

Marcin Lis

ĆWICZENIA



PRAKTYCZNE

iQuery

Zaoszczędź mnóstwo czasu
– jeszcze dziś poznaj jQuery!

Po co Ci jQuery, czyli genialny sposób na uproszczenie tworzenia witryn WWW
Ważne pola eksploatacji, czyli zapewnienie pożądanych efektów w różnych przeglądarkach
Rozszerzenia i jQuery UI, czyli przydatne funkcje i elementy doskonałej strony internetowej



Helion

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek 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.

Redaktor prowadzący: Ewelina Burska

Projekt okładki: Maciej Pasek

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie?cwjque>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe ćwiczeń są dostępne pod adresem:

<ftp://ftp.helion.pl/przyklady/cwjque.zip>

ISBN: 978-83-246-4798-9

Copyright © Helion 2013

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wstęp	5
Rozdział 1. Podstawy jQuery	7
jQuery w kodzie strony	7
Przykład prostego skryptu	9
Tajemniczy znak „\$”	12
Struktura kodu	14
jQuery a JavaScript	20
jQuery i inne biblioteki	23
Narzędzia diagnostyczne	24
Rozdział 2. Dostęp do elementów strony	27
Podstawowe możliwości wyboru	27
Wybór ze względu na hierarchię	31
Dostęp do węzłów potomnych	35
Wyszukiwanie atrybutów	36
Obsługa elementów formularzy	41
Pozostałe selektory	44
Rozdział 3. Manipulowanie węzłami DOM	47
Zawartość węzła	47
Właściwości i atrybuty	53
Używanie funkcji zwrotnych	56
Tworzenie nowej treści	60
Usuwanie i zamiana węzłów	65
Poruszanie się po drzewie DOM	67
Rozdział 4. Obsługa zdarzeń	71
Zdarzenia na stronie WWW	71
Procedury obsługi zdarzeń	73
Zdarzenia złożone jQuery	81

	Obiekt zdarzenia	84
	Propagacja zdarzeń	88
	Usuwanie procedur obsługi	91
Rozdział 5.	Efekty na stronach WWW	95
	Ukrywanie i pokazywanie elementów	95
	Efekty fade-in i fade-out	101
	Rozwijanie elementów	104
	Własne animacje (łączenie efektów)	108
	Animowane przemieszczanie elementów	112
Rozdział 6.	jQuery i Ajax	117
	Czym jest Ajax?	117
	Pobieranie treści z serwera	118
	Dynamiczne generowanie treści	123
	Metody \$.get i \$.post	126
	Monitorowanie postępu ładowania	134
	Obsługa błędów	135
	Różne formaty danych	137
Rozdział 7.	Rozszerzenia	143
	Korzystanie z rozszerzeń	143
	Nowe metody globalne jQuery	148
	Ingerencja w istniejące moduły	153
	Własne funkcje w zasięgu globalnym	154
	Funkcje operujące na kolekcjach obiektów	159
Rozdział 8.	Obsługa interfejsu za pomocą myszy	171
	Współrzędne kursora myszy	171
	jQuery UI — wygodna obsługa interfejsu	176
	Przesuwanie i modyfikacja rozmiaru elementów	178
	Zaznaczanie i zmiana kolejności elementów	183
	Efekty drag & drop	189
Rozdział 9.	Widżety jQuery UI	193
	Przyciski	193
	Pobieranie daty	198
	Automatyczne uzupełnianie pól	202
	Okna dialogowe	206
	Zakładki	210



Efekty na stronach WWW

Ukrywanie i pokazywanie elementów

Na stronie WWW często używa się rozmaitych efektów, które pozwalają na uatrakcyjnienie witryny, ale umożliwiają też uzyskanie różnych funkcjonalności. Przydatne jest na przykład ukrywanie i pokazywanie elementów uzależnione od jakiegoś zdarzenia, jak choćby kliknięcia. Służą do tego metody `show` (pokazywanie) oraz `hide` (ukrywanie). Każda z nich (podobnie jak w przypadku innych metod opisywanych w tym rozdziale) może przyjmować do trzech opcjonalnych argumentów. Schematy wywołań można zapisać następująco:

```
show([czas_efektu])  
show(czas_efektu[, funkcja_kończąca])  
show(czas_efektu[, funkcja_transformująca[, funkcja_kończąca])
```

Opcjonalność argumentów została zaznaczona nawiasami kwadratowymi. Wywołania dla metody `hide` są analogiczne. Funkcja kończąca to funkcja, która zostanie wywołana, gdy efekt się zakończy, natomiast funkcja transformująca to funkcja pozwalająca wpływać na szybkość etapów animacji (te argumenty nie będą używane w dalszych przykładach).

Najprostsze wywołania będą miały postać:

```
obiekt.hide();  
obiekt.show();
```

Przy czym *obiekt* to obiekt jQuery opakowujący element bądź elementy witryny wybrane za pomocą dowolnego selektora. Pierwsza instrukcja

spowoduje natychmiastowe ukrycie wskazanych elementów, natomiast druga — ich pokazanie (podobny efekt można osiągnąć, bezpośrednio przypisując odpowiednie style CSS za pomocą metod opisanych w poprzednich rozdziałach).

ĆWICZENIE

5.1 Ukrywanie i wyświetlanie elementów w odpowiedzi na kliknięcie

Do strony z listingu 4.1 (rozdział 4.) dodaj dwa przyciski oraz napisz skrypt, dzięki któremu za pomocą tych przycisków będzie można ukrywać i odsłaniać treść wszystkich rozdziałów (tytuły rozdziałów powinny być przy tym zawsze widoczne). Aktywny powinien pozostawiać tylko ten przycisk, którego akcja może być w danej chwili wykonana.

