

PHP i jQuery

Techniki zaawansowane

Wydanie II

Keith Wald
Jason Lengstorf

Tytuł oryginału: Pro PHP and jQuery, 2nd Edition

Tłumaczenie: Krzysztof Wołowski

ISBN: 978-83-283-3035-1

Original edition copyright © 2016 by Jason Lengstorf and Keith Wald.
All rights reserved.

Polish edition copyright © 2017 by HELION SA.
All rights reserved.

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 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.

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/phjqz2.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/phjqz2>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

	O autorach	9
	O recenzencie	11
Część I	Podstawy jQuery	13
Rozdział 1	Wprowadzenie do jQuery	15
	Zalety jQuery w porównaniu do JavaScriptu	15
	Biblioteki JavaScript	15
	Korzyści z używania jQuery	16
	Historia jQuery	16
	Konfiguracja środowiska testowego	17
	Instalacja Firefoksa	17
	Instalacja Firebuga	17
	Konfiguracja lokalnego środowiska testowego	18
	Odwołanie do jQuery w kodzie strony	19
	Ładowanie pobranej kopii biblioteki	19
	Ładowanie kopii biblioteki przechowywanej na zdalnym serwerze	19
	Plik testowy	19
	Funkcja jQuery	20
	Wybór elementów DOM za pomocą składni CSS	21
	Podsumowanie	32
Rozdział 2	Typowe operacje i metody jQuery	33
	Podstawy budowy skryptów jQuery	33
	Metody jQuery	33
	Poruszanie się w obrębie drzewa DOM	34
	Tworzenie i wstawianie elementów DOM	42
	Odczyt i modyfikacja atrybutów CSS	55
	Przetwarzanie zbiorów wynikowych	62
	Animacje i pozostałe efekty	65

	Obsługa zdarzeń	70
	Obsługa zapytań AJAX	75
	Podsumowanie	81
Część II	Zaawansowane techniki programowania w PHP	83
Rozdział 3	Programowanie obiektowe	85
	Czym jest programowanie obiektowe?	85
	Obiekty i klasy	85
	Różnica między obiektem a klasą	86
	Struktura klasy	86
	Definiowanie właściwości klasy	87
	Definiowanie metod klasy	88
	Mechanizm dziedziczenia w klasach	95
	Rodzaje dostępu do właściwości i metod	99
	Komentarze DocBlock	105
	Kod obiektowy a kod proceduralny	107
	Łatwość implementacji	107
	Lepsza organizacja	111
	Łatwiejsze utrzymanie kodu	111
	Podsumowanie	112
Rozdział 4	Aplikacja Kalendarz wydarzeń	113
	Planowanie aplikacji	113
	Struktura bazy danych	113
	Szkielet aplikacji	113
	Hierarchia folderów aplikacji	114
	Modyfikacja środowiska programistycznego	116
	Tworzenie kalendarza	117
	Projektowanie bazy danych	117
	Połączenie z bazą danych w klasie	119
	Klasa Calendar	120
	Właściwości klasy Calendar	121
	Konstruktor klasy Calendar	122
	Wczytywanie danych wydarzenia	128
	Generowanie kodu HTML kalendarza i wydarzeń	134
	Generowanie kodu HTML do wyświetlania pełnych opisów wydarzeń	148
	Podsumowanie	153
Rozdział 5	Formularze do tworzenia, edytowania i usuwania wydarzeń	155
	Tworzenie i edycja wydarzeń	155
	Zabezpieczenie formularza tokenem	157
	Plik z formularzem	159
	Arkusz stylów dla strony administratora	160
	Zapis nowych wydarzeń w bazie danych	162
	Plik do przetwarzania formularza	165
	Przycisk do tworzenia wydarzeń na stronie głównej kalendarza	167

