jQuery

OD NOWICJUSZA DO WOJOWNIKA NINJA

EARLE CASTLEDINE, CRAIG SHARKIE

sitepoint

Helion

PODKRĘĆ STRONĘ Z JQUERY!

Tytuł oryginału: jQuery: Novice to Ninja by Earle Castledine and Craig Sharkie

Tłumaczenie: Marek Pętlicki

ISBN: 978-83-246-3618-1

© Helion S.A. 2012
Authorized translation of the English edition of jQuery: Novice to Ninja, 1st Edition ISBN 9780980576856
© 2010, SitePoint Pty. Ltd. This translation is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

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

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

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

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

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/jqnoni
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Wstęp ... 11
Kto powinien przeczytać tę książkę ... 11
Co znajdziesz w tej książce ... 12
Podziękowania ... 14
Konwencje zastosowane w książce ... 14

Rozdział 1. **Zakochać się w jQuery ... 17**
Co szczególnego ma w sobie jQuery? ... 18
Zgodność z różnymi przeglądarkami ... 18
Selektory CSS3 ... 19
Użyteczne narzędzia ... 19
jQuery UI ... 19
Wtyczki ... 20
Czystość kodu HTML ... 21
Szeroka popularność ... 21
Gdzie jest haczyk? ... 22
Pobieranie i instalacja jQuery ... 23
Pobieranie jQuery ... 23
Google CDN ... 24
Codzienne wersje rozwojowe i Subversion ... 25
Wersja nieskompresowana czy skompresowana? ... 25
Anatomia skryptu jQuery ... 26
Alias jQuery ... 26
Analiza wywołań jQuery ... 26
Elementy dokumentu HTML, czyli DOM ... 27
Jeśli się zdecydujesz… ... 29

Rozdział 2. **Wybieranie, dekorowanie i rozszerzanie ... 31**
Czekamy na załadowanie strony ... 32
Selektory — serce jQuery ... 32
Proste selektory ... 33
Zawężanie selekcji ... 35
Testujemy selekcję ... 35
Filtry ... 36
Wybieranie wielu elementów ... 36
Jak tworzyć dobre selektory ... 37

Kup książkę Poleć książkę

Dekorowanie — obsługa CSS w jQuery 37
Odczytywanie właściwości CSS 37
Ustawianie właściwości CSS 38
Klasy 40
Rozszerzanie — dodawanie efektów za pomocą jQuery 42
Ukrywanie i wyświetlanie elementów 42
Stopniowe udoskonalanie 46
Dodawanie elementów 47
Usuwanie istniejących elementów 49
Modyfikowanie istniejących elementów 50
Podstawowe animacje — efektowne ukrywanie i wyświetlanie 50
Funkcje zwrotne 52
Kilka sztuczek 53
Podświetlanie przy najechaniu myszą 53
Słodkie tajemnice 54
Zanim przejdziemy dalej 55

Rozdział 3. **Animacje, przewijanie i zmiana rozmiaru 57**

Animacje 57
Animowanie właściwości CSS 57
Animacja koloru 59
Dynamika animacji 60
Zaawansowane opcje dynamiki 61
Podskakujące panele 62
Kolejka animacji 65
Łańcuchy metod 65
Zatrzymywanie łańcucha 66
Animowana nawigacja 67
Animowana nawigacja, wersja 2 69
Biblioteka jQuery UI 71
Daj się poruszyć! 73
Przewijanie 73
Zdarzenie przewijania 73
Pływająca nawigacja 74
Przewijanie dokumentu 75
Modyfikacja paska przewijania 77
Zmiana rozmiaru 78
Zdarzenie zmiany rozmiarów 79
Elementy obsługujące zmianę rozmiaru 81
Właśnie tak się animuje, przewija i zmienia wymiary 86

Kup książkę Poleć książkę

Rozdział 4. **Ilustracje i pokazy slajdów 87**
Lightbox 87
Własna wersja lightboksa 88
Diagnozowanie skryptu za pomocą console.log 91
ColorBox — wtyczka typu lightbox 92
Przycinanie ilustracji za pomocą Jcrop 95
Pokazy slajdów 98
Pokazy slajdów z przenikaniem 98
Przewijane pokazy slajdów 109
Widżet galerii obrazów w stylu iPhoto 115
Pełny obraz 120

Rozdział 5. **Menu, karty, dymki i panele 121**
Menu 121
Rozwijane menu pionowe 122
Ikony stanu zwinięcia lub rozwinięcia menu 126
Rozwijanie menu po zatrzymaniu nad nim wskaźnika myszy 127
Rozwijane menu poziome 128
Menu akordeonowe 132
Prosty akordeon 132
Akordeony wielopoziomowe 135
Akordeon w jQuery UI 137
Karty 138
Prosta implementacja kart 138
Karty w jQuery UI 140
Panele 143
Pojawiający się formularz logowania 143
Panel wysuwany 145
Dymki podpowiedzi 147
Proste dymki podpowiedzi 148
Zaawansowany dymek podpowiedzi 151
Zamówienie z menu 156

Rozdział 6. **Konstrukcja, AJAX i interakcje 157**
Konstrukcja — najlepsze praktyki 157
Czystsze jQuery 158
Szablony po stronie klienta 162
Programowanie pod kątem wersji przeglądarki (jest złe) 164
Przyspieszony kurs technologii AJAX 166
Czym jest AJAX? 166
Ładowanie zewnętrznego kodu HTML 166
Rozszerzanie możliwości odnośników za pomocą techniki hijax 167

Kup książkę Poleć książkę

Wybieranie fragmentu kodu HTML za pomocą selektorów ... 168
Zaawansowane mechanizmy metody load() ... 169
Przygotuj się na przyszłość — live oraz die ... 170
Pobieranie danych za pomocą metody $.getJSON() ... 171
Wyszukiwarka komentarzy Twittera po stronie klienta ... 172
Narzędzia obsługi AJAX w jQuery ... 173
Często używane ustawienia żądań AJAX ... 174
Ładowanie zewnętrznych skryptów — $.getScript() ... 174
Żądania GET i POST ... 175
Zdarzenia związane z żądaniami AJAX ... 175
Interakcja z użyciem technologii AJAX ... 176
Galeria obrazów oparta na AJAKSIE ... 177
Słowa kluczowe obrazów ... 189
Wojownicy ninja, mistrzowie techniki AJAX? Obecni! ... 193

Rozdział 7. **Formularze, widżety i okna dialogowe ... 195**
Formularze ... 195
Prosta walidacja formularzy ... 196
Walidacja formularzy z wtyczką Validation ... 199
Wskaźnik maksymalnej długości tekstu ... 201
Wskazówki formularza ... 202
Zaznaczenie pól opcji ... 204
Programowe modyfikowanie wartości pól ... 205
Autouzupełnianie ... 208
Widżet oceny ... 210
Elementy kontrolne ... 215
Widżet wyboru daty ... 215
Suwaki ... 218
Przeciągnij i upuść ... 221
Sortowanie obiektów w jQuery UI ... 226
Pasek postępu ... 228
Okna dialogowe i powiadomienia ... 230
Proste modalne okno dialogowe ... 230
Dialog z jQuery UI ... 233
Powiadomienia w stylu Growl ... 236
Powiadomienia 1-up ... 238
Zmiana formularzy na lepsze ... 241

Rozdział 8. **Listy, drzewa i tabele ... 243**
Listy ... 243
Element selectable z jQuery UI ... 244
Sortowanie list ... 249
Zarządzanie listami pól wyboru ... 250

Kup książkę Poleć książkę

Drzewa ... 254
Drzewo rozwijane ... 254
Delegacja zdarzeń ... 257
Tabele ... 259
Blokada nagłówków tabeli ... 259
Powtarzanie nagłówka ... 262
Siatki danych ... 264
Zaznaczanie wierszy tabel z użyciem pól opcji ... 272
Początek listy sukcesów ... 274

Rozdział 9. **Wtyczki, motywy i techniki zaawansowane ... 277**
Wtyczki ... 277
Tworzenie wtyczki ... 278
Zagadnienia zaawansowane ... 285
Rozszerzanie jQuery ... 285
Zdarzenia ... 290
Inne szczegółowe zagadnienia szkolenia wojownika ninja jQuery ... 300
Unikanie konfliktów nazw ... 300
Kolejkowanie animacji i usuwanie kolejki ... 301
Traktowanie obiektów JavaScript tak jak obiektów jQuery ... 303
Tworzenie motywów wizualnych ... 304
Używanie galerii motywów ... 304
Definiowanie własnego motywu ... 305
Obsługa motywów we własnych komponentach ... 306
StarTrackr! — epilog ... 308

Dodatek A **Podręczna ściągawka ... 309**
Opcje metody $.ajax() ... 309
Flagi ... 309
Ustawienia ... 310
Funkcje obsługi zdarzeń AJAX i funkcje zwrotne ... 312
Opcje $.support ... 312
Zdarzenia ... 315
Właściwości zdarzeń ... 315
Metody zdarzeń ... 315
Własne obiekty zdarzeń ... 316

Dodatek B **Specyfika języka JavaScript ... 317**
Koercja typów ... 317
Operatory równości ... 318
Prawda i fałsz ... 319

Kup książkę Poleć książkę

Dodatek C **Zaawansowane narzędzia do tworzenia wtyczek 321**
Selektor i kontekst 321
Stos jQuery 322
Minimalizacja 323

Skorowidz 325

Kup książkę Poleć książkę

Rozdział 3

Animacje, przewijanie i zmiana rozmiaru

Klient jest niezwykle zadowolony z pierwszej rundy naszych zmian i poprawek i chce pójść jeszcze dalej. Jego firma współpracuje z przemysłem rozrywkowym i, jego zdaniem, strona WWW powinna odzwierciedlać ekscytującą i dynamiczną naturę tej gałęzi przemysłu. Wierzy też, że atrakcyjne animacje pomogą zwiększyć sprzedaż.

„Sądzę, że potrzebujemy czegoś zgodnego z Web 2.0, o którym wiele słyszałem. Czy możecie coś zrobić, żeby strona bardziej kojarzyła się z Web 2.0?".

Rzeczywiście, możemy. Klient wręcza nam kolejną listę sugerowanych poprawek, pełną ekscytujących zmian. Ta lista pomoże nam wyjść poza proste zadania polegające na ukrywaniu i wyświetlaniu elementów i przybliży nas do statusu wojownika ninja jQuery.

Animacje

Biblioteka jQuery powstała właśnie po to, żeby tworzyć animacje. Stopniowe zanikanie komunikatu o błędzie po nieudanym logowaniu, rozwijanie menu czy nawet wizualne przewijanie stron treści przypominające przekładanie kart książki lub gry polegające na strzelaniu do kosmitów: wszystkie te zadania są łatwe do realizacji dzięki wbudowanym metodom, wspomaganym niezliczoną rzeszą wtyczek.

Animowanie właściwości CSS

Do tej pory poznaliśmy pewne istotne podstawy animacji w jQuery: wsuwanie, zanikanie i efektowne ukrywanie oraz wyświetlanie elementów. Nie mieliśmy jednak większej kontroli nad tym, co jest animowane i w jaki sposób. Nadszedł czas na wprowadzenie bardzo ważnej metody jQuery, nazwanej `animate()`, która pozwala na zastosowanie animacji do wielu właściwości CSS, umożliwiając tworzenie niesamowitych efektów na stronach WWW. Przyjrzyjmy się praktycznemu przykładowi zastosowania tej metody:

Kup książkę Poleć książkę

rozdzial_03/01_animating_css/script.js (fragment)

```
$('p').animate({
  padding: '20px',
  borderBottom: '3px solid #8f8f8f',
  borderRight: '3px solid #bfbfbf'
}, 2000);
```

Powyższy kod realizuje animację na wszystkich akapitach strony, zmieniając atrybut `padding` z domyślnej wartości do `20px` oraz dodając ramkę u góry i po prawej stronie. Animacja będzie wykonywała się przez dwie sekundy (2000 milisekund).

W celu zastosowania metody `animate()` przekazujemy literał obiektu określający właściwości, które mają być animowane. Właściwości są określane jako pary `klucz:wartość` w taki sam sposób jak w metodzie `css()`. Należy jednak pamiętać o jednej istotnej zasadzie: zamiast `margin-left` należy zastosować nazwę `marginLeft`, a zamiast `background-color` nazwę `backgroundColor`, czyli nie wolno używać nazw właściwości z łącznikiem, a należy stosować konwencję nazw camelCase. Każda nazwa właściwości CSS zawierająca łącznik musi być zmodyfikowana w ten sposób[1].

Parametr czasu animacji działa tak samo jak w przypadku animacji, które tworzyliśmy w rozdziale 2.: można przekazać liczbę określającą czas w milisekundach lub słowne określenie czasu trwania: `slow`, `fast`, `normal`. Wartości właściwości wymiarowych CSS mogą być podane w pikselach, jednostce `em`, procentach lub punktach, na przykład można napisać `100px`, `10em`, `50%` lub `16pt`.

Użyte wartości mogą też być **względne** w stosunku do aktualnych: wystarczy przed wartością użyć znaków `+=` lub `-=`, co spowoduje, że zostanie ona odpowiednio zwiększona o dodaną wartość lub o nią zmniejszona. Użyjemy tych możliwości do tego, aby pozycje w menu nawigacyjnym poruszały się, gdy najedziemy na nie wskaźnikiem myszy, czyli musimy zastosować metodę obsługi zdarzenia `hover()`:

rozdzial_03/02_relative_css_animation/script.js (fragment)

```
$('#navigation li').hover(function() {
  $(this).animate({paddingLeft: '+=15px'}, 200);
}, function() {
  $(this).animate({paddingLeft: '-=15px'}, 200);
});
```

Po najechaniu wskaźnikiem myszy na pozycję w menu odsunie się ona płynnie w prawą stronę.

Metody `animate` można użyć do uzyskania precyzyjnej kontroli nad wyświetlaniem, ukrywaniem i funkcjami przełączania widoczności elementów, które poznaliśmy w rozdziale 2. Wystarczy dla odpowiedniej właściwości CSS zastosować wartość `show`, `hide` lub `toggle` zamiast wartości liczbowej:

[1] Inny sposób rozwiązania tego problemu polega na ujęciu w cudzysłowy nazw kluczy zawierających niedozwolone znaki — *przyp. tłum.*

Kup książkę Poleć książkę

rozdzial_03/03_animate_show_hide (fragment)

```
$('#disclaimer').animate({
  opacity: 'hide',
  height: 'hide'
}, 'slow');
```

Obserwowanie animacji elementów to niezwykle satysfakcjonujący wynik pracy programisty. W ramach ćwiczenia pobaw się animowaniem różnych atrybutów CSS i spróbuj uzyskać ciekawe i unikalne efekty. Metoda `animate()` obsługuje więcej opcji, wiele z nich służy do zaawansowanej kontroli parametrów animacji. Omówimy je szczegółowo w dalszej części rozdziału.