Przyciski można umieścić na przystosowanej do tego warstwie `div-options`. Przyjmie ona zatem następującą postać:

```
<div id="div-options">
  <button id='showChapters'>Pokaż rozdziały</button>
  <button id='hideChapters'>Ukryj rozdziały</button>
</div>
```

Wszystkie rozdziały powinny też mieć przypisaną klasę `rozdzial`. Skrypt wykonujący to zadanie przyjmie poniższą postać:

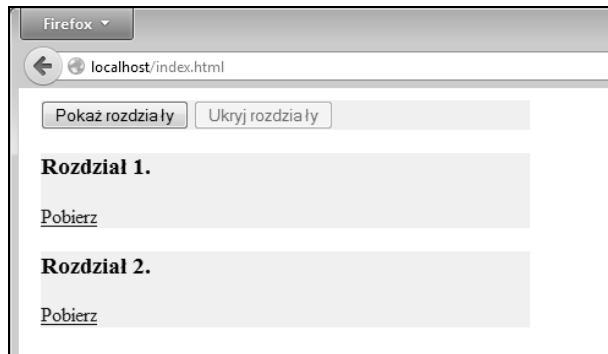
```
<script type="text/javascript">
$(document).ready(function(){
  $('.rozdzial p').hide();
  $('#showChapters').prop('disabled', false);
  $('#hideChapters').prop('disabled', true);
  $('#showChapters').click(function(){
    $('.rozdzial p').show();
    $('#showChapters').prop('disabled', true);
    $('#hideChapters').prop('disabled', false);
  });
  $('#hideChapters').click(function(){
    $('.rozdzial p').hide();
    $('#showChapters').prop('disabled', false);
    $('#hideChapters').prop('disabled', true);
  });
});
</script>
```

Selektor wybierający treść rozdziałów to `.rozdzial p`, a zatem są to wszystkie akapity zawarte w elementach o klasie `rozdzial`. Dzięki temu po ukryciu treści rozdziałów na stronie nadal pozostaną ich tytuły oraz odnośniki pozwalające na pobranie zawartości (rysunek 5.1). Treść jest ukrywana na początku kodu dzięki wywołaniu metody `hide`:

```
$('.rozdzial p').hide();
```

Rysunek 5.1.

Wygląd strony
z ćwiczenia 5.1
po ukryciu treści
rozdziałów



Następnie ustalany jest początkowy stan przycisków. Skoro rozdziały zostały ukryte, przycisk *Pokaż rozdziały* (`showChapters`) powinien być aktywny, a przycisk *Ukryj rozdziały* (`hideChapters`) — nieaktywny. Odpowiednie zmiany są więc wprowadzane za pomocą metody `prop`.

Oba przyciski otrzymały procedury obsługi zdarzenia `click` (na takiej samej zasadzie jak opisywana w rozdziale 4.). Po kliknięciu przycisku *Pokaż rozdziały* dla wszystkich elementów wybranych przez selektor `.rozdzial p` (każdy akapit w każdym rozdziale) wykonywana jest metoda `show`; jednocześnie właściwość `disabled` przycisku `showChapters` otrzymuje wartość `true` (przycisk jest wyłączany), a właściwość `disabled` przycisku `hideChapters` otrzymuje wartość `false` (przycisk jest włączany).

Z kolei po kliknięciu przycisku *Ukryj rozdziały* wykonywane są czynności odwrotne: wywołanie metody `hide`, przypisanie wartości `false` właściwości `disabled` przycisku `showChapters` oraz przypisanie wartości `true` właściwości `disabled` przycisku `hideChapters`.

Oprócz opisanych wyżej metod do ukrywania i pokazywania elementów można też użyć metody `toggle`. Jest ona swoistym połączeniem `show` i `hide`. Wywołanie `toggle` zmienia bowiem stan elementu na przeciwny,

tn. jeśli był ukryty, zostanie pokazany, a jeżeli był widoczny, zostanie ukryty. Metoda ta przyjmuje takie same argumenty jak omówione na początku rozdziału. Proste zastosowanie `toggle` zostało pokazane w ćwiczeniu 5.2.

Ć W I C Z E N I E

5.2 Przełączanie widoczności

Napisz skrypt, który na stronie z listingu 4.1 pozwoli na wyświetlanie i ukrywanie treści każdego z rozdziałów osobno. Zmiana stanu powinna się odbywać po kliknięciu w dowolnym miejscu warstwy z danym rozdziałem.

Ponieważ widoczność każdego rozdziału ma się zmieniać po każdym kliknięciu, najwygodniejsze będzie użycie metody `toggle` (nie trzeba będzie wtedy badać aktualnego stanu rozdziału). Przełączanie widoczności ma być także niezależne dla każdego rozdziału, zatem każda warstwa o klasie `rozdzial` otrzyma swoją własną procedurę obsługi kliknięcia. Warstwy zostaną więc wybrane dzięki selektorowi `.rozdzial`, a kod skryptu przyjmie następującą postać:

```
<script type="text/javascript">
$(document).ready(function(){
    $(' .rozdzial p').hide();
    $(' .rozdzial').click(function(){
        $('p', this).toggle();
        //lub
        //$ (this).children('p').toggle();
    });
});
</script>
```

Na początku za pomocą metody `hide` ukrywane są wszystkie akapity zawarte w elementach o klasie `p` (analogicznie do przypadku przedstawionego w poprzednim ćwiczeniu). Z ukrywania można też jednak zrezygnować, usuwając pierwszą instrukcję.

Każda warstwa o klasie `rozdzial` ma przypisaną procedurę obsługi zdarzenia `click`. Kliknięcie musi spowodować ukrycie lub pokazanie akapitów zawartych w danej warstwie. Aby je wybrać, można użyć kilku metod. Dwie z nich zostały zaprezentowane w kodzie. Pierwszy sposób to zastosowanie selektora `p` z jednoczesnym ograniczeniem kontekstu wyszukiwania do obiektu bieżącego (`this`):

```
$('p', this)
```

To oznacza, że zostaną wybrane elementy `p` znajdujące się tylko w bieżącej warstwie (i innych elementach w niej zagnieżdżonych).

Sposób drugi to zastosowanie metody `children`:

```
$(this).children('p').toggle();
```

Wybiera ona elementy potomne dla danego selektora, w tym przypadku wszystkie elementy typu `p` zawarte w klikniętej warstwie (`this`).