Przycisk edycji na stronie ze szczegółami wydarzenia	170
Opcje administratora w metodzie displayEvent()	172
Stylizacja strony wydarzenia	173
Usuwanie wydarzeń	175
Kod HTML przycisku usuwania	176
Metoda obsługująca potwierdzenie usunięcia wydarzenia	176
Plik do obsługi formularza potwierdzenia	180
Podsumowanie	182
Rozdział 6 Kontrola uprawnień użytkownika	183
Tabela użytkowników w bazie danych	183
Plik odpowiedzialny za wyświetlanie formularza logowania	183
Klasa administratora	185
Definicja klasy	185
Budowa metody sprawdzającej dane logowania	186
Modyfikacja rdzenia aplikacji	195
Wylogowywanie z aplikacji	199
Przycisk Wyloguj	201
Metoda przetwarzająca formularz wylogowania	201
Modyfikacja rdzenia aplikacji	203
Kontrola dostępu do narzędzi administratora	204
Wyświetlanie opcji administratora	205
Ograniczenie dostępu do stron administracyjnych	209
Podsumowanie	212
Część III Zastosowanie jQuery w aplikacjach PHP	213
Rozdział 7 Optymalizacja interfejsu użytkownika za pomocą jQuery	215
Stopniowe ulepszanie aplikacji z wykorzystaniem jQuery	215
Cele stopniowego ulepszania	216
Dołączenie biblioteki jQuery	216
Plik konfiguracyjny JavaScript	216
Nowy arkusz stylów dla elementów utworzonych przez jQuery	217
Okno modalne z danymi wydarzenia	219
Dołączenie procedury obsługi zdarzenia click	220
Wstrzymanie domyślnego działania odnośnika i dodanie klasy active	220
Odczyt łańcucha zapytania za pomocą wyrażeń regularnych	221
Okno modalne	222
Pobieranie i wyświetlanie informacji o wydarzeniu przy użyciu AJAX-a	226
Przycisk Zamknij	231
Efekty specjalne przy otwieraniu i zamykaniu okna	233
Podsumowanie	240
Rozdział 8 Edycja kalendarza za pomocą AJAX-a i jQuery	241
Otwieranie formularza do tworzenia wydarzeń	241
Zapytanie AJAX wyświetlające formularz	242
Modyfikacja pliku do obsługi żądań AJAX	243
Zamykanie okna przyciskiem Anuluj	246

Zapis nowych wydarzeń w bazie danych	246
Modyfikacja pliku ajax.inc.php	249
Dodawanie wydarzeń bez odświeżania strony	251
Deserializacja danych z formularza	251
Obiekty dat	255
Dodawanie wydarzenia do kalendarza	259
Pobranie identyfikatora nowego wydarzenia	261
Edycja wydarzeń w oknie modalnym	265
Określenie wartości pola action formularza	266
Zapis identyfikatora wydarzenia	267
Usunięcie danych wydarzenia z okna	269
Duplikaty wydarzeń	270
Potwierdzenie usunięcia w oknie modalnym	274
Wyświetlenie okna potwierdzenia	274
Modyfikacja procedury obsługi wysyłania formularza	275
Usuwanie wydarzenia z kodu HTML kalendarza	279
Podsumowanie	282

Część IV Zaawansowane techniki jQuery i PHP 285

Rozdział 9 Kontrola poprawności danych formularza przy użyciu wyrażeń regularnych 287

Podstawy wyrażeń regularnych	287
Elementarna składnia wyrażeń regularnych	287
Podstawowe modyfikatory w wyrażeniach regularnych	291
Odwołania wsteczne	292
Klasy znaków	294
Granice słowa	298
Operatory powtórzenia	298
Wykrywanie początku i końca łańcucha	298
Alternatywy	299
Wyrażenia opcjonalne	299
Budowa bardziej złożonego wyrażenia regularnego	299
Kontrola poprawności daty po stronie serwera	301
Wyrażenie regularne do walidacji dat	302
Testowe dane	302
Dopasowanie formatu daty	303
Metoda odpowiedzialna za walidację w klasie Calendar	305
Wyświetlenie komunikatu o błędzie w przypadku niepoprawnej daty	307
Kontrola poprawności daty po stronie klienta	310
Nowy plik JavaScript do sprawdzania poprawności łańcucha daty	310
Dołączenie nowego pliku w stopce	311
Zabezpieczenie przed wysłaniem formularza w przypadku nieudanej walidacji	311
Podsumowanie	314