Animacja koloru

Gdy już przekonasz się, jak wiele frajdy daje metoda `animate()`, zapewne zechcesz animować kolor. Jednak ta operacja jest nieco bardziej skomplikowana, ponieważ pośrednie wartości koloru, między początkowym a końcowym punktem animacji, muszą być wyliczane w specjalny sposób. W przeciwieństwie do wysokości czy szerokości elementu, które zmieniają się w prosty, liniowy sposób, do wyliczenia koloru jQuery potrzebuje dodatkowych obliczeń, aby upewnić się, że barwa znajduje się, powiedzmy, w trzech czwartych odległości między niebieskim a pomarańczowym.

Ten mechanizm wyliczania koloru został pominięty w podstawowej bibliotece. Taka decyzja ma sens, jeśli się głębiej zastanowić: w większości zastosowań mechanika animacji koloru nie jest niezbędna, dzięki czemu plik podstawowej biblioteki może zachować rozsądne rozmiary. Jeśli ktoś chce animować kolory, potrzebna mu jest wtyczka Color Animations[2].

Użycie wtyczek

Oficjalne repozytorium wtyczek projektu jQuery[3] zawiera spory zbiór wtyczek, który stale się rozrasta. Niektóre z nich są bardzo użyteczne, inne mniej. Wtyczki można wyszukiwać po nazwie, kategorii (jak efekty czy narzędzia) albo po ocenie przyznanej przez społeczność użytkowników jQuery.

Gdy już znajdziesz interesującą Cię wtyczkę, pobierz ją do odpowiedniego miejsca w swoim projekcie (najczęściej będzie to ten sam katalog, w którym zapisałeś plik biblioteki jQuery). Warto przeczytać plik ***readme*** lub odpowiednią dokumentację na stronie WWW, ale najczęściej wtyczkę wystarczy o prostu zaimportować w dokumencie HTML w ten sam sposób jak inne biblioteki języka JavaScript.

W jaki sposób korzystać z wtyczki, to już zagadnienie specyficzne dla każdej z nich. W tym przypadku nie uniknie się konieczności zapoznania z dokumentacją.

Po pobraniu i zaimportowaniu wtyczki Color Animations można animować kolory za pomocą metody `animate()` — tak samo jak inne właściwości CSS. Spróbujmy zmienić kolor notki prawnej przez okres dwóch sekund, żeby zwrócić na nią uwagę użytkowników strony.

[2] *http://plugins.jquery.com/project/color*

[3] *http://plugins.jquery.com/*

Kup książkę Poleć książkę

rozdzial_03/04_color_animation (fragment)

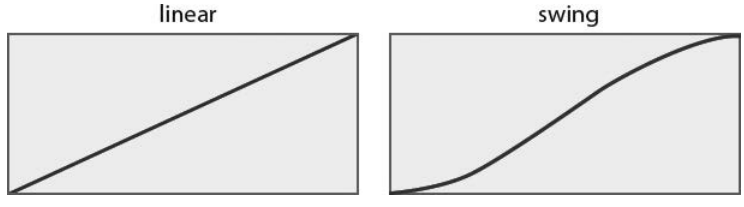
```
$('#disclaimer').animate({'backgroundColor':'#ff9f5f'}, 2000);
```

Dzięki tej technice notka prawna jest znacznie lepiej widoczna.

Dynamika animacji

Dynamika animacji (ang. *easing*) określa sposób, w jaki prędkość animacji zmienia się w czasie. Dynamika animacji jest zdefiniowana za pomocą algorytmu matematycznego i pozwala zmieniać prędkość w trakcie trwania animacji. Dzięki temu, że korzystamy z jQuery, możemy odłożyć na inną okazję konieczność odkurzenia wiadomości z zakresu równań matematycznych.

Standardowa biblioteka jQuery oferuje dwie opcje dynamiki animacji: `linear` i `swing`. Każde wywołanie metody `animate()` pozwala na użycie jednej z tych dwóch opcji kontroli dynamiki. Różnicę między nimi można zobaczyć w formie wykresów na rysunku 3.1. Wykres prezentuje sposób, w jaki prędkość animacji zmienia się w czasie dla każdej z tych opcji.



Rysunek 3.1. Standardowe opcje dynamiki animacji w jQuery

Dynamika typu `swing` polega na powolnym rozpoczęciu, następnie prędkość znacznie się zwiększa, a przy końcu znów spada. Wizualnie dynamika typu `swing` wygląda znacznie naturalniej w porównaniu z dynamiką liniową, dlatego jQuery stosuje ją domyślnie w metodzie `animate()`, jeśli dynamika nie zostanie określona w sposób jawny.

Dynamika liniowa (opcja `linear`) nie wykorzystuje przyspieszania ani spowalniania, animacje odbywają się ze stałą prędkością. Takie animacje w większości przypadków wyglądają dość nudno i sztywno, ale warto wypróbować tę opcję, może okazać się użyteczna w specyficznych zastosowaniach.

W ramach przykładu dodamy animację w efekcie kliknięcia pierwszego akapitu na naszej stronie WWW: po pierwszym kliknięciu powiększymy jego wysokość, wykorzystując dynamikę liniową, a po drugim zmniejszymy ją do oryginalnych rozmiarów, stosując dynamikę typu `swing`. Różnica jest dość subtelna, ale jeśli powtórzysz animację kilka razy, łatwo ją zauważysz. Animacja zmniejszania wysokości wydaje się bardziej naturalna.

rozdzial_03/05_easing/script.js (fragment)

```
$('p:first').toggle(function() {
  $(this).animate({'height':'+=150px'}, 1000, 'linear');
}, function() {
  $(this).animate({'height':'-=150px'}, 1000, 'swing');
});
```

Kup książkę Poleć książkę

W powyższym listingu znajdziemy sporo szczegółów specyficznych dla jQuery, warto zatem na chwilę się zatrzymać, żeby przeanalizować ten kod i zrozumieć, co się dzieje:

- za pomocą metody `filter()` selektora wybieramy tylko pierwszy akapit,
- do wybranego akapitu podpinana jest metoda obsługi zdarzenia `toggle()` (która jest wywoływana przy kolejnych kliknięciach),
- w każdej funkcji obsługi zdarzenia wybieramy obiekt `this`, odwołujący się do elementu, na którym zostało odpalone zdarzenie (czyli do klikniętego akapitu),
- pierwsza funkcja obsługi zdarzenia wykorzystuje format `+=`, powodujący zwiększenie wysokości akapitu o 150 pikseli, wykorzystując animację z dynamiką liniową,
- druga funkcja obsługi zdarzenia wykorzystuje format `-=`, powodujący zmniejszenie wysokości akapitu o 150 pikseli, wykorzystując animację z dynamiką typu `swing`.

Jeśli zrozumiałeś ten przykład, należą Ci się gratulacje! Naprawdę łapiesz, o co chodzi w jQuery!

Zaawansowane opcje dynamiki

Jak wspomnieliśmy, dynamika animacji typu `swing` daje znacznie przyjemniejszy efekt, prawdopodobnie odpowiedni dla większości zastosowań. Jednak `swing` i `linear` to jedynie wierzchołek góry lodowej, jeśli chodzi o możliwości w tym zakresie. Do dyspozycji mamy znacznie więcej typów dynamiki. Większość z nich jest zdefiniowana we wtyczce Easing[4], dostępnej w repozytorium wtyczek jQuery.

Wtyczki wchodzące w skład jQuery UI

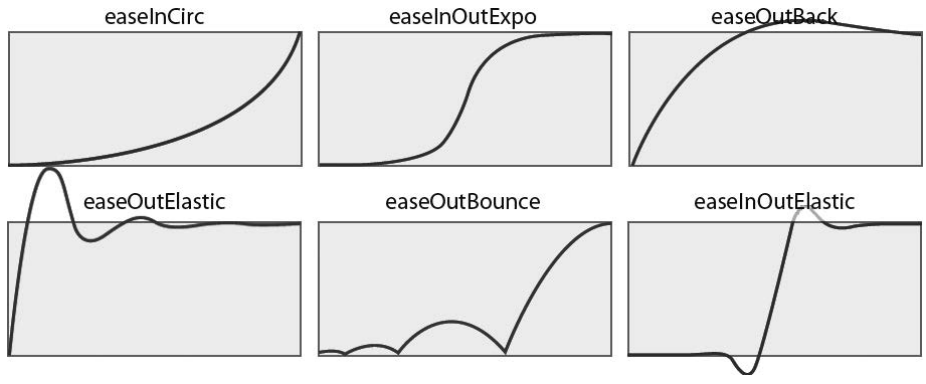
Biblioteka Easing jest również dostępna w sekcji efektów biblioteki jQuery UI, którą będziemy omawiać niebawem. Jeśli zaczyna Cię męczyć duża liczba wtyczek, zajrzyj do punktu „Biblioteka jQuery UI" w dalszej części rozdziału. Ta biblioteka gromadzi dużą liczbę popularnych wtyczek, między innymi do obsługi animacji kolorów, przekształceń między klasami i dynamiki animacji. Wystarczy załadować jedną bibliotekę jQuery UI, nie trzeba importować każdej z tych wtyczek z osobna.

Wystarczy pobrać plik JavaScript z kodem wtyczki i załadować go w pliku HTML za importem biblioteki jQuery. Wtyczka Easing nie oferuje nowych metod, zamiast tego mamy do dyspozycji około 30 opcji dynamiki animacji. Wyjaśnianie zasad działania każdej z nich byłoby doskonałym testem dla wyobraźni najlepszych pisarzy, zamiast tego proponujemy jednak zapoznać się z rysunkiem 3.2, który wyjaśnia algorytmy działania kilku z opcji w sposób graficzny.

Można zauważyć, że niektóre algorytmy wychodzą poza wykres. W takim przypadku animowany obiekt przechodzi za punkt zatrzymania, żeby po chwili wrócić do niego i tam się zatrzymać. Efekt wygląda podobnie do zabawki zaczepionej na gumce, która delikatnie sprężynując, przyciąga ją na swoje miejsce.

[4] *http://plugins.jquery.com/project/Easing*

Kup książkę Poleć książkę



Rysunek 3.2. Zaawansowane opcje dynamiki animacji

Dodatkowych algorytmów dynamiki animacji używa się tak samo jak dostępnych standardowo: wystarczy podać ich identyfikator w parametrze metody `animate()`. Mamy spory wybór, warto więc wypróbować choćby kilka z nich, żeby nabrać wyobrażenia o ich działaniu:

rozdzial_03/06_other_easing_options/script.js (fragment)

```
$('p:first').animate({height: '+=300px'}, 2000, 'easeOutBounce');
$('p:first').animate({height: '-=300px'}, 2000, 'easeInOutExpo');
$('p:first').animate({height: 'hide'}, 2000, 'easeOutCirc');
$('p:first').animate({height: 'show'}, 2000, 'easeOutElastic');
```

Tylko popatrz, jak skacze ten akapit! Zapewne zastanawiasz się, skąd wzięły się nazwy opcji dynamiki animacji, albo chcesz poznać ich pełną listę. Twórcą algorytmów jest Robert Penner, który algorytmy dynamiki animacji zdefiniował w formie formuł matematycznych i umieścił na swojej stronie WWW[5].

Najlepszym sposobem na zapoznanie się z formułami stojącymi za każdą z opcji dynamiki jest analiza kodu źródłowego wtyczki Easing.

Czas na zabawę

Zrób sobie przerwę i przetestuj opcje dynamiki animacji zdefiniowane przez wtyczkę. Istnieje niewielka szansa, że będziesz kiedykolwiek w stanie użyć wszystkich, ale ich znajomość pozwoli dokonać świadomego wyboru tej z nich, która najlepiej odpowiada Twoim oczekiwaniom. Poza tym zabawa z funkcjami pomaga utrwalić ich znajomość, co jest bardzo ważną bronią w rękach wojownika ninja jQuery!

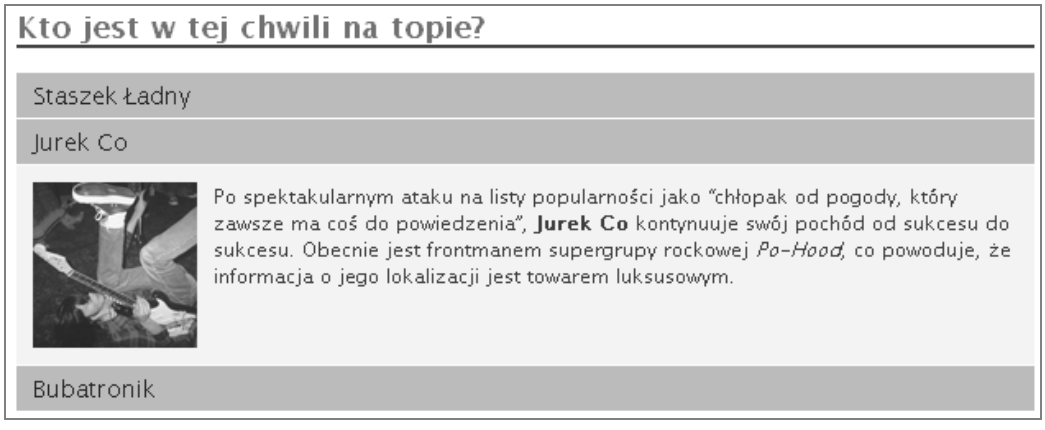
Podskakujące panele

Dowiedzieliśmy się nieco na temat metody `animate()`, zajrzyjmy więc do najnowszej listy zmian zgłoszonych przez klienta. Dzisiejsza lista dotyczy niezwykle istotnej nowości w serwisie StarTrackr! — chodzi o codziennie modyfikowaną listę celebrytów pod nazwą „Kto jest w tej chwili na topie?" (w skrócie KJWTCNT). Ta lista ma zawierać informacje o celebrytach, o których z jakichś

[5] *http://www.robertpenner.com/easing/*

powodów (pozytywnych lub negatywnych) nagle zrobiło się głośno, wraz z ich zdjęciem i krótką informacją biograficzną. Listę zaimplementujemy za pomocą animacji i technik definiowania dynamiki w postaci ruchomych paneli, które mogą być niezależnie otwierane lub zamykane.

Wygląd tego widżetu przedstawia rysunek 3.3.