Każda z przedstawionych wyżej metod może być wywołana z argumentem określającym czas trwania animacji. Wtedy elementy, których dotyczy wywołanie, nie będą się pojawiały i znikwały od razu, tylko z określonym opóźnieniem. Efekty będą więc animowane. Ukrywanie i pokazywanie jest realizowane przez stopniowe zmniejszanie lub zwiększanie szerokości i wysokości danego elementu. Czas trwania animacji może być określany liczbowo lub przez użycie parametrów predefiniowanych. Podanie liczby oznacza czas w milisekundach, np. wywołanie:

```
$('#p').hide(800);
```

spowoduje animowane ukrywanie elementów typu `p` przez 800 milisekund.

Parametry predefiniowane to:

- ❑ `slow` — animacja wolna (600 milisekund),
- ❑ `fast` — animacja szybka (200 milisekund).

Podanie innego słowa niż `slow` lub `fast` powoduje użycie wartości 400, np. wywołanie `$('#p').show('medium');` jest równoznaczne z wywołaniem `$('#p').show(400);`.

Ć W I C Z E N I E

5.3 Animowane wyświetlanie akapitów

Napisz odpowiednik skryptu z ćwiczenia 5.2, w którym treść rozdziałów będzie ukrywana i pokazywana z użyciem animacji. Prędkość odkrywania elementu powinna być inna od prędkości ukrywania.

W skrypcie z powodzeniem można użyć metody `toggle` oraz takich samych selektorów jak w ćwiczeniu 5.2. Metoda `toggle` musi jednak otrzymać w postaci argumentu wartość określającą szybkość animacji,

inną w przypadku odkrywania treści i inną w przypadku ukrywania. Prędkości zostaną więc zapisane na początku skryptu w dwóch zmiennych: `showSpeed` (odkrywanie) i `hideSpeed` (ukrywanie). Powstanie też trzecia zmienna (`actualSpeed`), określająca prędkość, która ma być użyta w bieżącym wywołaniu metody `hide` lub `show`.

Po każdym kliknięciu danego rozdziału dla zawartych w nim akapitów będzie wywoływana metoda `toggle` i zostanie jej przekazana aktualna wartość zmiennej `actualSpeed`. Następnie wartość tej zmiennej zostanie zmieniona na przeciwną. Jeśli była to wartość zapisana w `showSpeed`, zmieni się na wartość zapisaną w `hideSpeed`, i odwrotnie, jeśli była to wartość zapisana w `hideSpeed`, zmieni się na wartość zapisaną w `showSpeed`. Dzięki temu każde kolejne wywołanie metody `toggle` będzie powodowało animację o innej prędkości. Kod skryptu przyjmie następującą postać:

```
<script type="text/javascript">
var showSpeed = 1000;
var hideSpeed = 500;
var actualSpeed;
$(document).ready(function(){
    $('.rozdzial p').hide();
    actualSpeed = showSpeed;
    $('.rozdzial').click(function(){
        $('p', this).toggle(actualSpeed);
        //lub
        //$('this').children('p').toggle(actualSpeed);
        if(actualSpeed == showSpeed){
            actualSpeed = hideSpeed;
        }
        else{
            actualSpeed = showSpeed;
        }
    });
});
</script>
```

Ć W I C Z E N I E

5.4 Uwzględnianie bieżącego stanu elementu

Wykonaj zadanie z ćwiczenia 5.3, używając metod `show` i `hide` zamiast `toggle`.

Jeżeli zamiast `toggle` mają być użyte metody `show` i `hide`, niezbędne jest rozpoznanie aktualnego stanu elementu. Można w tym celu użyć

dodatkowej właściwości, na przykład podobnie jak w ćwiczeniu 4.5 z rozdziału 4., jednak wygodniejsze będzie skorzystanie z selektora `:hidden` (tabela 2.3, rozdział 2.) wybierającego elementy ukryte. W połączeniu z metodą `is` pozwoli to stwierdzić, czy elementy są ukryte, czy też nie. Przykładowe wywołanie:

```
$('#p').is(':hidden')
```

zwróci wartość `true`, jeżeli elementy typu `p` (akapity) są ukryte, a wartość `false` — w przeciwnym przypadku. Kod może mieć zatem następującą postać:

```
<script type="text/javascript">
var showSpeed = 1000;
var hideSpeed = 500;
$(document).ready(function(){
  $('#.rozdzial p').hide();
  $('#.rozdzial').click(function(){
    if($('#p', this).is(':hidden')){
      $('#p', this).show(showSpeed);
      //lub
      //$($this).children('p').show(showSpeed);
    }
    else{
      $('#p', this).hide(hideSpeed);
      //lub
      //$($this).children('p').hide(hideSpeed);
    }
  });
});
</script>
```

Efekty fade-in i fade-out

Effekt pojawiania się lub znikania elementu może być osiągnięty za pomocą zmiany jego przezroczystości. Nie trzeba jednak samodzielnie modyfikować stylów CSS. W jQuery dostępne są metody `fadeIn`, `fadeOut` i `fadeToggle`. Pierwsza z nich powoduje zmianę od pełnej przezroczystości do zerowej (czyli stopniowe pojawianie się elementu w witrynie), a druga odwrotnie — od zerowej przezroczystości do pełnej (czyli stopniowe znikanie elementu). Z kolei trzecia powoduje naprzemienną zmianę stanu elementu (podobnie jak opisana wyżej metoda `toggle`). Wszystkie trzy metody mogą przyjmować takie same

parametry, jakie zostały opisane w poprzednim podrozdziale. Różnica jest taka, że pominięcie wszystkich parametrów spowoduje wykonanie animacji o średniej prędkości (400 milisekund). Przykładowe wywołania:

```
$('#img').fadeIn();
```

i

```
$('#img').fadeIn(400);
```

są zatem równoważne.

Ć W I C Z E N I E

5.5 Pojawianie się i znikanie obrazu

Umieść na stronie przycisk oraz obraz. Kliknięcie przycisku powinno spowodować pokazywanie i ukrywanie obrazu. Animacja powinna korzystać z efektu zmiany przezroczystości obrazu.

