 87
- delegowanie, 92
  - klawiatury, 113
  - mysz, 97
  - propagacja, 93
- zmiana
- nieprzezroczystości menu, 163
  - rozmiaru okna przeglądarki (zdarzenie), 87
- zmiennie, 47
- zmniejszanie przezroczystości elementu, 119
- znacznik
- atrybut, 74
  - kotwicy, 74
- znak wieloznaczny, 54, 74
- znaki w polu danych wejściowych (ograniczenie liczby), 217
- znikanie elementów witryny, 128
- zresetowania formularza, 111
- zwiększanie
- przezroczystości elementu, 119
  - wydajności, 45
- zwięzłość kodu, 27
- zwolnienie
- klawisza, 113
  - przycisku myszy, 97

**Ż**

- żądanie
- GET, 256
  - POST, 257, 258

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

**John Rutter** jest projektantem i programistą stron internetowych z ponaddziesięcioletnim 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://www.rutter.com>), na którym publikuje kursy dotyczące technologii jQuery, PHP, Magento, WordPress, CSS i HTML.



**Helion**

Sprawdź najnowsze promocje:  
• <http://helion.pl/promocje>  
Książki sącześnie dostępne:  
• <http://helion.pl/bestsellery>  
Zamów informacje o nowościach:  
• <http://helion.pl/newsaci>

Helion SA  
ul. Rakowicki 1c, 44-100 Gliwice  
tel.: 32 230 18 43  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

**helion.pl**  
najlepsze  
internetowa

**WILEY**

ISBN 978-83-246-3316-6



Cena 59,00 zł

9 788324 633166

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

Zamówienia telefoniczne:  
**0 801 339900**

**0 601 339900**

Informatyka w najlepszym wydaniu