Rysunek 3.3. Panele informacji o celebrytach

W naszym HTML-u ten element strony zaimplementujemy w postaci obiektu `div` zawierającego informacje o wszystkich celebrytach. Każdy panel będzie składał się z nagłówka `h3`, po którym wystąpi kolejny obiekt `div` zawierający zdjęcie i krótki akapit:

rozdzial_03/07_bouncy_content_panes/index.html (fragment)

```
<div id="bio">
  <h2> Kto jest w tej chwili na topie?</h2>

  <h3>Staszek Ładny</h3>
  <div>
    <img src="../../images/beau_100.jpg" width="100" height="100 "/>
    <p>Notka o Staszku</p>
  </div>

  <h3>Jurek Co</h3>
  <div>
    <img src="../../images/johnny_100.jpg" width="100" height="100 "/>
    <p>Notka o Jurku</p>
  </div>

  <h3>Bubatronik</h3>
  <div>
    <img src="../../images/glenda_100.jpg" width="100" height="100 "/>
    <p>Notka o Bubatronik</p>
  </div>
</div>
```

Gdy użytkownik klika jeden z nagłówków, panel rozwija się lub zwija, zależnie od stanu. Panele można dowolnie stylizować, ale najczęściej stosowana technika polega na uzyskaniu nagłówka o wyróżniającym się kolorze tła, co jasno sugeruje, że to jest element, który można kliknąć.

Kup książkę Poleć książkę

Skacząca animacja

Warto mieć świadomość jednego szczegółu związanego z animowaniem obiektów: gdy animowany obiekt znajduje się blisko nagłówka, animacja „szarpie", co jest szczególnie widoczne przy ukrywaniu. Dzieje się tak dlatego, że margines nagłówka jest usuwany po ukryciu obiektu leżącego bezpośrednio za nim. Proste obejście tego problemu polega na ustawieniu marginesów nagłówka na zero.

Chcemy uniknąć wyświetlania paneli przy ładowaniu strony, więc w pierwszym kroku ukryjemy wszystkie obiekty zawierające treść:

rozdzial_03/07_bouncy_content_panes/script.js (fragment)

```
$('#bio > div').hide();
```

Jeśli jednak chcemy, żeby któryś z paneli był domyślnie wyświetlony, możemy to określić w tym miejscu. To uświadomi użytkownikom, że pod nagłówkami istnieją ukryte panele, więc warto je kliknąć. Implementacja tego w jQuery to proste zadanie: wykorzystujemy filtr `:first`, za pomocą którego wybieramy pierwszy panel, po czym go wyświetlamy.

```
$('#bio > div:first').show();
```

Selektor potomków

W powyższym przykładzie zastosowaliśmy nowy selektor, o którym warto powiedzieć nieco więcej. Jest to selektor **potomków**, wykorzystujący znak większości (>). Selektor potomków wybiera element nadrzędny (lewa strona znaku >), a następnie jego elementy podrzędne (jeden poziom w dół) dopasowane do podselektora (po prawej stronie znaku >). Jeśli pominiemy selektor potomka (znak >), ten kod wybierze wszystkie elementy `div` znajdujące się w elemencie o `id` równym `bio`, nawet jeśli znajdą się na innym poziomie zagnieżdżenia. Więcej na temat tego selektora można znaleźć w dokumentacji jQuery[6].

Nasz dokument zawiera już niezbędny kod HTML, wystarczy więc dodać trochę kodu jQuery do obsługi interakcji. Ujawnienie ukrytej treści wymaga obsługi zdarzenia `click`, odszukania następnego elementu (w którym znajduje się treść) i wyświetlenia go, co już robiliśmy w rozdziale 2. Tym razem jednak zastosujemy dynamikę animacji typu `bounce`, co spowoduje, że panele będą się zachowywać jak upadająca piłka, która chwilę podskakuje, zanim się zatrzyma:

rozdzial_03/07_bouncy_content_panes/script.js (fragment)

```
$('#bio h3').click(function() {
  $(this).next().animate(
    {'height':'toggle'}, 'slow', 'easeOutBounce'
  );
});
```

Doskonale sprawdzający się w tym przypadku efekt podskakującej piłki realizuje opcja dynamiki `easeOutBounce`. Koniecznie sprawdź jego działanie w przeglądarce.

[6] *http://docs.jquery.com/Selectors/child*

Kup książkę Poleć książkę

Kolejka animacji

Kolejnym zagadnieniem związanym z metodą `animate()` są opcje dodatkowe. Wywołanie wygląda następująco:

```
animate(parametry, opcje);
```

Opcje definiuje się w postaci literału obiektu (zbiór par `klucz:wartość`). Znamy już kilka z nich: czas trwania (ang. *duration*), dynamika (ang. *easing*) czy funkcja zwrotna wykonywana po zakończeniu animacji (`complete`). Istnieją jeszcze dwie: `step` i `queue`. Zanim je wyjaśnimy, rzućmy okiem na składnię wywołania funkcji `animate` z użyciem opcji:

rozdzial_03/08_animation_queue/script.js (fragment)

```
$('p:first').animate(
  {
    height: '+=100px',
    backgroundColor: 'green'
  },
  {
    duration: 'slow',
    easing: 'swing',
    complete: function() {alert('gotowe!');},
    queue: false
  }
);
```

Warto pamiętać, że większość możliwości metody `animate()` można wykorzystać z użyciem prostej składni. Składnia złożona jest używana wyłącznie w celu wykorzystania bardziej zaawansowanych możliwości, jak parametr `queue`.

Parametr ten służy do kontroli kolejki (ang. *queue*) animacji, które mają być wykonane na elemencie. Wywołanie metody `animate()` na elemencie powoduje dodanie tej animacji do kolejki. Element wykonuje kolejkę sekwencyjnie aż do jej wyczerpania. Łatwo to zauważyć, jeśli wykona się sekwencję szybkich kliknięć na animowanym obiekcie.

Istnieje jednak wiele sytuacji, w których tego typu zachowanie jest niepożądane. Czasem chcemy, aby kilka animacji odbywało się jednocześnie. Jeśli wyłączymy kolejkę animacji, kolejne animacje będą wykonywane równolegle.

Kolejka animacji jest kontrolowana za pomocą parametru `queue` oraz metod `stop()`, `queue()` i `dequeue()`. Ta kombinacja metod i parametrów daje nam elastyczną kontrolę nad działaniem animacji. Zanim jednak zagłębimy się w tajniki kontroli animacji, powinniśmy poznać jedną z najważniejszych technik związanych z jQuery.

Łańcuchy metod

Do tej pory wyrażenia jQuery wykonywaliśmy sekwencyjnie, jedno po drugim, ewentualnie zagnieżdżaliśmy je w ramach funkcji zwrotnych. W celu ponownego użycia selektora musieliśmy wywołać go ponownie lub użyć obiektu `this`. Biblioteka jQuery obsługuje jednak mechanizm

Kup książkę Poleć książkę

pozwalający wykonać kilka metod na tym samym selektorze. Mechanizm ten nazywamy łańcuchem (ang. *chaining*) i przejście na poziom ninja wymaga znajomości tego szczegółu.

Łańcuch pozwala wykonać większą liczbę wywołań metod jQuery w pojedynczym wyrażeniu. Łańcuch buduje się przez dopisanie metody do innego wyrażenia. Na przykład połączmy w łańcuch metody `hide()`, `slideDown()` i `fadeOut()`. Nasz element szybko znika, potem wjeżdża na swoje miejsce, a na końcu stopniowo zanika:

```
$('p:first').hide().slideDown('slow').fadeOut();
```

Możemy tworzyć łańcuchy metod o dowolnej długości. Należy jednak zachować czujność: tworzenie kodu w ten sposób łatwo uzależnia! Oprócz wykonywania łańcucha metod na tym samym obiekcie łańcuch może nawigować po obiektach DOM, dodawać i usuwać obiekty po drodze, co może prowadzić do dość potwornych wyrażeń.

Łańcuchy metod warto zapisywać w czytelny sposób, przenosząc kolejne wywołania do kolejnych wierszy. Taki zapis zajmuje więcej miejsca, ale kod jest znacznie czytelniejszy i łatwiejszy w utrzymaniu. Nasz poprzedni przykład możemy zapisać następująco:

```
$('p:first')
  .hide()
  .slideDown('slow')
  .fadeOut();
```

Należy pamiętać, że selektor jQuery po każdym wywołaniu metody zawiera zmodyfikowany przez nią wynik, a nie wartość początkową. Oznacza to, że w kolejnych wywołaniach metod z łańcucha możemy dodawać i usuwać elementy, a zmiany będą miały zastosowanie tylko do aktualnej selekcji.

Jeśli przyjrzysz się niektórym z naszych wcześniejszych przykładów, z pewnością znajdziesz przypadki łańcuchów metod, na przykład `$(this).next().toggle()`. Metoda `next()` przenosi selektor na kolejny obiekt (rodzeństwo), a metoda `toggle()` przełącza jego stan widoczności, nie naruszając oryginalnego obiektu.

W pracy z jQuery będziesz miał mnóstwo okazji do wykorzystania łańcuchów metod, dalsza część książki jest nimi po prostu wypełniona. Między innymi na tym polega radość używania jQuery!

Zatrzymywanie łańcucha

Jeśli pojawi się potrzeba zatrzymania na chwilę łańcucha metod w trakcie realizacji, można wykorzystać metodę `delay()`. Wystarczy podać czas oczekiwania (w milisekundach). Oto prosty przykład takiego użycia:

```
$('p:first')
  .hide()
  .slideDown('slow')
  .delay(2000)
  .fadeOut();
```

Kup książkę Poleć książkę

To wyrażenie wsunie akapit, odczeka dwie sekundy, po czym spowoduje jego stopniowe zanikanie. Metoda `delay()` może być doskonałym sposobem kontroli przebiegu wieloetapowych animacji.

Animowana nawigacja

Tym razem klient upiera się przy swoim: chce, żeby główne menu było animacją we Flashu, z wizualnymi efektami towarzyszącymi nawigacji. „Flash wygląda lepiej". Zapewniasz go, że Flasha masz w małym palcu i że na szybko coś przygotujesz.

OK, klient wyszedł z pokoju. Czas wykorzystać świeżo nabyte umiejętności w celu stworzenia efektownego paska nawigacji, przypominającego w działaniu animację Flash. Menu będzie miało falujący „bąbel" oznaczający daną pozycję, nad którą aktualnie znajduje się wskaźnik myszy. Zrobimy to, wykorzystując wyłącznie wolne technologie: HTML, CSS i JavaScript. A Flash? Nie potrzebujemy żadnego Flasha!

W przykładach wykorzystamy tylko podstawowe animacje, dzięki czemu będą one łatwe do analizy. Na początek zmodyfikujemy CSS obsługujący nasze menu, aby wyświetlało się w poziomie, a nie w pionie. Nasz kod HTML dla menu wygląda następująco:

rozdzial_03/09_animated_navigation/index.html (fragment)

```
<ul id="navigation">
<li><a href="#">Strona główna</a></li>
<li><a href="#">O nas</a></li>
<li><a href="#">Kup teraz!</a></li>
<li><a href="#">Pomysł na prezent</a></li>
</ul>
```

Naszym bąblem będzie pusty element `div`, ustawiający się za elementem nawigacyjnym, nad którym zatrzyma się wskaźnik myszy. Zatem pierwsze zadanie dla jQuery polega na utworzeniu takiego elementu i dopisaniu go do dokumentu:

rozdzial_03/09_animated_navigation/script.js (fragment)

```
$('<div id="navigation_blob"></div>').css({
  width: $('#navigation li:first a').width() + 10,
  height: $('#navigation li:first a').height() + 10
}).appendTo('#navigation');
```

Warto zwrócić uwagę na to, że w **środku** literału obiektu używamy selektora w celu odczytania wymiarów obiektu. Jeśli masz doświadczenie z programowaniem, nie zdziwi Cię to; nowicjuszy uspokajamy: to nie jest nic nadzwyczajnego, wszędzie tam, gdzie można wpisać wartość statyczną, można też użyć wyrażenia wyliczającego (lub odczytującego) wartość. Do odczytanych wymiarów dodajemy 10, dzięki czemu bąbel jest nieco większy od elementu menu, za którym ma się znaleźć.

Mamy już bąbel, teraz potrzebujemy zdarzeń, które wprawią go w ruch. Zdarzeniem będzie umieszczenie wskaźnika myszy nad pozycją w menu, więc użyjemy metody `hover()`. Jak pamiętamy, metoda ta oczekuje dwóch parametrów: pierwszy określa funkcję obsługi zdarzenia `mouseover`, drugi funkcję zdarzenia `mouseout`. Nasze wyrażenie będzie zatem miało następującą strukturę:

Kup książkę Poleć książkę

rozdzial_03/09_animated_navigation/script.js (fragment)

```
$('#navigation a').hover(function() {
  // funkcja obsługi mouseover
  ⋮
}, function() {
  // funkcja obsługi mouseout
  ⋮
});
```

Czas na odrobinę zabawy. Rzućmy okiem na pierwszą funkcję, która jest wywoływana, gdy zatrzymamy wskaźnik myszy nad pozycją menu:

rozdzial_03/09_animated_navigation/script.js (fragment)

```
// funkcja obsługi mouseover
$('#navigation_blob').animate(
  {width: $(this).width() + 10, left: $(this).position().left},
  {duration: 'slow', easing: 'easeOutElastic', queue: false}
);
```

Gdy wskaźnik myszy zatrzyma się nad pozycją w menu, wykonujemy animację dwóch atrybutów naszego bąbla: szerokości i pozycji.

Pozycja elementu jest odczytywana za pomocą metody `position()`. Metoda ta nie powoduje zmian, po prostu zwraca dwie wartości określające przesunięcie lewego górnego wierzchołka obiektu w stosunku do rodzica. Nas interesuje przesunięcie poziome, ponieważ bąbel będzie poruszał się od lewej do prawej.

Opcję `queue` ustawiamy na wartość `false`, ponieważ nie chcemy, aby animacje nawarstwiły się nam, powodując zakłócenia wizualne w obsłudze menu (w przypadku użytkownika szczególnie nerwowo poruszającego myszą nad menu). Po przeniesieniu wskaźnika na inną pozycję menu rozpocznie się inna animacja, niezależnie od tego, czy aktualna skończyła działanie.

Mimo to musimy poinformować jQuery o tym, co ma zrobić, gdy użytkownik przesunie wskaźnik myszy poza pozycję w menu. Ten fragment kodu jest dość podobny do poprzedniego, ale zawiera nieco więcej metod w łańcuchu:

rozdzial_03/09_animated_navigation/script.js (fragment)

```
// funkcja obsługi mouseout
$('#navigation_blob')
  .stop(true)
  .animate(
    {width: 'hide'},
    {duration: 'slow', easing: 'easeOutCirc', queue: false}
  )
  .animate(
    {left: $('#navigation li:first a').position().left;},
    'fast'
  );
}
```

Kup książkę Poleć książkę

Tym razem połączyliśmy w łańcuch dwie **animacje**: pierwsza ukrywa bąbel z użyciem estetycznej dynamiki animacji, druga przenosi go na pozycję początkową, czyli lewą stronę menu.

Warto zwrócić uwagę na metodę `stop()` wykonywaną na początku obsługi zdarzenia. Metoda ta robi dokładnie to, na co wskazuje jej nazwa: zatrzymuje animację. Akceptuje dwa dodatkowe parametry: `clearQueue` i `gotoEnd`. Parametr `clearQueue` ustawiamy na `true`, co powoduje wyczyszczenie kolejki animacji.