Aby umieścić na stronie przycisk i obraz, można użyć poniższego kodu:

```
<div>
<button id="button-showhideimage"></button>
<br />

</div>
```

Do definicji zostały użyte typowe znaczniki `<button>` i `<img>`. Oba elementy otrzymały identyfikatory pozwalające na odwoływanie się do nich w kodzie skryptu. W skrypcie należy obsłużyć zdarzenie `click` związane z przyciskiem. W procedurze obsługi zdarzenia trzeba sprawdzić, czy obraz jest aktualnie wyświetlany, czy też nie. Można to zrobić, korzystając z metody `is` i selektora `:hidden`, podobnie jak miało to miejsce w ćwiczeniu 5.4. Gdy obraz jest ukryty, należy go wyświetlić, stosując metodę `fadeIn`. Z kolei gdy obraz jest widoczny, trzeba go ukryć, używając metody `fadeOut`. W obu przypadkach konieczna jest również odpowiednia zmiana tekstu znajdującego się na przycisku. Pełny kod skryptu będzie zatem wyglądał następująco:

```
<script type="text/javascript">
$(document).ready(function(){
    $('#img-php5pk2').hide();
    $('#button-showhideimage').text('Pokaż obraz');
    $('#button-showhideimage').click(function(){
        if($('#img-php5pk2').is(':hidden')){
```

```
$('#img-php5pk2').fadeIn('slow');  
$('#button-showhideimage').text('Ukryj obraz');  
}  
else{  
$('#img-php5pk2').fadeOut('slow');  
$('#button-showhideimage').text('Pokaż obraz');  
}  
});  
</script>
```

ĆWICZENIE

5.6 Sterowanie szybkością efektu

Wykonaj zadanie z ćwiczenia 5.5 w taki sposób, aby użytkownik strony miał możliwość wybrania szybkości pojawiania się i znikania obrazu.

Do ustawienia szybkości animacji posłużą pola wyboru typu radio. Powinny one stanowić jedną grupę o takiej samej wartości atrybutu name, tak aby były to pola wzajemnie się wykluczające. Atrybut value każdego pola będzie zawierał wartość określającą szybkość animacji. Mogą to być określenia słowne (slow, fast i dowolne inne słowo, np. medium) lub też wartości liczbowe. Kod HTML sekcji body przyjmie zatem następującą postać:

```
<div>  
<button id="button-showhideimage"></button>  
<input type="radio" name="speed" value="slow" checked="checked" />Wolno  
<input type="radio" name="speed" value="medium" />Średnio  
<input type="radio" name="speed" value="fast" />Szybko  
<br /><br />  
  
</div>
```

Tym samym nad obrazem reprezentowanym przez znacznik znajdzie się prosty panel z opcjami, przedstawiony na rysunku 5.2.

W kodzie skryptu przyciskowi o identyfikatorze img-php5pk2 należy przypisać procedurę obsługi zdarzenia click. Ma ona wykonać podobne zadanie jak w ćwiczeniu 5.5, z tą jednak różnicą, że szybkość animacji ma być określona na podstawie tego, które z pól wyboru zostało zaznaczone. Trzeba zatem pobrać wartość przypisaną zaznaczonemu polu z grupy speed. W tym celu można użyć instrukcji:

```
var speed = $('input:radio[name=speed]:checked').val();
```

Rysunek 5.2.
Wygląd panelu
z opcjami
z ćwiczenia 5.6



Jej wykonanie spowoduje przypisanie zmiennej `speed` wartości odczytanej z atrybutu `value` zaznaczonego pola (ten sposób dostępu do zaznaczonej wartości był już wcześniej wykorzystany w ćwiczeniu 4.11 z rozdziału 4.).

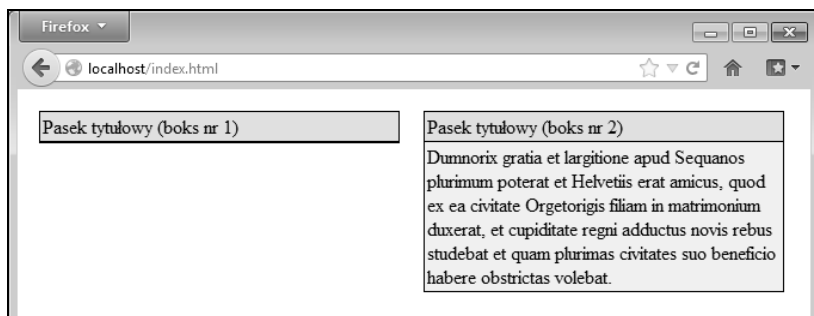
Po odczytaniu wartości aktywnego pola wyboru zmienna `speed` może być użyta jako argument metod `fadeIn` i `fadeOut`. Kod skryptu będzie zatem wyglądał następująco:

```
<script type="text/javascript">
$(document).ready(function(){
    $('#img-php5pk2').hide();
    $('#button-showhideimage').text('Pokaż obraz');
    $('#button-showhideimage').click(function(){
        var speed = $('#input:radio[name=speed]:checked').val();
        if($('#img-php5pk2').is(':hidden')){
            $('#img-php5pk2').fadeIn(speed);
            $('#button-showhideimage').text('Ukryj obraz');
        }
        else{
            $('#img-php5pk2').fadeOut(speed);
            $('#button-showhideimage').text('Pokaż obraz');
        }
    });
});
</script>
```

Rozwijanie elementów

Elementy witryny mogą być też zwijane bądź rozwijane. Ten efekt jest często stosowany w modułach (czy też boksach) zawierających różnego rodzaju treść. Moduł w postaci zwiniętej ma dostępny tylko pasek tytu-

łowy, natomiast rozwinięty udostępnia całą zawartość, tak jak zostało to pokazane na rysunku 5.3. Do animacji mogą być używane metody `slideDown` (rozwijanie), `slideUp` (zwijanie) i `slideToggle` (zmiana stanu na przeciwny). Wszystkie metody mogą przyjmować takie same zestawy argumentów, jakie zostały opisane na początku rozdziału.



Rysunek 5.3. Zwinięty i rozwinięty moduł na stronie WWW

Ć W I C Z E N I E

5.7 Moduły z rozwijaną treścią

Umieść na stronie dwa moduły (lub większą ich liczbę). Każdy moduł powinien mieć możliwość zwijania i rozwijania zawartej w nim treści.