Rozdział 10	Rozszerzanie biblioteki jQuery	315
	Dodawanie funkcji	315
	Przeniesienie walidacji daty do funkcji jQuery	315
	Modyfikacja pliku stopki	318
	Modyfikacja pliku init.js	318
	Dodawanie metod do obiektu jQuery	320
	Budowa własnej wtyczki	320
	Implementacja wtyczki	326
	Podsumowanie	327
	Dodatki	329
Dodatek A	Uwagi dotyczące PHP 7	331
	Deklarowanie typów	331
	Nowe wyjątki	333
	Stałe tablicowe	333
	Rozpakowywanie argumentów	335
	Dzielenie całkowite	335
	Lukier składniowy	336
	Skorowidz	338

ROZDZIAŁ 1



Wprowadzenie do jQuery

Aby w pełni zrozumieć bibliotekę jQuery i jej zastosowanie w nowoczesnych stronach WWW, warto na chwilę spojrzeć wstecz na jej początki, na to, jakie potrzeby miała zaspokoić i jak wyglądało programowanie w JavaScriptcie, zanim się pojawiła.

W tym rozdziale przybliżymy koncepcję bibliotek JavaScript oraz cele, które starają się one realizować, a także wyjaśnimy, dlaczego większość programistów wybiera właśnie jQuery. Omówimy też kilka kluczowych kwestii; pokażemy na przykład, jak wykorzystać bibliotekę w swoich aplikacjach, i zaprezentujemy działanie rdzenia jQuery — niezwykle wydajnego mechanizmu selektorów.

Zalety jQuery w porównaniu do JavaScriptu

Język JavaScript ma bardzo zróżnicowaną reputację w społeczności twórców oprogramowania. Duża część jego składni pozornie przypomina składnię innych znanych języków, takich jak C lub Java. Ale *semantyka* kodu JavaScript bywa odmienna, co często powoduje frustrację u niewtajemniczonych (znany architekt oprogramowania Douglas Crockford mówił i pisał o tym wielokrotnie; więcej informacji na ten temat znajdziesz w internecie).

Przeglądarki komplikują proces tworzenia stron WWW jeszcze bardziej. W każdej z nich (z nielicznymi wyjątkami) interpreter JavaScript jest zaimplementowany inaczej. W rezultacie nie ma praktycznie żadnej kontroli nad przeglądarką obsługiwaną przez użytkownika końcowego ani informacji, które z wykorzystywanych przez twórcę strony funkcji ona obsługuje. Ale za sprawą społeczności programistów stron WWW sytuacja nie jest tak beznadziejna, jak by się mogło na pierwszy rzut oka wydawać.

Biblioteki JavaScript

Duże wymagania JavaScriptu oraz kwestie związane z jego obsługą przez przeglądarki były zmartwieniem programistów przez długie lata. Z czasem ci ambitniejsi rozpoczęli budowę **bibliotek** JavaScript, określanych też jako **frameworki** JavaScript.

Biblioteki te miały na celu uproszczenie korzystania z JavaScriptu i udostępnienie pełni jego potencjału zarówno nowym, jak i doświadczonym programistom dzięki łatwym w obsłudze funkcjom, które wyręczały ich w wykonywaniu rutynowych, ale żmudnych zadań. Biblioteki przydają się szczególnie w obrębie techniki AJAX (termin pochodzi od asynchronicznego JavaScriptu i XML). Wkrótce się przekonasz, że AJAX odgrywa kluczową rolę przy uelastycznianiu aplikacji WWW, kierując żądania do serwera asynchronicznie i w miarę potrzeb (często nawet bez wiedzy użytkownika).

Biblioteki JavaScript zapewniają znacznie prostszą składnię przy typowych zadaniach, co przekłada się na przyspieszenie pracy programistów i ułatwienie nauki początkującym. Eliminują też niektóre trudności związane z dostosowaniem JavaScriptu do różnych przeglądarek, wykonując za Ciebie wszystkie kontrole zgodności przy użyciu wbudowanych metod, co pozwala na oszczędności czasu podczas tworzenia kodu.