Parametr `gotoEnd` jest wykorzystywany w celu ustalenia końcowego stanu elementu, gdyby kolejka animacji została wykonana: obiekt jest natychmiast ustawiany w ten stan (bez animacji). W tym przypadku tego nie chcemy, ponieważ zależy nam, aby kolejne animacje rozpoczęły się od miejsca, w którym bąbel znajduje się w danej chwili, nawet jeśli jest właśnie w połowie drogi między elementami menu.

Pobaw się nieco animacją menu, żeby sprawdzić, jak właściwy dobór parametrów dynamiki animacji wpływa na uzyskanie naturalnego wrażenia działania interfejsu.

To też doskonała okazja, aby sprawdzić inne parametry dynamiki animacji. Wystarczy zastąpić proponowane przez nas innymi z dostępnej listy. Aż trudno uwierzyć, jak wielka jest między nimi różnica! Można też animować inne atrybuty bąbla, na przykład zmieniać jego kolor.

Animowana nawigacja, wersja 2

Jedną z doskonałych cech takich bibliotek jak jQuery jest możliwość szybkiego przetestowania różnych alternatywnych rozwiązań i wybrania tego, które działa najlepiej.

Do powrotu klienta zostało nam kilka godzin. Spróbujmy rozwiązać to samo zadanie w nieco inny sposób. Zobaczymy, co z tego wyniknie.

W tym przypadku nasze pozycje w menu będą miały ukrytą ikonę, która będzie wyskakiwała po najechaniu myszą (co przedstawia rysunek 3.4).

Konfiguracja tego efektu jest prosta. Mamy zwykłą listę nienumerowaną, skonfigurowaną za pomocą CSS w formie poziomej, ale każdy element listy posiada dodatkowy, ukryty element definiujący ikonę. W tym celu w specjalny sposób jest ustawiana wysokość elementu listy, tak że widoczny jest tylko tekst, a na życzenie wysokość jest animowana tak, aby wyświetlić również ikonę.

Rysunek 3.4. Podskakujące animowane menu

Na początek zdefiniujemy style CSS. Nasze menu pozycjonujemy w sposób bezwzględny w stosunku do kontenera (element `div`), w którym musimy zadbać o wystarczającą ilość miejsca na wysuwane ikony. Ikony stanowią tło elementów tekstowych menu, ale tło jest przesunięte, dzięki czemu ikony

Kup książkę Poleć książkę

nie są widoczne. Wysokość elementów tekstowych wynosi 20px, a tło jest przesunięte o 30px w dół. W efekcie tło (czyli ikona) nie jest widoczne przy domyślnej wysokości elementów menu.

Każda pozycja menu ma ustawioną inną ikonę w charakterze tła:

rozdzial_03/10_animated_navigation_2/navigation.css (fragment)

```
#container {
  position: relative;
}
#navigation {
  position:absolute;
  width: inherit;
  top: 0;
  right: 0;
  margin: 0;
}
#navigation li {
  height:20px;
  float:left;
  list-style-type: none;
  width:70px;
  padding:3px;
  border-right:1px solid #3687AF;
  background-color: #015287;
  background-repeat: no-repeat;
  background-position: center 30px;
}
#navigation li a {
  color: #FFFFFF;
}
#navigation #home{
  background-image:url('icon_home.png');
}
⋮
```

Jedyną nowością w powyższym kodzie jest zatrzymanie akcji w celu wyczyszczenia kolejki animacji, zarówno w obsłudze zdarzenia `mouseover`, jak i `mouseout`. Następnie animujemy wysokość elementu w celu wyświetlenia ukrytych ikon lub zmniejszamy ją w celu ich ukrycia. Precyzyjnie dobrane ustawienia czasu trwania animacji i jej dynamiki pozwalają uzyskać ciekawy efekt podskakiwania, o który nam chodziło. W tym przypadku również zachęcamy do eksperymentów z różnymi ustawieniami i dostosowania efektu do własnych potrzeb.

rozdzial_03/10_animated_navigation_2/script.js (fragment)

```
$('#nav_stretch li').hover(
  function() {
    $(this)
      .stop(true)
      .animate(
        {height: '60px'},
        {duration: 500, easing: 'easeOutBounce'}
      )
  },
  function() {
    $(this)
```

Kup książkę Poleć książkę

```
      .stop(true)
      .animate(
        {height:'20px'},
        {duration: 500, easing: 'easeOutCirc'}
       )
  }
);
```

I w ten sposób mamy nie jeden, a dwa niezłe przykłady efektów, które możemy pokazać klientowi.

Biblioteka jQuery UI

Jak wspomnieliśmy w rozdziale 1., biblioteka jQuery UI jest kolekcją zaawansowanych widżetów i interakcji, stworzonych z użyciem jQuery. Znajdziemy tu kontrolki wyboru daty, akordeony i obsługę zdarzeń typu „przeciągnij i upuść". Tego typu elementy stają się coraz popularniejsze w aplikacjach WWW. Zanim jednak zaczniemy cieszyć się nowymi zabawkami, musimy zdobyć bibliotekę jQuery UI. Ten proces jest nieco bardziej skomplikowany w porównaniu z podstawową biblioteką jQuery i wtyczkami, których używaliśmy dotychczas. Kłopot wynika z faktu, że pełna wersja biblioteki jQuery UI waży niemal 500 kB (300 kB po zminimalizowaniu)! To dużo. Na szczęście większość projektów wykorzystuje zaledwie ułamek możliwości całej biblioteki. Z tego powodu na stronie projektu jQuery dostępne jest narzędzie pozwalające stworzyć własną, dopasowaną do potrzeb wersję jQuery UI, to znaczy taką, która zawiera tylko potrzebne nam elementy i nic więcej.

Opcje konfiguratora własnej wersji jQuery UI[7] mogą na pierwszy rzut oka wydać się nieco przytłaczające, ale nie należy się zrażać.

Narzędzie konfiguracyjne składa się z kilku sekcji: *Core*, *Interaction*, *Widgets*, *Effects* i *Themes*. Sekcja *Core* dotyczy głównej części biblioteki jQuery UI, na niej swoje działanie opierają komponenty z sekcji *Widgets* i *Interaction*. Dobre podejście polega na odznaczeniu wszystkiego, a następnie zaznaczeniu tylko tego, co niezbędne. Jeśli dany element wymaga zainstalowania innego, ten drugi zostanie zaznaczony automatycznie.

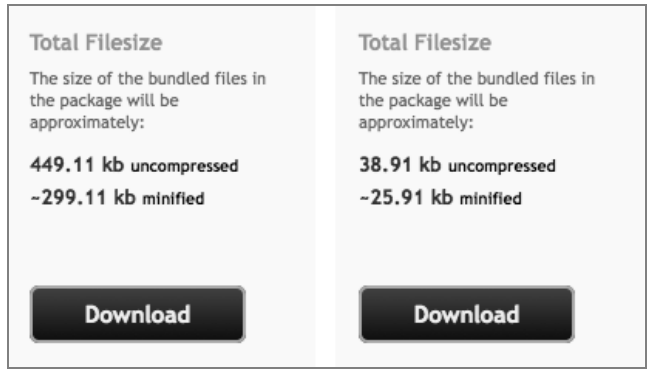
Na etapie tworzenia aplikacji warto zaopatrzyć się w pełną wersję. W ten sposób będziemy mieli wszystko pod ręką, na wszelki wypadek. Po zakończeniu tworzenia aplikacji można wrócić na stronę konfiguratora i stworzyć bibliotekę w wersji produkcyjnej, zawierającą tylko to, co niezbędne. Różnica w rozmiarze między wersją pełną a dostosowaną może być spora, co demonstruje rysunek 3.5.

Jedną z opcji, która ma duży wpływ na wygląd widżeów i interakcji, jest **motyw** (ang. *theme*). Do dyspozycji mamy sporą liczbę gotowych motywów, które wybiera się z listy rozwijanej. W rozdziale 9. zajmiemy się bardziej szczegółowo motywami, pokażemy też, jak tworzyć własne. Na razie jednak, nie chcąc psuć niezłego tempa, użyjemy domyślnego motywu i wracamy do pracy.

Archiwum utworzone przez konfigurator, dostosowane do wersji jQuery UI zawiera sporą liczbę plików. Znajdziemy tam mnóstwo kodu demonstracyjnego i dokumentacji. Warto poszperać w katalogu *development-bundle*, ale niecierpliwych zainteresuje tylko plik *jQuery-ui-1.7.2-min.js* (numer wersji jest oczywiście odpowiedni dla aktualnie najnowszej wersji stabilnej) oraz katalog wybranego motywu.

[7] *http://jqueryui.com/*

Kup książkę Poleć książkę



Rysunek 3.5. Porównanie pełnej wersji biblioteki jQuery UI z wersją dostosowaną

Katalog motywu należy umieścić w lokalizacji dostępnej z poziomu kodu HTML. W przykładach z tej książki jQuery UI umieściliśmy w katalogu *lib* (obok biblioteki jQuery), a pliki motywu w katalogu *css*.

Biblioteka jQuery UI zawiera pakiet Effects, który pozwala implementować interesujące efekty. Zawiera też dodatkowe metody i funkcje pomocne w tworzeniu zaawansowanych animacji. Część z tych możliwości mieliśmy okazję poznać dzięki wtyczkom Color Animations oraz Easing. Obie te wtyczki są zawarte w pakiecie Effects, zatem nie musimy już nic włączać indywidualnie, jeśli planujemy używać biblioteki efektów wchodzącej w skład jQuery UI.

Zanim przejdziemy do dalszych zagadnień, rzućmy okiem na dostępne efekty. Znów pomęczymy nasz pierwszy akapit na stronie i potrząśniemy nim nieco, podświetlimy go na żółto, a na końcu wysadzimy, rozpryskując na kawałki:

rozdzial_03/11_jquery_ui_effects/script.js (fragment)

```
$('p:first')
  .effect('shake', {times:3}, 300)
  .effect('highlight', {}, 3000)
  .hide('explode', {}, 1000);
```

Robi wrażenie! A to oczywiście zaledwie przedsmak możliwości dostępnych efektów. Niektórych z nich można używać tylko w taki sposób jak w powyższym przykładzie, inne stanowią uzupełnienie standardowych parametrów animacji: `hide` (ukrywania), `show` (wyświetlania) lub `toggle` (przełączania widoczności). Przykładem tej drugiej grupy są parametry: `blind`, `clip`, `puff`, `fold` i `slide`.

Nie będziemy ich wszystkich analizować, najlepiej, jeśli spędzisz niedzielne popołudnie na ich wypróbowaniu. Nie wszystkie efekty są brawurowe, ale wiele z nich sprawia wrażenie przydatnych w codziennych zastosowaniach, jak `highlight`, który jest standardowym sposobem oznaczania na przykład nowych wiadomości użytkownika.

Dobrym pomysłem jest samodzielne przetestowanie wszystkich efektów, żeby pamiętać o nich wówczas, gdy okażą się przydatne. A jeśli asortyment efektów dostępny z jQuery UI to dla Ciebie za mało, setki innych czekają w repozytorium wtyczek jQuery!

Kup książkę Poleć książkę

Daj się poruszyć!

Rozumiesz już podstawy tworzenia animacji z jQuery: selektory, funkcje obsługi zdarzeń, funkcje zwrotne, łańcuchy metod i — najważniejszą z nich — metodę `animate()`. Zapoznałeś się też pokrótce z rozbudowaną biblioteką jQuery UI. Najlepszy sposób opanowania tych umiejętności polega na wykorzystaniu ich w praktyce. Zatem idź i wykonuj animacje na wszystkim, co Ci wpadnie w oko. Spróbuj pobawić się z każdą właściwością każdego elementu na każdej stronie, aż poczujesz, że rzeczywiście stały się posłuszne Twojej woli.

Zanim przejdziemy od animacji do innych zaawansowanych mechanizmów związanych z jQuery, zwrócimy uwagę na przewijanie i zmianę rozmiaru elementów. Te zagadnienia są nieco miej efektowne od animacji, ale stanowią istotne elementy obsługi interfejsu użytkownika. Pomogą też utrwalić podstawowe pojęcia związane z jQuery: metody i łączenie ich w łańcuchy. Na co jeszcze czekamy?

Przewijanie

Przewijanie jest zagadnieniem zbliżonym do animacji, ponieważ w tym przypadku elementy również poruszają się na ekranie. W przeciwieństwie do animacji to jednak użytkownik kontroluje przebieg tej operacji. Istnieje mnóstwo sposobów dostosowania tego typu interakcji do własnych potrzeb. W tym punkcie przyjrzymy się menu, które zachowuje swoje położenie nawet mimo przewinięcia strony przez użytkownika, tworzeniem motywów pasków przewijania oraz sposobom zastosowania technik animacyjnych w zagadnieniach związanych z przewijaniem dokumentu.

Zdarzenie przewijania

Zanim będziemy w stanie udoskonalić mechanizm przewijania, musimy wiedzieć, kiedy ono następuje. Jak się okazuje, istnieje zdarzenie związane z przewijaniem. Zdarzenie to nazywa się `scroll` i jest odpalane w przypadku zmiany pozycji przewijania elementu, który ma taką właściwość, jak okno lub `div` z obsługą przewijania. Jeśli zatem użytkownik przewija stronę, odpalane jest zdarzenie, które możemy przechwycić i obsłużyć.

W celu przechwycenia zdarzenia przewijania musimy podpiąć się do elementu przewijanego — najczęściej jest to element okna `window`. Okno jest obiektem JavaScript, zatem przed jego wykorzystaniem musimy przekształcić je na element jQuery, używając selektora.

Oczywiście w celu obsługi zdarzenia przewijania musimy mieć obszar z paskami przewijania. Mamy kilka pomysłów na obsługę zdarzeń przewijania, które mogą spodobać się klientowi, ale w celu nauczenia się, jak działają zdarzenia przewijania, zasymulujemy środowisko przewijania w jednym z elementów `div` na stronie. W tym celu w arkuszu stylów CSS należy ustawić właściwość `overflow: scroll`:

rozdzial_03/12_scroll_event/scroll.css (fragment)

```
#news {
  height: 100px;
  width: 300px;
  overflow: scroll;
}
```

Kup książkę Poleć książkę

Powyższy kod przekształci sekcję wiadomości w mniejszy panel z paskiem przewijania. Teraz przechwycimy zdarzenie przewijania `scroll` i w jego efekcie wyświetlimy dowolny tekst.

rozdzial_03/12_scroll_event/script.js (fragment)

```
$('#news').scroll(function() {
  $('#header')
    .append('<span class="scrolled">Przewijanie!</span>');
});
```

Przy każdej próbie przewijania sekcji wiadomości u góry strony na czerwonym tle pojawi się tekst: „Przewijanie!”. Niezbyt to eleganckie, ale skutecznie demonstruje możliwości kodu. Spróbuj przewijać na różne sposoby: przeciągając myszą lub klikając pasek przewijania, używając kółka myszy czy też klawiszy strzałek po kliknięciu wewnątrz przewijanego obszaru. We wszystkich tych przypadkach jest odpalane zdarzenie przewijania.