Pojedynczy moduł może być skonstruowany za pomocą trzech warstw. Warstwa zewnętrzna opakuje cały boks, a wewnętrzne będą odzwierciedlały część nagłówkową oraz część z treścią. Każda warstwa otrzyma też odpowiedni identyfikator. Przykładowy kod HTML jednego modułu będzie więc następujący:

```
<div class="box">
  <div class="box-title">
    Pasek tytułowy (boks nr 1)
  </div>
  <div class="box-content">
    Tutaj treść modułu
  </div>
</div>
```

Na stronie należy umieścić dwa takie boksy, odpowiednio wypełniając je treścią i definiując zawartość nagłówka. Rozwijanie i zwijanie treści będzie się odbywało po kliknięciu warstwy nagłówka, a zatem wszystkie elementy strony o klasie `box-title` (selektor `.box-title`)

otrzymają procedurę obsługi zdarzenia `click`. W tej procedurze (funkcji anonimowej) trzeba sprawdzić, czy kliknięty moduł ma zwiniętą, czy rozwiniętą treść, i wykonać odpowiednio metodę `slideDown` lub `slideUp` (lub też zamiast tego użyć metody `slideToggle`).

Kluczowa jest zatem kwestia uzyskania dostępu do warstwy z treścią (`box-content`) bieżącego modułu. Nie jest to jednak skomplikowane. W funkcji anonimowej obsługującej zdarzenie `click` wskazanie `this` będzie zawierało odniesienie do warstwy z tytułem boksów (bo ta warstwa reaguje na kliknięcia). Jednym ze sposobów odnalezienia warstwy z treścią będzie więc odwołanie się do elementu nadrzędnego (będzie to warstwa `div-box`; odwołanie może być wykonane za pomocą metody `parent`) oraz odszukanie w tym elemencie warstwy potomnej o klasie `box-content` (można to zrobić za pomocą metody `find`).

A zatem pełne odwołanie do warstwy z treścią może mieć postać:

```
var boxContent = $(this).parent().find('div.box-content');
```

Po wykonaniu tej instrukcji zmienna `boxContent` będzie zawierała obiekt warstwy z treścią należącej do klikniętego modułu, opakowanej w obiekt jQuery. Można więc będzie bezpośrednio wykonywać metody `slideUp` i `slideDown`:

```
boxContent.slideDown('slow');  
boxContent.slideUp('slow');
```

Pełny kod skryptu będzie więc wyglądał następująco:

```
<script type="text/javascript">  
$(document).ready(function(){  
  $('.box-title').click(function(){  
    var boxContent = $(this).parent().find('div.box-content');  
  
    if(boxContent.is(':hidden')){  
      boxContent.slideDown('slow');  
    }  
    else{  
      boxContent.slideUp('slow');  
    }  
  });  
});  
</script>
```

W rozdziale 2. wiele ćwiczeń odnosiło się do kodu witryny zawierającej zagnieżdżone listy nienumerowane (listing 2.1). W ćwiczeniu 2.3

główne elementy listy zostały ułożone jeden obok drugiego, przez co tworzyły rodzaj poziomego menu. Korzystając ze zdarzenia `click` oraz metod `slideUp` i `slideDown` (lub `slideToggle`), można spowodować, aby poszczególne elementy listy mogły być dowolnie zwijane lub rozwijane. Sposób uzyskania takiego efektu został przedstawiony w ćwiczeniu 5.8.

Ć W I C Z E N I E

5.8 Rozwijalne menu poziome

W oparciu o kod z listingu 2.1 oraz ćwiczenia 2.3 (rozdział 2.) umieść na stronie poziome menu, którego pozycje będą mogły być zwijane i rozwijane poprzez kliknięcie.

W ćwiczeniu 2.3 styl powodujący umieszczenie głównych elementów listy w poziomie, jeden obok drugiego, był zapisywany w klasie `horizontal-list`, a klasa ta była nadawana dynamicznie elementom typu `li` zawartym w liście o identyfikatorze `listaPublikacji`. Tego zachowania nie trzeba zmieniać. Ponieważ jednak tym razem elementy będą mogły być ukrywane, a to powodowałoby zmianę szerokości wyświetlania składowych menu, do klasy najlepiej dodać atrybut ustalający szerokość elementów:

```
.horizontal-list{
  float : left;
  width : 300px;
}
```

Dodatkowo wszystkim elementom typu `li` można nadać styl `cursor : pointer`, tak aby było wiadomo, że są one „klikalne”. W skrypcie elementom typu `li` należy przypisać procedurę obsługi zdarzenia `click`. Kliknięcie powinno powodować zwinięcie lub rozwinięcie gałęzi listy znajdującej się pod klikniętym elementem. Odniesienie do klikniętego elementu znajdziemy dzięki wskazaniu `this`, a wszystkie listy podrzędne (składowe) — dzięki odszukaniu potomnych elementów typu `ul`. Przeszukiwanie można przeprowadzić dzięki metodzie `find` (podobnie jak w ćwiczeniu 5.7). Skrypt realizujący to zadanie może mieć zatem następującą postać:

```
<script type="text/javascript">
$(document).ready(function(){
  $('#listaPublikacji > li').addClass('horizontal-list');
  $('#listaPublikacji').addClass('no-bullet');
  $('li').click(function(evt){
```

```
var element = $(this).find('ul');
if(element.is(':hidden')){
    element.slideDown();
}
else{
    element.slideUp();
}
evt.stopPropagation();
});
});
</script>
```

Własne animacje (łącznie efektów)

Możliwości jQuery nie ograniczają się do udostępnienia metod przedstawionych w trzech poprzednich podrozdziałach. Z powodzeniem można definiować własne animacje. Służy do tego metoda `animate`, której podstawowe wywołanie ma postać¹:

```
animate(mapa_właściwości[, czas_efektu][, funkcja_transformująca]
➔[, funkcja_kończąca])
```

Argumentem obligatoryjnym jest jedynie mapa właściwości, pozostałe są opcjonalne i mają takie same znaczenie jak w przypadku innych metod omówionych w rozdziale. Mapa właściwości składa się z listy atrybutów CSS wraz z wartościami, które mają zostać osiągnięte. Schematycznie można to przedstawić następująco:

```
{atrybut1:wartość1, atrybut2:wartość2, ..., atrybutN:wartośćN}
```