■ **Uwaga** Różnica między użyciem narzędzi AJAX biblioteki jQuery a wykorzystaniem metod JavaScript zostanie wyjaśniona w rozdziale 2.

Wybór bibliotek JavaScript jest duży. Kilka z najbardziej obecnie popularnych to Prototype (www.prototypejs.org), MooTools (<http://mootools.net>), Yahoo! UI Library (<http://yuilibrary.com/>), AngularJS (<https://angularjs.org/>) oraz Dojo (<https://dojotoolkit.org/>).

Ich rola i możliwości są różne, ale my skupimy się na najpopularniejszej bibliotece, skoncentrowanej głównie na ułatwianiu najczęstszych interakcji z przeglądarką internetową: jQuery.

Korzyści z używania jQuery

Każdy framework JavaScript ma swoje zalety. Biblioteka jQuery nie jest tu wyjątkiem — jej mocne strony to:

- mały rozmiar pliku (około 80 kB w wersji 2.1.4),
- wyjątkowo prosta składnia,
- możliwość łączenia wywołań metod kaskadowo,
- elastyczny charakter i mnogość dostępnych wtyczek,
- bardzo liczna społeczność internetowa,
- szczegółowa dokumentacja pod adresem <http://api.jquery.com>,
- opcjonalne rozszerzenia pozwalające rozbudować bibliotekę o nowe możliwości, na przykład jQueryUI.

Historia jQuery

Biblioteka jQuery, pomysł programisty Johna Resiga, została po raz pierwszy zaprezentowana na spotkaniu entuzjastów nowych technologii w Nowym Jorku w 2006 roku (więcej informacji na temat formuły takich spotkań znajdziesz pod adresem <http://barcamp.org>). Resig tłumaczy na swojej stronie internetowej, że stworzył jQuery, ponieważ nie był zadowolony z ówczesnie dostępnych bibliotek i czuł, że można je ulepszyć poprzez usunięcie bezproduktywnych elementów składni i jednocześnie dodanie uproszczonych konstrukcji dla często wykonywanych zadań (<http://ejohn.org/blog/selectors-in-javascript/>).

jQuery z miejsca stała się hitem w społeczności programistów i szybko nabierała rozpędu. Wielu z nich pomogło udoskonalić bibliotekę, co zaowocowało jej pierwszym stabilnym wydaniem (w wersji 1.0) 26 sierpnia 2006 r.

Od tego czasu jQuery doczekała się wielu kolejnych wersji (w chwili pisania tego tekstu najnowsza to 2.1.4), a także setek utworzonych przez entuzjastów wtyczek. **Wtyczka** to rozszerzenie jQuery, które nie wchodzi w skład rdzenia biblioteki. Więcej o wtyczkach jQuery i o tym, jak je tworzyć, dowiesz się w rozdziale 10.

Konfiguracja środowiska testowego

Ponieważ nie ma lepszego sposobu na zrozumienie nowego języka od wzięcia spraw w swoje ręce, na początek przygotujemy środowisko testowe, które umożliwi nam wykonanie kilku ćwiczeń wprowadzających w tematykę jQuery. Na szczęście, utworzenie takiego środowiska wymaga tylko dwóch czynności: instalacji Firefoksa oraz instalacji Firebuga.