Pływająca nawigacja

Wiemy już, kiedy użytkownik przewija okno. W jaki sposób możemy użyć tej informacji w celu udoskonalenia strony? Jednym z często stosowanych przykładów jest pływający panel nawigacyjny. Chodzi o taki sposób prezentowania menu nawigacyjnego, w którym jest ono wyświetlane w stałym miejscu okna przeglądarki, niezależnie od tego, w którym miejscu przewijanego dokumentu w danej chwili się znajdujemy. Tego typu nawigacja zachowuje się tak, jakby menu podążało za użytkownikiem niezależnie od tego, jak przewija dokument. Dzięki obsłudze zdarzenia `scroll` ten efekt jest łatwy do uzyskania: wystarczy, że odczytamy względne położenie strony, po czym przeniesiemy nawigację w to miejsce.

W pierwszym kroku musimy przygotować CSS do obsługi pływającej nawigacji. Do naszego elementu o atrybucie `id` równym `navigation` dodamy deklarację `position: relative;`, dzięki czemu będziemy mogli przemieszczać go względem dokumentu, modyfikując właściwość `top`. Na potrzeby tego ćwiczenia ustawiliśmy bardzo dużą wartość właściwości `height` — po to, by uzyskać pasek przewijania (chyba że ktoś ma monitor o bardzo wysokiej rozdzielczości).

rozdzial_03/13_floating_nav_1/scroll.css (fragment)

```
#navigation {
  position: relative;
}
#content {
  height: 2000px;
}
```

Teraz możemy zająć się naszą pływającą nawigacją. Na pierwszy rzut oka zadanie może się wydać banalnie proste: wystarczy odpowiednio zareagować na zdarzenie przewijania, ustawiając wartość `top` elementu nawigacji:

rozdzial_03/13_floating_nav_1/script.js (fragment)

```
$(window).scroll(function() {
  $('#navigation').css('top', $(document).scrollTop());
});
```

Kup książkę Poleć książkę

Przetestuj ten kod w przeglądarce. Zauważysz, że cel został osiągnięty: po przewinięciu strony panel nawigacyjny zostaje przeniesiony we właściwe miejsce. Skąd wie, gdzie ma się ustawić? Zapytaliśmy o to za pomocą metody `scrollTop()`. Metoda ta zwraca położenie przewijanego elementu, w tym przypadku całego dokumentu: `$(document).scrollTop()`. To jest pozycja lewego górnego wierzchołka okna przeglądarki względem dokumentu.

Oczywiście wszystko działa, ale jest to niezbyt eleganckie: menu gwałtownie „skacze" na właściwe miejsce, przez co prezentuje się mało atrakcyjnie. Powód tego stanu rzeczy powinien być oczywisty, jeśli uważnie przeanalizowałeś pierwszy przykład obsługi zdarzenia przewijania: przy każdej próbie przewinięcia okna przeglądarka odpala **mnóstwo** zdarzeń przewijania. Każde z nich zmusza menu do zmiany pozycji, dlatego nie ma co się dziwić, że nie działa to płynnie.

Wiemy już, w jaki sposób powstrzymać animacje przed kolejkowaniem (patrz punkt „Animowana nawigacja"): służy do tego metoda `stop()`. To pozwoli nam opanować szarpanie, ale nasz efekt nadal jest daleki od doskonałości. Może warto by wzbogacić go animacją z odrobiną eleganckiej dynamiki? Zrobimy to w następujący sposób:

rozdzial_03/14_floating_nav_2/script.js (fragment)

```
$(window).scroll(function() {
  $('#navigation')
    .stop()
    .animate({top: $(document).scrollTop()},'slow','easeOutBack');
});
```

Różnica jest ogromna, a została uzyskana za pomocą tak niewielkiej ilości kodu!

Przewijanie dokumentu

Gdy na jednej stronie znajduje się duża liczba informacji, często stosuje się zbiór odnośników wewnętrznych umieszczonych na początku dokumentu.

Te odnośniki przenoszą użytkownika do odpowiedniego miejsca dokumentu. W punkcie docelowym często umieszcza się odnośnik przenoszący na początek dokumentu, żeby użytkownik mógł wybrać inną pozycję z menu. Dodajmy taki mechanizm na naszej stronie.

Na początek do stopki dokumentu dodamy odnośnik przenoszący na górę strony. W tym celu wystarczy ustawić adres odnośnika (atrybut `href`) na wartość `#`.

Chcielibyśmy, żeby w efekcie kliknięcia odnośnika strona płynnie przewinęła się na początek. Do tego służy atrybut `scrollTop`, który podamy metodzie `animate()`. Chcemy też anulować domyślne działanie odnośnika, w przeciwnym razie przeglądarka przeniesie użytkownika na początek strony, zanim zdążymy wykonać animację. Jeśli stawiasz w języku JavaScript pierwsze kroki, zdradzimy Ci, jak to zrobić: każda funkcja obsługi zdarzenia (w tym kliknięcia przycisku) może zwrócić wartość `false`, co spowoduje anulowanie oryginalnego działania związanego ze zdarzeniem:

```
$('a[href=#]').click(function() {
  $('html').animate({scrollTop: 0},'slow');
  return false; // Zwrócenie false anuluje domyślne działanie związane z kliknięciem
}
```

Kup książkę Poleć książkę

W powyższym kodzie znajdziemy obsługę nowego typu selektora: **selektora atrybutu**. Warunek związany z atrybutem umieszcza się w nawiasach kwadratowych ([]), co pozwala ograniczyć selektor do elementów, których wybrane atrybuty spełniają pewne warunki. W naszym przykładzie szukamy tylko takich elementów `a`, których atrybut `href` ma wartość `#`.

Ten kod działa i jest dość czytelny oraz prosty, ale niestety ma pewną subtelną wadę. Jeśli przeglądarka użytkownika działa w trybie quirks, selektor `$('html')` nie będzie działał. Jeśli nie wiesz, co to jest tryb quirks, przeczytaj podręcznik do CSS w serwisie SitePoint[8]. Strony WWW **powinno** się tworzyć w taki sposób, aby przeglądarka działała w trybie standardowym, ale czasem zdarza się potrzeba pracy ze starym kodem, w przypadku którego nie możemy sobie pozwolić na taki luksus. Aby powyższy kod zadziałał w trybie quirks, musimy użyć selektora `$('body')`. Możemy również użyć selektora `$('html, body')`, co pozwala się upewnić, że kod zadziała w każdym z trybów pracy przeglądarki. Niestety, to ostatnie rozwiązanie sprawia problemy w niektórych wersjach przeglądarki Opera, która (zgodne z logiką) próbuje w tym samym momencie przewinąć obydwa elementy.

Masz prawo się oburzyć: „Przecież jQuery miało rozwiązywać problemy niekompatybilności przeglądarek!". W rzeczywistości to prawda w **większości** przypadków. Ten jednak jest nieco subtelniejszy i nie został rozwiązany w podstawowej wersji jQuery. Na szczęście problem jest dość prosty do obejścia. Co więcej, ktoś już o to zadbał i zaimplementował rozwiązanie (dodając gratis mnóstwo ciekawych funkcji związanych z przewijaniem) w postaci wtyczki ScrollTo.

Wtyczka ScrollTo, dostępna w repozytorium wtyczek[9], zawiera mechanizmy służące do przewijania dokumentu i elementów. Znajdziemy w niej narzędzia w zupełności wystarczające do realizacji wszelkich potrzeb związanych z przewijaniem, jakie napotkaliśmy do tej pory.

Po pobraniu i zaimportowaniu wtyczki możemy poprawić naszą funkcję obsługi kliknięcia odnośnika w sposób niewrażliwy na błędy w różnych wersjach przeglądarek:

rozdzial_03/15_page_scroll/script.js (fragment)

```
$('a[href=#]').click(function() {
  $.scrollTo(0,'slow');
  return false;
});
```

Powyższa składnia zapewne wygląda nieco dziwnie, ponieważ wywołujemy metodę `scrollTo` bezpośrednio z aliasu funkcji `jQuery()`. Wtyczka działa w sposób inteligentny i wie, że w takim wywołaniu ma przewinąć cały dokument. Jeśli chcemy przewinąć jeden z elementów dokumentu, zastosujemy tradycyjne wywołanie z selektorem:

```
$('div#scrolly').scrollTo(0, 'slow').
```

Wtyczka ScrollTo ma mnóstwo opcji, a jej działanie nie ogranicza się do przewijania obiektów o wartości bezwzględne. Można przekazać wartość względną (np. `+=50px`), element DOM (strona

[8] *http://reference.sitepoint.com/css/doctypesniffing*

[9] *http://plugins.jquery.com/project/ScrollTo*

Kup książkę Poleć książkę

zostanie przewinięta do tego elementu), selektor, literał obiektu definiujący współrzędne `x` i `y` lub słowo kluczowe `max`, co spowoduje przewinięcie dokumentu do końca. Przewijać można w poziomie i w pionie, dostępne są też opcje pozwalające precyzyjnie dostosować miejsce docelowe. Informacje na temat wszystkich dostępnych opcji można znaleźć na stronie domowej wyczki[10].

Modyfikacja paska przewijania

Klient nerwowym krokiem zbliża się do Twojego biurka i ze zmartwioną miną pokazuje „absolutnie ostateczny", zatwierdzony podpisami projekt strony. „Ale te paski przewijania nie będą tak wyglądać, prawda? Znaczy, one są takie szare i brzydko wyglądają".

Z reguły od rozpoczęcia projektu mija niewiele czasu, gdy klient prosi o zmianę systemowych pasków przewijania na niestandardowe, szczególnie jeśli chodzi o elementy wewnętrzne, jak przewijane panele `div`. Takie zlecenie wydaje się dość dobrze uzasadnione, ale należy zdać sobie sprawę z pewnych implikacji związanych z użytecznością strony.

Użytkownicy stron WWW mają pewne nawyki, przyzwyczajenia, które ułatwiają im korzystanie z interfejsu. Niestandardowe implementacje mogą wprowadzić zamieszanie, szczególnie jeśli działają inaczej od standardowych elementów, które zastępują. Ignorowanie przyzwyczajeń użytkowników może prowadzić do ich frustracji, dlatego tego typu modyfikacje elementów interfejsu użytkownika należy przeprowadzać ze szczególną ostrożnością.

Mimo to warto mieć świadomość, że celnie zmodyfikowany element interfejsu może stać się kluczowym elementem wyróżniającym stronę wśród konkurencji. Z tego powodu zawsze należy rozważyć wady i zalety takich modyfikacji z punktu widzenia użytkownika końcowego. W naszym przypadku: klient nasz pan. Robimy!

Nie ma jednak potrzeby budowania paska przewijania od zera: tego typu elementy interfejsu są skomplikowane w konstrukcji, a klient nie będzie szczęśliwy, gdy okaże się, że długie godziny poświęcone na tworzenie paska przewijania poszły na marne, bo coś nie działa, jak należy. Szczególnie jeśli do dyspozycji mamy doskonałą wtyczkę, która realizuje to zadanie: jScrollPane.

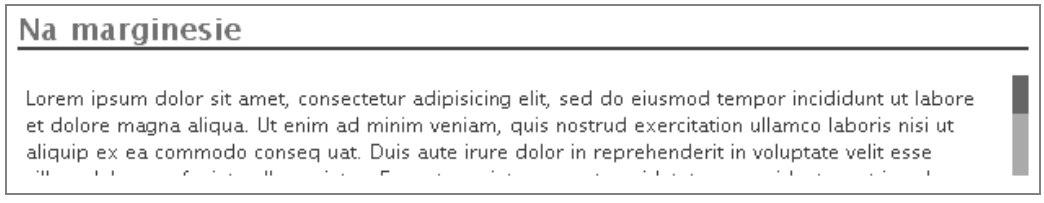
Wtyczka jScrollPane służy do implementowania niestandardowych pasków przewijania, które można umieścić w dowolnym elemencie o rozmiarze przekraczającym rozmiar kontenera. Wtyczkę tę można znaleźć w domyślnym repozytorium jQuery, ale jej bardziej aktualna wersja jest dostępna do pobrania w serwisie Google Code[11].

Do poprawnego działania wtyczki niezbędne są dwa pliki: kod JavaScript zapisany w pliku *jScrollPane-1.2.3.min.js* oraz pliku CSS *jScrollPane.css*. Plik CSS definiuje domyślne style pasków i stanowi dobry punkt wyjścia do implementacji własnego stylu. Wystarczy rozszerzyć lub zmodyfikować te style, zmieniając kolory i grafiki, żeby w prosty sposób stworzyć własny motyw wizualny. W naszych przykładach wykorzystamy domyślne style: estetyczny szary kolor, który pasuje do większości stron WWW (co widać na rysunku 3.6).

[10] *http://flesler.blogspot.com/2007/10/jqueryscrollto.html*

[11] *http://code.google.com/p/jscrollpane/*

Kup książkę Poleć książkę



Rysunek 3.6. Zmodyfikowany pasek przewijania

Zmodyfikowane paski przewijania można ustawić w dowolnym elemencie strony, wystarczy wywołać na jego selektorze metodę `jScrollPane()`. Parametry w tej metodzie są opcjonalne, ale mamy ich do dyspozycji kilkanaście. Służą one do definiowania strzałek paska przewijania czy do ustawiania paska po lewej lub prawej stronie panelu. Pełną listę parametrów można znaleźć na stronie WWW wtyczki[12]. W naszym przykładzie ustawimy margines między zawartością a paskiem przewijania, szerokość paska i ukryjemy górną oraz dolną strzałkę.

rozdzial_03/16_custom_scrollbar/script.js (fragment)

```
$('#fine_print').jScrollPane({
  scrollbarWidth: 10,
  scrollbarMargin: 10,
  showArrows: false
});
```

Nasz nowy pasek przewijania wygląda i działa doskonale, ale łatwo zauważyć, że nie obsługuje przewijania za pomocą rolki myszy. To celowy zabieg, obsługa rolki została pominięta w jQuery w celu ograniczenia rozmiaru biblioteki. Istnieje wtyczka, która uzupełnia ten brak[13], a biblioteka jScrollPane została napisana z myślą o tej wtyczce. Dzięki temu do obsługi rolki wystarczy zaimportować wtyczkę obsługi rolki, a jScrollPane wykryje to automatycznie i będzie obsługiwać rolkę do przewijania paneli.

Zmiana rozmiaru

Zmiana rozmiaru może dotyczyć wielu różnych zagadnień. Pierwsza rzecz, jaka przychodzi do głowy, to zmiana rozmiaru okna przeglądarki (efekt, który często sprawia problemy programistom stron WWW). Często stosuje się też zmianę rozmiaru okien wewnątrz aplikacji oraz zmianę rozmiarów ilustracji lub innych elementów.