W celu zwiększenia czytelności tę konstrukcję można podzielić na kilka wierszy:

```
{atrybut1:wartość1,
  atrybut2:wartość2,
  ...,
  atrybutN:wartośćN}
```

Należy korzystać z takich atrybutów, których wartości można przedstawić w postaci liczbowej (np. `height`, `opacity` itp.). Inne zapisy mogą być nieobsługiwane lub wymagać dodatkowych rozszerzeń (np. atrybut `color`). Zapis *wartośćN* może przyjmować kilka postaci. W najprost-

¹ Oprócz przedstawionej istnieje również wersja dwuargumentowa.

szym przypadku jest to po prostu wartość liczbową określającą wartość, do której ma dążyć atrybut w trakcie animacji. Przykładowe wywołanie:

```
animate({height:600}, 300)
```

oznacza, że wartość atrybutu `height` elementu ma w trakcie animacji dążyć do wartości 600 (pikseli) i osiągnąć tę wartość w ciągu 300 milisekund. (Pominięcie określenia czasu trwania efektu spowoduje zastosowanie wartości domyślnej wynoszącej 400).

Ponieważ często konieczne będzie podanie jednostki miary, możliwe jest użycie konstrukcji:

```
atrybut: 'wartośćjm'
```

np.:

```
animate({height:'600px'})
```

lub:

```
atrybut: wartość + 'jm'
```

np.:

```
animate({height:600 + 'px'})
```

Zamiast określonej wartości można też używać słów: `hide`, `show`, `toggle`, które pozwalają na ukrywanie i pokazywanie elementów przy minimalizacji lub maksymalizacji wartości wybranego atrybutu.

Dostępne są również zapisy:

```
atrybut: '+=wartość'
```

i

```
atrybut: '-=wartość'
```

W takich przypadkach wartość docelowa zostanie wyliczona przez dodanie do lub odjęcie od wartości bieżącej wartości występującej w wyrażeniu, np.:

```
animate({height: '+=100'})
```

Ć W I C Z E N I E

5.9 Proste użycie metody `animate`

Napisz skrypt, który pozwoli na rozwijanie i zwijanie modułów z ćwiczenia 5.7. Do animacji użyj metody `animate`.

Moduły są rozwijane i zwijane w pionie, zatem należy manipulować właściwością `height`. Nie ma jednak potrzeby używania konkretnych wartości liczbowych oznaczających wysokość warstwy z treścią danego modułu. Najłatwiej użyć po prostu słowa `toggle`. Wtedy każde kliknięcie tytułu będzie zmieniało stan boks na przeciwny, a rozwijanie będzie się odbywało do jego pierwotnej wysokości. Wystarczy zatem wywołanie:

```
boxContent.animate({height: 'toggle'});
```

Kod skryptu będzie więc bardzo prosty i przyjmie postać:

```
<script type="text/javascript">
$(document).ready(function(){
    $('.box-title').click(function(){
        var boxContent = $(this).parent().find('div.box-content');
        boxContent.animate({height: 'toggle'});
    });
});
</script>
```

Stosując mapę właściwości i metodę `animate`, można manipulować jednocześnie wieloma właściwościami wybranego elementu (elementów). Przykładowo zmianom może podlegać jednocześnie wysokość i przezroczystość. Tego typu efekt został zaprezentowany w ćwiczeniu 5.10.

Ć W I C Z E N I E

5.10 Łączenie kilku efektów w jednej animacji

Wykonaj zadanie z ćwiczenia 5.5 w taki sposób, aby przy ukrywaniu i odkrywaniu obrazu jednocześnie zmieniane były szerokość oraz przezroczystość. Wysokość obrazu ma pozostać bez zmian.

Kod HTML może pozostać taki sam jak w ćwiczeniu 5.5, zmieni się natomiast cały skrypt, za wyjątkiem dwóch pierwszych instrukcji. Ponieważ początkowym stanem jest ukrycie obrazu, najpierw należy użyć metody `hide` oraz przypisać odpowiedni tekst przyciskowi. Następnie trzeba odczytać i zapisać w zmiennych pomocniczych szerokość oraz wysokość obrazu. Posłużą do tego metody `width` (szerokość) i `height` (wysokość):

```
var imgWidth = $('#img-php5pk2').width();
var imgHeight = $('#img-php5pk2').height();
```

Dzięki temu skrypt będzie działał poprawnie dla plików graficznych o dowolnych rozmiarach i nie będzie trzeba przy tym wprowadzać poprawek w kodzie.

Ukrycie obrazu za pomocą metody `hide` nie jest jednak w tym przypadku wystarczające. Skoro bowiem pierwszą wykonywaną akcją będzie odkrywanie, należy mu przypisać początkowe atrybuty CSS o takich wartościach, jakie zostaną osiągnięte po zakończeniu animacji ukrywania, a więc pełną przezroczystość (`opacity:0`) i zerową szerokość (`width:'0px'`). Należy też zdefiniować stałą wysokość (`height:imgHeight`). Te zadania można wykonać za pomocą metody `css`:

```
$('#img-php5pk2').css({  
  opacity:0, height:imgHeight + 'px', width:'0px'  
});
```

W procedurze obsługi zdarzenia `click` przycisku `button-showhideimage` należy sprawdzić, czy obraz jest aktualnie ukryty (na takiej samej zasadzie jak w ćwiczeniu 5.5). Jeśli jest ukryty, najpierw trzeba go „odkryć”, używając metody `show`, a następnie wywołać metodę `animate` z mapą atrybutów wskazującą atrybuty `opacity` i `height`:

```
$('#img-php5pk2').animate(  
  {opacity:1, width:imgWidth},  
  'slow');
```

Po takim wywołaniu w czasie 600 milisekund (wskazuje na to argument `'slow'`) wartość atrybutu `opacity` będzie dążyła do 1, a wartość atrybutu `width` — do wartości zapisanej w zmiennej `imgWidth`. Tym samym obraz pojawi się na ekranie.