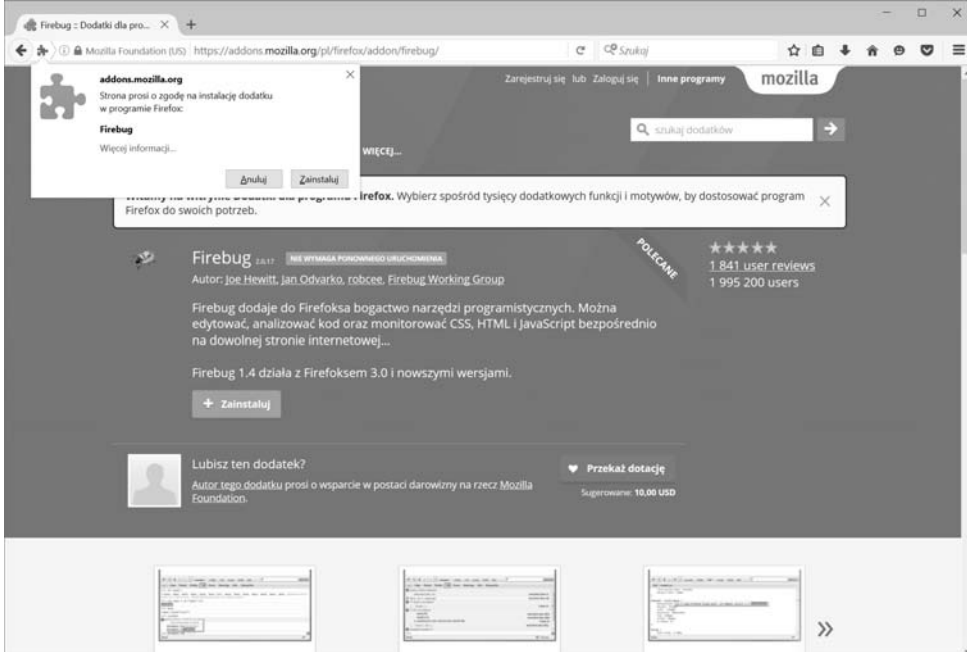
W tej książce zakładamy, że wszystkie ćwiczenia będziesz wykonywać, używając przeglądarki Firefox wraz z jej wtyczką Firebug. Największą zaletą tego tandemu jest doskonała konsola do testowania kodu JavaScript w Firebugu.

Instalacja Firefoksa

Aby pobrać i uruchomić Firefoksa, przejdź do adresu <http://firefox.pl> i pobierz najnowszą wersję przeglądarki (42.0 w chwili pisania tego tekstu), dostępną pod adresem <https://www.mozilla.org/pl/firefox/products/>.