Biblioteka jQuery obsługuje mechanizm gromadzenia informacji o zmianie rozmiaru okna w wyniku interakcji z użytkownikiem, a dzięki jQuery UI mamy możliwość wpływania na rozmiary dowolnych elementów na stronie. Wypróbujmy to!

[12] *http://www.kelvinluck.com/assets/jquery/jScrollPane/jScrollPane.html*

[13] *http://plugins.jquery.com/project/mousewheel*

Kup książkę Poleć książkę

Zdarzenie zmiany rozmiarów

Zdarzenie `resize` jest zaimplementowane w podstawowej bibliotece jQuery. Jest odpalane po zmianie rozmiaru okna dokumentu. Istnieje kilka powodów, dla których powinniśmy obsługiwać to zdarzenie. Zanim zajmiemy się praktycznym przykładem, zobaczmy, jak ono działa:

rozdzial_03/17_resize_event/script.js (fragment)

```
$(window).resize(function() {
  alert("Zmieniłeś rozmiar okna!");
});
```

Otwórz plik *index.html* w przeglądarce i spróbuj zmienić rozmiar okna. Za każdym razem zostanie wyświetlony komunikat. Wyprowadzanie użytkowników z równowagi przy każdej próbie zmiany rozmiarów okna przeglądarki z pewnością nie przyczyni się do zwycięstwa w konkursie na najbardziej przyjazną użytkownikom aplikację WWW, zatem spróbujmy wykorzystać to zdarzenie do czegoś bardziej praktycznego.

Przełączanie układu

Jeśli pracowałeś ze stylami CSS przez jakiś czas, z pewnością zdajesz sobie sprawę, że trwa niecichnąca debata na temat tego, co jest lepsze: pływające czy sztywne układy. Z jednej strony, pływające układy wykorzystują ekran w maksymalnym stopniu, niezależnie od rozmiarów okna przeglądarki, z drugiej strony, układy sztywne pozwalają na tworzenie precyzyjnych projektów graficznych, wypozycjonowanych co do piksela, które nie będą się rozjeżdżały niezależnie od rozmiaru okna.

W przypadku serwisu StarTrackr! możemy zaoferować coś pośredniego: zaprojektować dwa osobne sztywne układy i przełączać się między nimi w zależności od rozmiarów okna.

Zaczynamy! Domyślna strona StarTrackr!, z którą pracowaliśmy dotychczas, ma skromne 650 pikseli szerokości. Naszym pierwszy zadaniem będzie napisanie stylów, które będą używane w przypadku, gdy okno będzie miało szerokość co najmniej 850 pikseli:

rozdzial_03/18_layout_switcher/wide.css

```
body #container {
  width: 850px;
}
body #container p {
  width: 650px;
}
body #header {
  background-image: url('../../css/images/header-corners-wide.png');
}
body #celebs table {
  width: 650px;
  margin-left: 5px;
}
```

Warto zwrócić uwagę na to, że na początku każdej reguły zastosowaliśmy pozornie zbędną deklarację `body`. Chodzi jednak o to, że dzięki bardziej specyficznej deklaracji selektora CSS te reguły będą miały pierwszeństwo nad bardziej ogólnymi regułami zdefiniowanymi w podstawowym arkuszu stylów.

Kup książkę Poleć książkę

W następnym kroku musimy napisać kod jQuery, który doda lub usunie nasz arkusz stylów. Wystarczy sprawdzić, czy element body ma szerokość większą niż 900px, a jeśli tak — dopisać arkusz stylów do elementu head lub usunąć go (w przeciwnym razie).

rozdzial_03/18_layout_switcher/script.js (fragment)

```
if ($('body').width() > 900) {
  $('<link rel="stylesheet" href="wide.css" type="text/css" />')
    .appendTo('head');
} else {
  $('link[href=wide.css]').remove();
}
```

Tutaj jednak natrafiamy na pewien problem. Ten kod musi być wywołany w obsłudze dwóch zdarzeń: po załadowaniu strony i po zmianie rozmiarów okna. Kusząca może wydać się opcja skopiowania identycznego kodu w dwa miejsca i pozbycia się problemu.

Oprzyj się tej pokusie! Kopiowanie kodu to **zawsze** zły pomysł. Wyobraź sobie, że po kilku miesiącach okaże się, że 900 pikseli to zła wartość graniczna i układ należy przełączać na granicy 1000 pikseli. Odszukujesz kod, zmieniasz wartość i przeładowujesz stronę. Wszystko wydaje się w porządku, ale tak nie jest, bo poprawiłeś kod tylko dla metody obsługi zdarzenia document-ready, a zapomniałeś, że identyczny kawałek kodu znajduje się w obsłudze zdarzenia resize dokumentu. Tego typu problem jest bardzo prawdopodobny — im bardziej skomplikowany staje się kod, tym większe ryzyko i tym trudniej wykryć, gdzie tkwi przyczyna błędu.

Na szczęście prawie każdy język programowania obsługuje mechanizmy pomocne w tego typu sytuacjach i JavaScript (a dzięki temu jQuery) nie jest tu wyjątkiem. Dotychczas do obsługi zdarzeń wykorzystywaliśmy funkcje anonimowe, ale tym razem stworzymy funkcję nazwaną:

rozdzial_03/18_layout_switcher/script.js (fragment)

```
$(document).ready(function() {
  stylesheetToggle();
  $(window).resize(stylesheetToggle);
});

function stylesheetToggle() {
  if ($('body').width() > 900) {
    $('<link rel="stylesheet" href="wide.css" type="text/css" />')
      .appendTo('head');
  } else {
    $('link[href=wide.css]').remove();
  }
}
```

Naszej funkcji nadaliśmy nazwę stylesheetToggle i wykorzystujemy ją dwukrotnie: raz w obsłudze zdarzenia załadowania dokumentu i raz w obsłudze zmiany rozmiaru. Należy zwrócić uwagę na to, że w deklaracji obsługi zdarzenia wystarczy przekazać **nazwę** funkcji: w tym przypadku nie deklarujemy kodu funkcji, nie należy zatem używać słowa kluczowego function ani znaków nawiasów klamrowych czy okrągłych.

Kup książkę Poleć książkę

Elementy obsługujące zmianę rozmiaru

Biblioteka jQuery UI zawiera wtyczkę Resizable, która wchodzi w skład grupy wtyczek związanych z interakcją. Wtyczka ta umożliwia dodanie uchwytu zmiany wymiarów do elementów (prawy dolny wierzchołek). Dzięki temu uchwytowi użytkownik może rozciągać i zmniejszać elementy za pomocą myszy (tak samo jak okna systemu operacyjnego). Jak większość elementów wchodzących w skład jQuery UI, wtyczka Resizable pozwala na szeroki zakres konfiguracji i jest prosta w użyciu. Jeśli wykorzystałeś pełną wersję jQuery UI, możliwości wtyczki Resizable są gotowe do użycia. Jeśli nie, wróć do konfiguratora własnej wersji jQuery UI i wybierz element Resizable. Do działania wtyczka ta potrzebuje tylko biblioteki podstawowej i motywu.

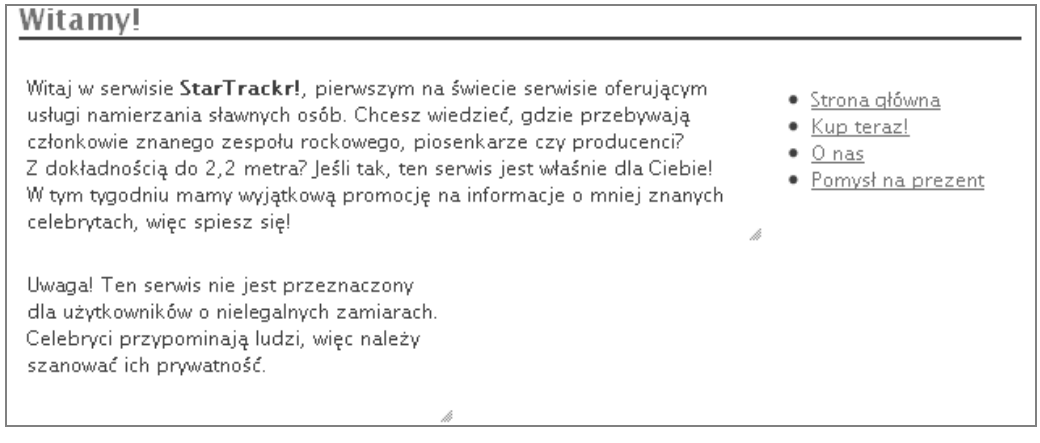
Użycie wtyczki Resizable w podstawowej formie jest bardzo proste. Po prostu wystarczy wykonać metodę `resizable()` na odpowiednim selektorze:

rozdzial_03/19_resizable_elements/script.js (fragment)

```
$('p').resizable();
```

Jeśli wykonamy ten kod w naszym serwisie StarTrackr!, uzyskamy dość niezwykły efekt: można zmieniać rozmiar każdego akapitu na stronie!

Wygląda to dość ciekawie: cała strona stała się bardzo plastyczna. W domyślnym wywołaniu wtyczka Resizable dodaje do obiektów uchwyty w prawym dolnym wierzchołku. Styl tych uchwytów jest dopasowany do aktualnego motywu jQuery UI, zatem jeśli chcesz zmienić ten wygląd, wybierz odpowiedni motyw. Domyślny wygląd uchwytów zmiany rozmiaru jest przedstawiony na rysunku 3.7.



Rysunek 3.7. Akapity z możliwością zmiany rozmiaru

Przyjrzyjmy się prostej sytuacji, w której ta funkcja bardzo się przydaje: zmianie rozmiaru pól `textarea`.

Pole textarea z możliwością zmiany rozmiaru

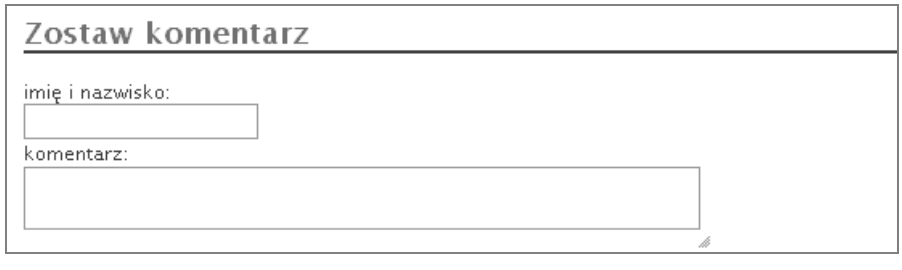
Czasem chęć stworzenia użytecznego interfejsu stoi w sprzeczności z potrzebą pięknego i zrównoważonego projektu graficznego. Dzięki jQuery możemy zająć się również tym problemem i nieco usprawnić aplikację na polu użyteczności.

Jednym z obszarów, w których forma i funkcjonalność często się ścierają, są formularze HTML. Jednym z powodów tego stanu rzeczy jest to, że użytkownicy strony WWW mają często odmienne wymagania. Załóżmy na przykład, że udostępniamy pole tekstowe (`textarea`) w formularzu opinii o stronie. Niektórzy użytkownicy nie napiszą nic, inni niewiele, a jeszcze inni napiszą bardzo dużo. Aby zachować elegancki układ formularza, można nadać niewielkie początkowe wymiary polu tekstowemu i dać użytkownikowi możliwość powiększenia go w razie potrzeby. Dzięki temu użytkownik, który lubi dużo pisać, będzie miał wrażenie, że zachęcamy go do wypowiedzi. Oto sposób, w jaki możemy zrealizować to zadanie za pomocą wtyczki Resizable wchodzącej w skład jQuery UI:

rozdzial_03/20_resizable_textarea/script.js (fragment)

```
$('textarea').resizable({
  grid : [20, 20],
  minWidth : 153,
  minHeight : 30,
  maxHeight : 220,
  containment: 'parent'
});
```

Powyższy kod powoduje, że pola tekstowe `textarea` pozwalają zmieniać swoje wymiary, tak samo jak w poprzednim przykładzie nasze akapity. Efekt jest pokazany na rysunku 3.8. W celu udoskonalenia działania efektu oraz zademonstrowania elastyczności wtyczki zastosowaliśmy dodatkowe parametry. Do dyspozycji mamy mnóstwo opcji konfiguracyjnych, z którymi można się zapoznać w dokumentacji na stronie jQuery UI[14].



Rysunek 3.8. Pole textarea z opcją zmiany rozmiaru

W opcjach zdefiniowaliśmy między innymi zakres możliwości rozciągania elementu: służą do tego właściwości `minHeight`, `minWidth` i `maxHeight`. Łatwo zauważyć, że pominęliśmy właściwość `maxWidth`, zamiast niej zastosowaliśmy parametr `containment`: służy on do określenia kontenera, który będzie ograniczał rozmiary elementu. Jako wartości tej właściwości można użyć selektora jQuery albo specjalnej wartości `parent`, która spowoduje automatyczne wybranie rodzica elementu.

[14] *http://docs.jquery.com/UI/API/1.7/Resizable*

Kup książkę Poleć książkę

Dodatkowo zastosowaliśmy atrybut `grid`, który określa skok zmiany rozmiaru. Uważamy, że taka opcja daje ciekawe wrażenie w zakresie integracji zmiany rozmiaru elementu. Parametr `grid` określa się w postaci listy dwuelementowej: rozmiar w poziomie i w pionie.

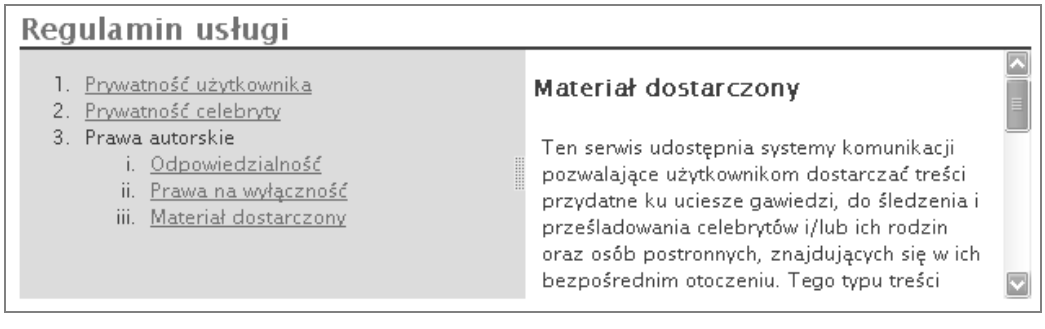
Jeszcze jeden parametr wymaga uwagi. Chodzi o `handles`. Parametr ten pozwala określić wierzchołki, w których zostaną umieszczone uchwyty zmiany rozmiaru, a w efekcie wskazuje możliwy kierunek tej zmiany. Parametr `handles` akceptuje następujące wartości: `n`, `e`, `s`, `w`, `ne`, `se`, `sw`, `nw` i `all`. Można zastosować dowolną ich liczbę, rozdzielając je przecinkiem — na przykład `{ handles : 'n', 'se'}` doda uchwyty na górnej krawędzi oraz w prawym dolnym wierzchołku.