Gdy obraz jest widoczny i trzeba go ukryć, postępowanie będzie podobne. Po zakończeniu animacji konieczne będzie jednak dodatkowe użycie metody `hide`. Można ją wywołać po metodzie `animate` albo też przekazać metodzie `animate` w postaci argumentu odpowiednią funkcję. Wyglądałoby to następująco:

```
$('#img-php5pk2').animate(  
  {opacity:0, width:0},  
  'slow',  
  function(){  
    $('#img-php5pk2').hide();  
  });
```

Tym razem mapa atrybutów wskazuje, że zarówno atrybut `opacity`, jak i atrybut `width` mają dążyć do 0. Trzecim argumentem jest jednak funkcja anonimowa zawierająca wywołanie metody `hide` na rzecz obiektu

wybranego selektorem `img-php5pk2`. Ta funkcja zostanie wywołana po zakończeniu animacji, co spowoduje ukrycie obrazu.

Pełny kod skryptu realizującego zadanie postawione w ćwiczeniu będzie miał zatem poniższą postać:

```
<script type="text/javascript">
$(document).ready(function(){
  $('#img-php5pk2').hide();
  $('#button-showhideimage').text('Pokaż obraz');

  var imgWidth = $('#img-php5pk2').width();
  var imgHeight = $('#img-php5pk2').height();

  $('#img-php5pk2').css({
    opacity:0, height:imgHeight + 'px', width:'0px'
  });

  $('#button-showhideimage').click(function(){
    if($('#img-php5pk2').is(':hidden')){
      $('#img-php5pk2').show();
      $('#img-php5pk2').animate(
        {opacity:1, width:imgWidth},
        'slow');
      $('#button-showhideimage').text('Ukryj obraz');
    }
    else{
      $('#img-php5pk2').animate(
        {opacity:0, width:0},
        'slow',
        function(){
          $('#img-php5pk2').hide();
        });
      $('#button-showhideimage').text('Pokaż obraz');
    }
  });
});
</script>
```

Animowane przemieszczanie elementów

Ponieważ metoda `animate` pozwala na automatyczną zmianę wszelkich właściwości, które mogą być reprezentowane przez wartości liczbowe, może być użyta do zmiany położenia wybranych elementów strony, na przykład modułów przedstawionych w ćwiczeniu 5.7. Wystarczy

modyfikować atrybuty CSS: `left` i `top`. Przy wykonywaniu tego typu animacji przydatne są metody pozwalające na ustalenie rozmiarów elementu:

- ❑ `width` — szerokość właściwa,
- ❑ `innerWidth` — szerokość z uwzględnieniem wypełnienia (ang. *padding*),
- ❑ `outerWidth` — szerokość z uwzględnieniem marginesu,
- ❑ `height` — wysokość właściwa,
- ❑ `innerHeight` — wysokość z uwzględnieniem wypełnienia,
- ❑ `outerHeight` — wysokość z uwzględnieniem marginesu.

Należy jedynie pamiętać, że aby zmiana właściwości `left` bądź `top` mogła odnieść skutek, element musi mieć pozycjonowanie bezwzględne (*absolute*), względne (*relative*) lub ustalone (*fixed*), nie może to natomiast być domyślne pozycjonowanie statyczne (*static*).

Ć W I C Z E N I E

5.11 Automatyczna zmiana położenia elementów

Napisz skrypt, dzięki któremu kliknięcie tytułu wybranego modułu z ćwiczenia 5.7 spowoduje animowane przesunięcie całego boksu w prawą stronę, do krawędzi okna.

Kod HTML z ćwiczenia 5.7 można pozostawić bez zmian, należy jednak wprowadzić do klasy `box` atrybut `position` zmieniający domyślny sposób pozycjonowania. Przyjmijmy pozycjonowanie bezwzględne:

```
position : absolute;
```

Spowoduje to, że po wczytaniu kodu strony do przeglądarki oba moduły znajdą się jeden nad drugim (widoczny będzie tylko moduł *boks nr 2*), bowiem w kodzie nie zostały ustalone osobne wartości atrybutów `top` i `left` (można je jednak zdefiniować — wedle uznania).

W skrypcie trzeba ustalić szerokość klikniętego boksu oraz szerokość elementu nadrzędnego (w którym boks jest zawarty; w tym przypadku będzie to szerokość dokumentu). Ponieważ przesuwany ma być cały boks, a kliknięcie dotyczy nagłówka, zostanie zastosowane odwołanie do elementu nadrzędnego za pomocą metody `parent`. Najlepiej odczytać szerokość tego elementu wraz z marginesem, a więc zastosować wywołanie:

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

Aby uzyskać szerokość elementu nadrzędnego dla całego boksu, konieczne będzie ponowne użycie metody `parent` (będzie to zatem odwołanie do elementu nadrzędnego, który jest elementem nadrzędnym dla klikniętego tytułu):

```
var areaWidth = $(this).parent().parent().width();
```

Kiedy obie szerokości są gotowe, można wykonać animację. Należy przesunąć boks w prawo, do krawędzi okna. Właściwość `left` po zakończeniu animacji powinna zatem osiągnąć wartość wynikającą z odjęcia szerokości boks od szerokości okna (dzięki temu moduł nie „wyjędzie” poza widoczną część dokumentu). Kod skryptu przyjmie zatem postać zaprezentowaną poniżej:

```
<script type="text/javascript">
$(document).ready(function(){
    $(' .box-title').click(function(){
        var width = $(this).parent().outerWidth();
        var areaWidth = $(this).parent().parent().width();
        $(this).parent().animate({left:areaWidth - width}, 1000);
    });
});
</script>
```

Ć W I C Z E N I E

5.12 Przesuwanie elementu w różnych kierunkach

Napisz skrypt, w którym pojedynczy moduł z ćwiczenia 5.7 po każdym kliknięciu (w dowolnym miejscu modułu) będzie przesuwany w inną stronę: najpierw do prawego krańca, potem w dół, następnie do lewego krańca, później w górę, ponownie w prawo itd. Będzie zatem „wędrował” w obrębie swojego elementu nadrzędnego (możesz dodać nową warstwę stanowiącą taki element).