Tego typu elementy są często stosowane w formularzach, w których można wprowadzać różną ilość tekstu.

Podział panelu

Mimo że zadbaliśmy o właściwe wyświetlanie notki prawnej, adwokat klienta nadal martwi się o to, że może on ponieść konsekwencje niewłaściwego wyeksponowania regulaminu użytkowania serwisu. Z punktu widzenia użyteczności problem polega na tym, że regulamin to bardzo długi dokument podzielony na podsekcje. Niemniej musimy go zaprezentować na stronie głównej. Być może podział panelu pomoże rozwiązać problem.

Podział panelu to taki element interfejsu, który pozwala użytkownikowi zmienić wzajemne proporcje jego elementów bez zmiany ich ogólnego wspólnego rozmiaru. Tego typu rozwiązania są powszechnie stosowane w aplikacjach typu desktop, ale wraz z rozwojem technologii stron WWW zaczęły zdobywać popularność również i w aplikacjach internetowych. Prototyp możemy zbudować za pomocą wtyczki Resizable: zasymulujemy kontener zawierający w jednym panelu spis treści, a w drugim treść regulaminu. Wygląd tego widżetu prezentuje rysunek 3.9.



Rysunek 3.9. Pionowy podział panelu

Na razie skupimy się wyłącznie na mechanizmie zmiany rozmiaru. Dynamiczne ładowanie podsekcji w panelu z regulaminem zostanie szczegółowo omówione w rozdziale 5.

Nasz dzielony panel będzie składał się z dwóch elementów `div` reprezentujących każdy z paneli składowych, zagnieżdżonych wewnątrz elementu kontenera o stałych wymiarach. Spis treści będzie elementem typu blokowego, dzięki czemu po zmianie rozmiaru układ nam się nie rozjedzie:

Kup książkę Poleć książkę

rozdzial_03/21_horizontal_pane_splitter/index.html (fragment)

```
<div id="splitter">
  <div class="pane" id="tocPane">
    <div class="inner">
    ...
    </div>
  </div>
  <div class="pane" id="contentPane">
    <div class="inner">
    ...
    </div>
  </div>
</div>
```

Teraz dodamy nieco prostych stylów do arkusza *splitter.css*. Jak widać, wysokość kontenera ustawiliśmy na sztywno, a każdy z elementów podrzędnych będzie początkowo zajmował 50% szerokości. Oczywiście proporcje początkowe można dowolnie zmienić, jeśli uznasz to za stosowne. Jeśli chcesz użyć ramki CSS (`border`), zdefiniuj jej wymiary w pikselach i zadbaj o to, żeby zgadzały się one z wymiarami kontenera.

rozdzial_03/21_horizontal_pane_splitter/splitter.css

```
#splitter{
  height:150px;
  margin-top:30px;
  margin-bottom:50px;
}
#splitter .pane {
  width:50%;
  height:100%;
  float:left;
}

#splitter h2 {
  margin-bottom:0;
  padding-bottom:0;
}

#tocPane {
  overflow: hidden;
  background:#d6dde5 url(handle.png) no-repeat right center;
}

#tocPane .inner {
  width: 300px;
}

#contentPane {
  overflow: auto;
}

#contentPane .inner {
  padding: 0 5px;
}
```

Kup książkę Poleć książkę

Następnie stworzymy nasz kod jQuery. Aby zaimplementować pionowy podział panelu, musimy spowodować, że pierwszy element będzie umożliwiał zmianę rozmiaru z uchwytem po prawej, dzięki czemu będzie go można rozciągać tylko w prawo.

Jeśli uruchomimy ten przykład wyłącznie z elementem z możliwością zmiany rozmiaru, zauważymy, że to już prawie wszystko. Jedyny problem polega na tym, że po zmianie rozmiaru lewego panelu szerokość prawego pozostaje bez zmian, zamiast dostosować się do rozmiarów pierwszego, wypełniając całą dostępną szerokość. Aby obsłużyć to wymaganie, musimy wykonać pewne obliczenia w ramach funkcji obsługi zdarzenia zmiany rozmiaru pierwszego elementu:

rozdzial_03/21_horizontal_pane_splitter/script.js (fragment)

```
$('#splitter > div:first').resizable({
  handles: 'e',
  minWidth : '100',
  maxWidth : '400',
  resize: function(){
    var remainingSpace = $(this).parent().width() - $(this).outerWidth();
    var divTwo = $(this).next();
    var divTwoWidth = remainingSpace - (divTwo.outerWidth() - divTwo.width());
    divTwo.css('width', divTwoWidth + 'px');
  }
});
```

Każda zmiana rozmiaru lewego panelu spowoduje odpowiednią zmianę rozmiaru panelu po prawej. Potrzebujemy tu odrobiny matematyki: bierzemy szerokość kontenera (która stanowi całkowitą szerokość naszego dwupanelowego widżetu) i odejmujemy od niej szerokość lewego panelu, do czego służy metoda `outerWidth()`. Metoda ta jest przydatna do sprawdzania całkowitej szerokości elementu, razem z ramką (`border`) i wypełnieniem (`padding`). Jeśli podamy jej opcjonalny parametr `true`, wartość zostanie obliczona również z marginesem (`margin`). Jak łatwo się domyślić, dostępna jest również analogiczna metoda `outerHeight()`, która zwraca wysokość elementu.

Zmienne w języku JavaScript

Przeanalizujmy ten wiersz:

```
var remainingSpace = $(this).parent().width() -$(this).outerWidth();
```

Przypisuje on wynik obliczeń nowej zmiennej o nazwie `remainingSpace`. Od tego miejsca w naszym kodzie możemy użyć nazwy `remainingSpace` wszędzie tam, gdzie potrzebujemy podstawić tę obliczoną wartość.

Kolejny wiersz jest następujący:

```
var divTwo = $(this).next();
```

Ten wiersz realizuje podobne zadanie, z tą różnicą, że nowo utworzonej zmiennej `divTwo` przypisujemy obiekt jQuery. Ta zmienna może być używana w taki sam sposób jak odpowiadająca jej selekcja jQuery.

Używanie zmiennych w ten sposób pomaga uzyskać wysoki poziom czytelności kodu, ponieważ każde wyrażenie jest tak krótkie, jak to możliwe. Dodatkowo kod może stać się wydajniejszy, ponieważ odczyt wartości ze zmiennej jest z reguły dużo szybszy niż wyliczenie tej wartości.

Kup książkę Poleć książkę

Po obliczeniu ilości miejsca pozostałego po zmianie rozmiaru jesteśmy prawie gotowi, aby ustawić szerokość drugiego elementu. Pozostaje jedynie uwzględnić jego ramkę i wypełnienie. Niestety, metoda `outerWidth` nie pozwala na definiowanie szerokości, a jedynie na jej odczyt, zatem musimy zrobić to samodzielnie.

Aby obliczyć, ile z szerokości elementu przypada na jego ramkę i wypełnienie, musimy odjąć szerokość samego elementu (wynik metody `width()`) od całkowitej szerokości (wynik metody `outerWidth()`). Po odjęciu tego wyniku otrzymujemy wartość zmiennej `remainingSpace`, która informuje nas o tym, ile miejsca zostało do wykorzystania. W tym momencie funkcja obsługi zdarzenia się kończy i mamy odpowiednio zaktualizowane wymiary naszego dwupanelowego widżetu.

Gdybyśmy zechcieli zaimplementować podobny widżet, ale z podziałem poziomym, musielibyśmy wprowadzić drobną modyfikację. Nasze elementy są umieszczone jeden nad drugim (a nie obok siebie), a metoda `resizable()` wymaga zastosowania opcji `handles` o wartości `'s'` (uchwyt na dolnej krawędzi). Kod obsługi zdarzenia zmiany wymiarów będzie niemal identyczny, z tą różnicą, że zamiast odczytywać i ustawiać szerokości elementów, będziemy używać wysokości:

rozdzial_03/22_vertical_pane_splitter/script.js (fragment)

```
$('#splitter > div:first').resizable({
  handles: 's',
  minHeight : '50',
  maxHeight : '200',
  resize: function(){
    var remainingSpace = $(this).parent().height() - $(this).outerHeight();
    var divTwo = $(this).next();
    var divTwoHeight = remainingSpace - (divTwo.outerHeight() - divTwo.height());
    divTwo.css( 'height', divTwoHeight + 'px' );
  }
});
```

Tego typu widżety pomimo swojej prostoty mogą być bardzo użyteczne. Wymagają niewielkiej ilości kodu, ale przydają się w wielu zastosowaniach. Jeśli jednak potrzebujesz bardziej skomplikowanego mechanizmu, jak wiele paneli z możliwością regulacji rozmiarów lub zagnieżdżenia, warto zastosować gotowe rozwiązanie w postaci wtyczki Splitter.

Właśnie tak się animuje, przewija i zmienia wymiary

Co za rozdział! Opanowaliśmy podstawy animacji, przewijania i zmiany rozmiarów elementów, nauczyliśmy się, w jaki sposób można tworzyć łańcuchy metod w jQuery i jak pisać zwarty kod o dużych możliwościach bez utraty jego czytelności. Stopniowo wykorzystujemy nasze umiejętności z zakresu jQuery do implementowania wspaniałych efektów. Jednak warto zauważyć w tym miejscu, że nie same efekty są istotną wiedzą, którą możesz uzyskać w tej książce, a koncepcje, które są wykorzystane do ich uzyskania.

Nawet najbardziej skomplikowane wizualnie efekty można uzyskać za pomocą prostych działań, połączonych ze sobą we właściwy sposób. To od programisty zależy, czy zechce usiąść i przemyśleć zadanie, opracować sposób jego rozwiązania i zaimplementować je w efektowny sposób.

Kup książkę Poleć książkę

Skorowidz

$

$(document).ready(), 32

A

AJAX, 166
akordeon, 132
akordeon w jQuery UI, 137
akordeony wielopoziomowe, 135
alias $(), 26
alias jQuery, 26
analiza danych XML, 190
animacje, 50
  czas trwania, duration, 65
  dynamika, easing, 60, 65
  funkcja zwrotna, callback, 65
  kolejka, 65
  koloru, 59
  skaczące, 64
  stopniowe pojawianie się, 51
  stopniowe zanikanie, 51
  wsuwanie, 51
  wysuwanie, 51
animacji,
  kolejkowanie, 301
  usuwanie, 301
animowana nawigacja, 67, 69
API, Application Programming Interface, 171
aplikacja ThemeRoller, 305
arkusz splitter.css, 84
arkusze stylów CSS, 27
atak Cross-site Scripting, 168
atrybut
  $.event.special, 298
  class, 27
  gallery.trigger, 117
  grid, 83
  href, 75
  id, 27, 34
  overflow, 224
  pageX, 120
  prototype, 278
  scrollTop, 75
  timeout, 152
  title, 147
autouzupełnianie, 208

B

biblioteka jQuery, *Patrz* jQuery
biblioteka jQuery User Interface, *Patrz* jQuery UI
biblioteka Prototype.js, 300
blokowanie zdarzeń, 297
browser sniffing, 19

C

Castledine Earle, 14
CDN, Content Delivery Network, 24
console.log, 91
cudzysłowy, 234

D

definiowanie
  selektora jQuery, 247
  własnego motywu, 305
  własnych zdarzeń, 291
dekrementacja, 106
delegacja zdarzeń, 257
dodawanie efektów, 42
dodawanie elementów, 47
dodawanie elementów do klas, 41
dodawanie metod do jQuery, 285
dokument HTML, 27
DOM, Document Object Model, 27
domknięcie, 162
drzewo DOM, 27, 124
drzewo rozwijane, 254
dymek podpowiedzi zaawansowany, 151
dymki podpowiedzi, 147
dynamiczne przetwarzanie, 253
dynamika animacji, 60
dynamika liniowa, 60
dynamika typu swing, 60

Kup książkę Poleć książkę

E

edycja wiersza, 269
efekt lightbox, 230
efekt przesuwania, 107
efekt tasowania, 109
efekt znikania, 224
element div, 34
element selectable, 245
element span, 254
element typu selectable, 244
elementy kontrolne, 266
EULA, End User License Agreement, 230

F

fałszywy nagłówek, 262
filtr, 36
  animated, 147
  checked, 196
  eq, 36, 99
  even, 36
  first, 36
  last, 36
  not, 147
  odd, 36
  radio, 213
  selected, 196, 252
filtry selektorów, 36
formularz, 192, 241
formularz logowania, 143
formularze HTML, 195
funkcja
  addHandlers(), 212
  alert(), 35, 91
  closeDialog(), 232
  innerHTML(), 47
  jQuery(), 26, 47
  openDialog(), 231
  positionLightboxImage(), 91
  reveal(), 268
  rotatePics(), 105
  setTimeout(), 106, 118, 180, 183
  setTips(), 152
  sleep(), 187
  sort(), 249, 250
  supports(), 19
  template(), 163
  trim(), 288
funkcje
  anonimowe, 52
  obsługi zdarzeń, 43, 119, 312
  opóźniające, 101
  zagnieżdżone, 268
  zwrotne, callback, 52, 240, 282
  zwrotne w stylu jQuery, 284

G

galeria motywów, 304
galeria obrazów, 115
GET, 169
Google CDN, 24
grafika sprite, 151

H

hijax, 167

I

ikona ładowania, 181
ikona ładowania globalna, 182
ikona ładowania konfiguracja, 183
implementacja
  akordeonów, 132
  dymków, 152
  kart, 138
  mechanizmu wtyczki, 279
  Suckerfish, 129
inkrementacja, 106
instalacja jQuery, 23
instrukcja if, 45
interfejs jQuery UI, *Patrz* jQuery UI

J

język JavaScript, 317
jQuery, 22, 56
jQuery UI, 19, 71
JSON, JavaScript Object Notation, 171

K

karty, 138
karty w jQuery UI, 140
katalog ColorBox, 93
katalog development-bundle, 71

Kup książkę Poleć książkę

katalog lib, 72
katalog localization, 200
klasa
  check-all, 204
  closed, 255
  lightbox, 89
  maxlength, 201
  opened, 255
  spoiler, 54
  tooltop, 150
  ui-helper-reset, 306
  ui-icon-alert, 307
  ui-selecting, 246
  ui-state-error, 307
  ui-widget-header, 307
  waiting, 128
  warning, 28
koercja typów, 317
kolejkowanie animacji, 301
komentarze, 158
konflikt nazw, 300
konsola narzędzia Firebug, 92
konstrukcja, 157
kontekst, 185

L

licznik zdarzeń, 299
lightbox, 87, 230
  animacje, 88
  ładowanie treści, 88
  przejścia, 88
  wideo, 88
lista zaznaczanych elementów, 244
listy, 243
listy wyboru, 251, 253
literał obiektu, 39

Ł

ładowanie obrazu, 225
łańcuch, 66
łańcuchy metod, 65, 206

M

mechanizm odpinania funkcji, 295
mechanizm zaznaczanych elementów, 244
menu, 121
  akordeonowe, 132
  poziome, 128, 130
  rozwijane, 122, 127, 130
metoda $.ajax(), 173, 175, 309
  flagi, 309
  funkcje zwrotne, 312
  obsługa zdarzeń, 312
  ustawienia, 310
metoda, 27
  $.ajaxSetup(), 174
  $.datepicker.setDefaults(), 217
  $.each(), 172, 178
  $.extend(), 289
  $.fn.extend(), 285
  $.get(), 175
  $.getJSON(), 171, 191
  $.getScript(), 174
  $.isFunction(), 283
  $.post(), 175, 192
  addClass(), 41
  adopt(), 228
  andSelf(), 213
  animate(), 57, 224, 301
  append(), 261
  appendTo(), 150
  attr(), 90, 253
  bind(), 207, 295
  checkScroll(), 184
  clearTimeout(), 118
  click(), 43, 128
  clone(), 164, 263
  closest(), 259
  colorbox(), 93
  createStars(), 212
  css(), 27
  data(), 114, 150, 203
  delay(), 66
  dequeue(), 65
  dialog(), 234
  die(), 170
  disableSelection(), 227
  doSlide(), 295
  draggable(), 221, 222
  droppable(), 221, 223
  empty(), 206
  end(), 201
  eq(), 105
  extend(), 285

Kup książkę Poleć książkę

metoda
fadeIn(), 179
fadeOut(), 181
filter(), 61, 134, 253
find(), 113
fixHeader(), 260, 261
GALLERY.load(), 179
gallery.offset(), 117
gallery.slide(), 118
getJSON(), 168
hide(), 43, 202
hover(), 53, 58, 67
html(), 50, 206
indexOf(), 253
innerfade(), 107
insertAfter(), 48
insertBefore(), 55
is(), 45
Jcrop(), 96
jQuery.fn.extend(), 285
jScrollPane(), 78
live(), 170
load(), 166, 169
mouseout(), 119
next(), 55, 202
nextUntil(), 213
outerHeight(), 85
outerWidth(), 85
parent(), 111
position(), 68
prevAll(), 213
preventDefault(), 125, 126, 202
previous(), 55
prevUntil(), 213
queue(), 65, 302
ready(), 207
remove(), 49, 298
removeClass(), 42
reset(), 188
resizable(), 86
scrollTop(), 75
selectable(), 245
serialize(), 192
showCelebs(), 220
slice(), 267, 274
slideDown(), 52, 131
slider(), 219
slideToggle(), 52
slideUp(), 52, 303
sortable(), 226
stop(), 65
stopPropagation(), 126
substring(), 229
swap(), 252
TABLE.formwork(), 270
teardown(), 298
text(), 50
toggle(), 51
toggleClass(), 256
trigger(), 207
trim(), 206
unbind(), 295
validate(), 199
metody kontroli kart, 142
metody pomocnicze w jQuery, 247
metody z prefiksem $., 286
metody zdarzeń, 315
minimalizacja kodu JavaScript, 25
modalne okno dialogowe, 230
moduł Selectable jQuery UI, 247
modyfikowanie elementów, 50
modyfikowanie wartości pól, 205
motyw Lightness, 304
motyw, theme, 71
możliwości przeglądarki, 165

N

nadpisywanie funkcji, 288
nagłówek h1, 34
nagłówek tabeli, 259
narzędzia dla widżetu wyboru daty, 217
narzędzia obsługi AJAX, 173
narzędzie Firebug, 40, 92
narzędzie ThemeRoller, 306
nazwa trkr, 301
nieskończone przewijanie, 183

O

obiekt
GALLERY, 178
JavaScript, 303
jQuery, 303
SWAPLIST, 251
this, 43
TT, 152
obiekty odwzorowań, 158

Kup książkę Poleć książkę

obiekty zdarzeń, 316
obraz animacji, 225
obsługa
  błędów, 186
  IE6, 156, 305
  komunikatów, 230
  motywów, 306
  opcji, 280
  zdarzenia document-ready, 32
  zdarzeń, 125
  zdarzeń jQuery, 290
odnośnik rating-0, 235
odpalenie zdarzenia, 43
odpinanie zdarzeń, 294
okno dialogowe, 230
okno dialogowe w jQuery UI, 233
opcja
  animate, 219
  autoOpen, 234
  cache, włączenie buforowania, 142
  ctiveClass, 223
  draggable, 235
  event, typ zdarzenia, 142
  fx, efekt specjalny, 142
  helper, 223
  minSize, 97
  setSelect, 97
  spinner, komunikat ładowania, 142
opcje dynamiki, 61
opcje dynamiki animacji, 60
opcje kart, 141
opcje metody $.ajax(), 309
  contentType, 310
  context, 310
  data, 311
  dataType, 311
  jsonp, 311
  password, 311
  scriptCharset, 311
  timeout, 311
  type, 311
  url, 311
  username, 311
opcje metody draggable(), 223
  axis, 223
  containment, 223
  grid, 223
  hoverClass, 223
opcje metody $.support(), 312
  boxModel, 313
  changeBubbles, 313
  cssFloat, 313
  hrefNormalized, 313
  htmlSerialize, 313
  leadingWhitespace, 313
  noCloneEvent, 314
  opacity, 314
  scriptEval, 314
  style, 314
  submitBubbles, 314
  tbody, 314
opcje wtyczek, 280
  literały obiektów, 280
  proste wartości, 280
operator
  nierówności, 318
  przypisania, 318
  równości, 318
  ścisłej nierówności, 319
  ścisłej równości, 319
  trójargumentowy, 103, 213
  zaprzeczenia logicznego, 320
opóźnienie, 100
osadzanie, inline, 40

P

panel wysuwany, 145
panele, 143
paragraf, 34
parametr, 27
parametr context, 185
parametr data, 299
parametr timeout, 109
pasek postępu, 228
pasek przewijania, 77
plik
  celebs.json, 191
  Jcrop.gif, 96
  jQuery-ui-1.7.2-min.js, 71
  jScrollPane.css, 77
  jScrollPane-1.2.3.min.js, 77
  readme, 59
pływająca nawigacja, 74
pobieranie, 23
podświetlanie, 53
podział panelu, 83
podział tabeli na strony, 265

Kup książkę Poleć książkę

pokaz slajdów, 98
  kod JavaScript, 100
  prawdziwe przenikanie, 103
  przenikanie, 98, 102
  przewijana galeria, 112
  przewijanie miniatur, 110
  zanikanie, 98
pole formularza, 34
pole textarea, 82
poruszanie się, traversing, 37
POST, 169
potomek, child, 27
powiadomienia, 230
powiadomienia 1-up, 238
powiadomienia w stylu Growl, 236
powtarzanie nagłówka tabeli, 263
propagacja zdarzeń, 124
przechwytywanie zdarzeń, 43
przeciągnij i upuść, 221
przeciągnij i zniszcz, 221
przełączanie elementów, 51
przełączanie układu, 79
przenikanie, cross-fade, 98
przestrzeń nazw colorize, 296
przestrzeń nazw zdarzeń, event namespacing, 294
przestrzeń nazw, namespace, 26, 159
przewijaczka wiadomości, 107
przewijanie, 73
przewijanie dokumentu, 75
pseudoselektor
  hover, 129
  visible, 45

R

reguły filtra, 134
repozytorium wtyczek jQuery, 21
reszta z dzielenia, 105
RIA, Rich Internet Applications, 121
rodzeństwo, siblings, 28, 48
rodzic, parent, 27
rozszerzanie jQuery, 285
rozwijane menu poziome, 130

S

selekcja
  filtry, 36
  testowanie, 35
  wybór wielu elementów, 36
  zawężanie, 35
selektor, 27, 33, 37
  #disclaimer, 45
  atrybutu, 76
  CSS3, 19
  potomków, 64
  zawężanie selekcji, 35
serwis StarTrackr!, 98
Sharkie Craig, 14
siatki danych, 264
skrypt zewnętrzny, 174
słowo kluczowe function, 80
sortowanie, 226
sortowanie list, 249
standard CSS3, 19
standard XHTML, 93
style osadzone, 40
style zagnieżdżonych list elementów, 129
Subversion, 25
Suckerfish, 129
suwak z jQuery UI, 218
suwaki, 218

T

technika hijax, 167
technologia AJAX, 166
technologia GPS, 31
technologia RFID, 31
ThemeRoller, 305
tworzenie animacji, *Patrz* animacje
tworzenie elementów, 90
tworzenie motywów wizualnych, 304
tworzenie pustego obiektu, 116
tworzenie własnych selektorów, 289
tworzenie wtyczki, 278
typ boolowski, 319

U

ukrywanie elementów, 42
usuwanie elementów, 49
usuwanie elementu z klasy, 42
usuwanie kolejki, 301

W

walidacja formularzy, 196
walidacja online, 201
wartości boolowskie, 320

Kup książkę Poleć książkę

wartość względna, 58
wersje nocne, 25
widoczność elementów, 44
widżet
  akordeonu, 133
  akordeonu wielopoziomowego, 136
  CELEB, 191
  lightbox, 231
  oceny, 210
  selectable, 245
  SORTER, 286
  TABLE, 260
  wyboru daty, 215
widżety typu lightbox, 91
własność scrollHeight, 184
właściwości CSS, 37
właściwości zdarzeń, 290, 315
właściwość
  boxModel, 313
  changeBubbles, 313
  cssFloat, 313
  hrefNormalized, 313
  htmlSerialize, 313
  leadingWhitespace, 313
  lengh, 35
  noCloneEvent, 314
  opacity, 147, 165, 314
  scriptEval, 314
  style, 314
  submitBubbles, 314
  tbody, 314
  z-index, 104
wskaźnik długości tekstu, 201
wtyczka
  bgiframe, 235
  ColorBox, 92
  Cycle, 108
  DataTables, 272
  highlightOnce, 282
  Hover Intent, 131
  InnerFade, 107
  Jcrop, 95
  Resizable, 81
  ScrollTo, 112
  ThickBox, 92
  Validation, 199
wtyczki, 59, 277
  implementacja, 279
  konfiguracja, 278
  obsługa opcji., 280
wtyczki jQuery, 280
wysyłanie danych, 191
wyświetlanie elementów, 42
wywołanie, 27
  funkcja jQuery(), 27
  metoda, 27
  parametr, 27
  selektor, 27
wywołanie document-ready
  akcja ready, 32
  parametr alert(), 32
  selektor document, 32
wywoływanie kodu z opóźnieniem, 100

Z

zakres widoczności nazw, 161
zamiar zatrzymania wskaźnika myszy, 131
zaznaczanie, 252
zaznaczanie kolumn, 273
zaznaczanie sekwencji, 273
zdarzenia, 290, 315
  blokowanie, 297
  definiowanie, 291
  metody zdarzeń, 315
  odpinanie, 294
  przestrzeń nazw, 294
  własne obiekty zdarzeń, 316
  właściwości, 290, 315
zdarzenia click, 297
zdarzenia dotykowe, 297
zdarzenia niestandardowe, 296
zdarzenia touchstart, 297
zdarzenie
  ajaxComplete, 176
  ajaxError, 188
  ajaxSuccess, 176
  beforeSend, 180, 181
  blur, 203
  click, 55, 144
  complete, 176
  document-ready, 32, 43, 80
  error, 187
  event.special, 297
  focus, 203
  hover, 130, 147
  mouseout, 53, 67, 296
  mouseover, 53, 120
  multihover, 298

Kup książkę Poleć książkę

zdarzenie
  onSelect, 97
  przewijania, 73
  reveal, 293
  submit, 198
  success, 176
  zmiany rozmiarów, 79
złożenie, 289
zmiana rozmiaru okna, 78
zmienna callback, 240
zmienne w języku JavaScript, 85
znak kropki (.), 34
znak krzyżyka (#), 34
znak łącznika, 40

Ź

źródła danych, 162

Ż

żądania AJAX, 174, 175
żądanie GET, 169
żądanie POST, 169

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

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

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

JavaScript ma za sobą długą historię z okresami większej i mniejszej popularności. Sporo osób pamięta jeszcze czasy, gdy każdy szanujący się użytkownik wyłączał obsługę tego języka w przeglądarce. Te czasy minęły bezpowrotnie! Trudno dziś wyobrazić sobie strony internetowe bez technologii AJAX oraz wygodnego, dynamicznego i efektownego interfejsu użytkownika. Warto jednak sięgnąć po rozwiązanie, dzięki któremu wykorzystanie języka JavaScript będzie przyjemniejsze, zabawniejsze i co najważniejsze, bardziej wydajne.

Biblioteka jQuery odpowiada na te wszystkie potrzeby. Jest doskonałą implementacją kodu upraszczającego integrację i zwiększającego potencjał interaktywności języka JavaScript na stronach WWW. Dzięki temu w paru eleganckich liniach kodu możesz zawrzeć mnóstwo możliwości, które w czystym JavaScripcie zajęłyby kilkanaście, a może i kilkadziesiąt linii. To jednak nie wszystko, co zyskujesz, gdy korzystasz z jQuery. Co jeszcze może Cię zainteresować? Dostęp do wielu wtyczek, gotowych skryptów i wsparcia społeczności, banalnie proste wykorzystanie technologii AJAX oraz wygodna obsługa formularzy to tylko niektóre z atutów tego narzędzia. W trakcie lektury tej książki poznasz dogłębnie bibliotekę jQuery, jej atuty i pułapki. To idealna pozycja dla każdego webmastera!

WYKORZYSTAJ CAŁY POTENCJAŁ JAVASCRIPTU I JQUERY!

- Pobieranie i instalacja jQuery
- Anatomia skryptu jQuery
- Wybieranie elementów — selektory
- Ustawianie właściwości CSS
- Dodawanie, modyfikowanie, usuwanie elementów
- Animacje
- Przewijanie dokumentu, zmiana wyglądu paska przewijania
- Efektowne wyświetlanie zdjęć i pokazy slajdów
- Menu, karty, dymki i panele
- Wykorzystanie technologii AJAX
- Obsługa formatu JSON
- Formularze — obsługa, walidacja
- Tworzenie własnych wtyczek
- Przygotowywanie motywów graficznych

sitepoint

sięgnij po WIĘCEJ

KOD KORZYŚCI

Cena 57,00 zł

ISBN 978-83-246-3618-1

9 788324 636181

Nr katalogowy: 7926

Księgarnia internetowa: http://helion.pl

Zamówienia telefoniczne: 0 801 339900 0 601 339900

helion.pl księgarnia internetowa

Sprawdź najnowsze promocje: http://helion.pl/promocje
Książki najchętniej czytane: http://helion.pl/bestsellery
Zamów informacje o nowościach: http://helion.pl/nowosci

Helion

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

Informatyka w najlepszym wydaniu