Kody HTML i CSS dotyczące modułu należy zapożyczyć z ćwiczenia 5.7. Moduł można umieścić w dodatkowej warstwie o określonej szerokości i wysokości oraz zdefiniowanym obramowaniu. Taka warstwa będzie ograniczała pole poruszania się modułu:

```
<div id="boxes">
<!-- Tutaj definicja modułu -->
</div>
```

Prosty styl CSS dla warstwy mógłby wyglądać następująco:

```
#boxes{
  width:800px;
  height:300px;
  border : 1px solid black;
}
```

Podobnie jak w ćwiczeniu 5.11, w klasie `box` określającej styl dla modułów należy wprowadzić atrybut zmieniający domyślne pozycjonowanie. Ponieważ tym razem na kliknięcia będzie reagował cały boks, a nie tylko jego tytuł, warto też dodać do tej klasy definicję kursora:

```
cursor : pointer;
position : absolute;
```

W skrypcie zostanie zdefiniowana pozycja początkowa modułu, tak aby zawsze do niej powracał (po wykonaniu czterech przesunięć). Będą za to odpowiadały zmienne `startPosLeft` i `startPosTop`. Element zostanie zaś ustawiony na początkowej pozycji dzięki instrukcji:

```
$('.box').prop('left', startPosLeft).prop('top', startPosTop);
```

(Gdyby elementów typu `box` było więcej, wszystkie miałyby taką samą pozycję startową).

Każde kliknięcie modułu ma powodować przesunięcie w innym kierunku. Kierunek przesunięcia można by określać na podstawie odczytywanej pozycji bieżącej, jednak wygodniejszym rozwiązaniem jest użycie właściwości przechowującej identyfikator fazy ruchu, który ma być wykonany. Można przyjąć następujące identyfikatory:

- ❑ 1 — przesunięcie w prawo,
- ❑ 2 — przesunięcie w dół,
- ❑ 3 — przesunięcie w lewo,
- ❑ 4 — przesunięcie w górę.

Właściwość przechowująca te wartości będzie nosiła nazwę `phase`. Odczytu i zapisu tej właściwości można dokonać za pomocą metody `prop`.

Rozpoznanie bieżącej fazy ruchu znajdzie się w instrukcji wyboru `switch`. W fazie 1. konieczne będzie uzyskanie szerokości modułu oraz warstwy nadrzędnej, tak aby możliwe było obliczenie pozycji w poziomie (na podobnej zasadzie jak w ćwiczeniu 5.11, z uwzględnieniem pozycji początkowej). W fazie 2. w analogiczny sposób zostanie obliczona pozycja w pionie (na podstawie wysokości modułu i warstwy

nadrzędnej oraz pozycji początkowej)). W fazach 3. i 4. nie trzeba natomiast wykonywać obliczeń — wystarczy przesunąć warstwę do pozycji początkowej w poziomie (faza 3.) i w pionie (faza 4.). Ostatecznie kod skryptu przyjmie następującą postać:

```
<script type="text/javascript">
$(document).ready(function(){
    var startPosLeft = 10, startPosTop = 10;
    var speed = 1000;
    $('.box').prop('left', startPosLeft).prop('top', startPosTop);

    $('.box').click(function(evt){
        switch($(this).prop('phase')){
            case '1':
            case undefined:
                var width = $(this).outerWidth();
                var areaWidth = $(this).parent().width();
                $(this).animate({left:areaWidth - width - startPosLeft}, speed);
                $(this).prop('phase', '2');
                break;
            case '2':
                var height = $(this).outerHeight();
                var areaHeight = $(this).parent().height();
                $(this).animate({top:areaHeight - height - startPosTop}, speed);
                $(this).prop('phase', '3');
                break;
            case '3':
                $(this).animate({left:startPosLeft}, speed);
                $(this).prop('phase', '4');
                break;
            case '4':
                $(this).animate({top:startPosTop}, speed);
                $(this).prop('phase', '1');
                break;
        }
    });
});
</script>
```

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

jQuery. ĆWICZENIA PRAKTYCZNE

Z jQuery praca
stanie się prostsza!



Bez jQuery trudno dziś wyobrazić sobie szybkie tworzenie serwisów WWW. Kiedy tylko powstała ta biblioteka, jej możliwości docenili wszyscy – od projektantów amatorów po poważne koncerny, takie jak Google czy Microsoft. Dziś jQuery jest rozwijana na zasadach wolnego oprogramowania i coraz rzadziej można spotkać strony internetowe, których twórcy ignorują jej potężne zalety, takie jak łatwość dostępu do elementów witryny, manipulacja strukturą strony i węzłami DOM, spójna obsługa zdarzeń czy tworzenie animacji.

Jeśli wiesz już, jak stworzyć serwis internetowy za pomocą HTML-a i CSS, a także języka JavaScript, ale nie zetknąłeś się dotychczas z jQuery, koniecznie musisz to nadrobić. Idealnym przewodnikiem będzie dla Ciebie ta książka – dzięki serii sensownych, logicznie ułożonych ćwiczeń nauczysz się wszystkiego, co może być Ci potrzebne do efektywnego korzystania z mocy biblioteki. Dowiesz się, jak używać funkcji jQuery i selektorów, modyfikować atrybuty węzłów, stosować funkcję zwrrotną i dodawać nowe elementy do witryny. Poznasz procedury obsługi zdarzeń i sposoby nadawania określonego wyglądu stronom WWW. Zrozumiesz, jak jQuery współdziała z JavaScriptem i Ajaxem, a ponadto zorientujesz się w możliwościach biblioteki jQuery UI oraz rozszerzeń, które dodatkowo wspomogą Twoją pracę. Nie czekaj, zabierz się do ćwiczeń!

- Podstawy jQuery
- Dostęp do elementów strony
- Manipulowanie węzłami DOM
- Obsługa zdarzeń
- Efekty na stronach WWW
- jQuery i Ajax
- Rozszerzenia
- Obsługa interfejsu za pomocą myszy
- Widżety jQuery UI

helion.pl
księgarnia
internetowa



Helion

Nr katalogowy: 12291



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



Zamówienia telefoniczne:

0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/newosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIECEJ



KOD KORZYŚCI

ISBN 978-83-246-4798-9



Cena 34,90 zł

Informatyka w najlepszym wydaniu

9 788324 647989