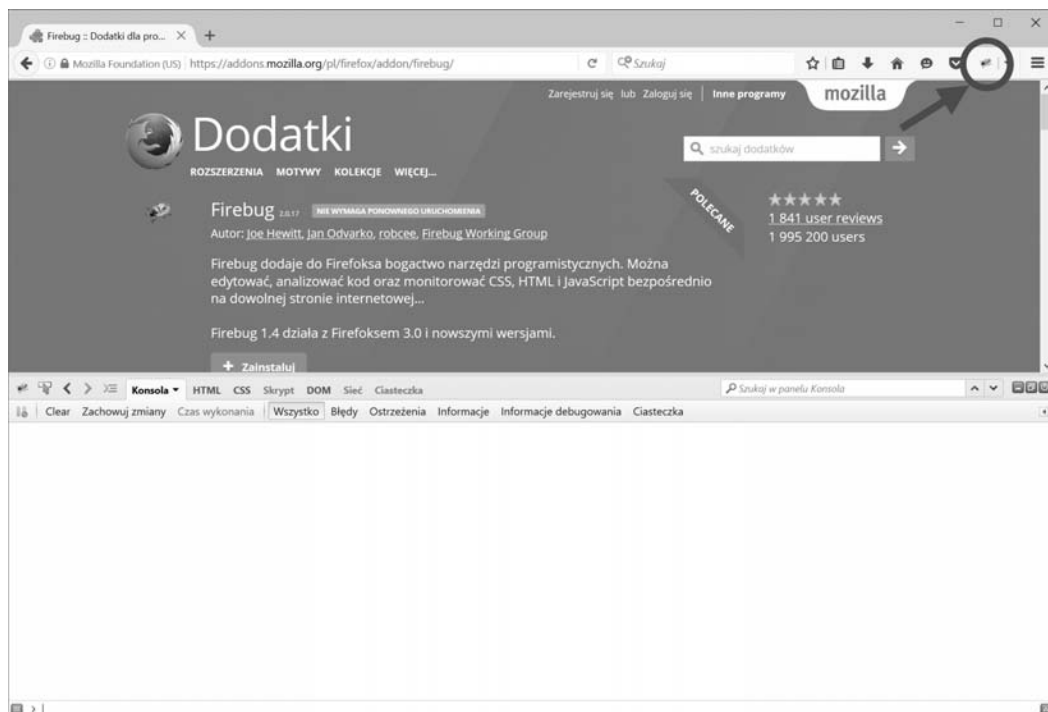
Instalacja Firebuga

Aby zainstalować Firebuga, otwórz Firefoksa i przejdź do adresu <http://getfirebug.com/downloads>, a następnie kliknij łącze **Download** przy najnowszej wersji (w chwili pisania 2.0.13). Zostaniesz przeniesiony do pozycji Firebug w katalogu *Dodatki* dla programu Firefox. Teraz kliknij przycisk **Zainstaluj**, co spowoduje wyświetlenie okna dialogowego instalacji w przeglądarce (zob. rysunek 1.1). Kliknij przycisk **Zainstaluj** i poczekaj na koniec instalacji.



Rysunek 1.1. Okno dialogowe instalacji Firebuga

Po zakończeniu instalacji na pasku stanu pojawi się ikona wyglądająca jak światełko. Służy ona do aktywacji paneli Firebug. Domyślnie widoczny jest panel konsoli (zob. rysunek 1.2).



Rysunek 1.2. Panel konsoli w dodatku Firebug

- **Uwaga** Możliwości Firebuga znacznie wykraczają poza debugowanie kodu JavaScript. Wtyczka ta jest nieocenionym narzędziem każdego projektanta stron WWW. Aby dowiedzieć się o niej więcej, odwiedź adres <http://getfirebug.com>.

Konfiguracja lokalnego środowiska testowego

Chociaż utworzenie lokalnego środowiska testowego nie jest konieczne do wykonania ćwiczeń prezentowanych w tej książce, zalecamy ten krok jako dobrą praktykę. Testowanie w środowisku lokalnym pozwala na szybsze, bezpieczniejsze programowanie i jest zazwyczaj łatwiejsze niż na zdalnym serwerze.

Instalacja XAMPP

Aby szybko i łatwo skonfigurować środowisko lokalne na komputerze, należy pobrać i zainstalować pakiet XAMPP. Wykonaj następujące kroki:

1. Przejdź do strony <http://www.apachefriends.org/pl/xampp.html> i pobierz najnowszą wersję XAMPP-a dla swojego systemu operacyjnego (w chwili pisania jest to wersja 7.0.1). W tej książce będziemy używać PHP w wersji 7.x; w rozdziale 3. dowiesz się dlaczego.

2. Otwórz pobrany plik. Na komputerze PC uruchom plik *.exe*, wybierz katalog i rozpocznij instalację. Na komputerze Mac zamontuj obraz DMG i przeciągnij folder *XAMPP* do folderu *Applications*.
3. Otwórz panel sterowania w folderze *XAMPP-a* i uruchom Apache.
4. Wpisz w pasku adresu przeglądarki **http://localhost/**, aby upewnić się, iż *XAMPP* działa. Jeśli działa, zobaczysz stronę główną *XAMPP-a*.

Oprócz wersji dla Windows i Mac istnieją też dystrybucje *XAMPP-a* dla systemów Linux i Solaris. W każdym systemie operacyjnym proces instalacji przebiega nieco inaczej. Dodatkowe informacje na temat uruchomienia lokalnego środowiska testowego na swoim komputerze znajdziesz w dokumentacji.

Odwołanie do jQuery w kodzie strony

Skorzystanie z jQuery wymaga załadowania biblioteki w dokumencie HTML, żeby skrypty miały dostęp do jej metod. Jeżeli biblioteka nie zostanie załadowana jako pierwsza, wszystkie skrypty bazujące na składni jQuery będą najprawdopodobniej powodować błędy JavaScript. Na szczęście, ładowanie jQuery jest bardzo proste i — co więcej — może się odbywać na kilka sposobów.

Ładowanie pobranej kopii biblioteki

Pierwszy sposób ładowania jQuery w projekcie polega na zapisaniu kopii biblioteki w strukturze plików projektu, a następnie odwołaniu się do niej w kodzie, podobnie jak do każdego innego pliku JavaScript:

```
<script src="js/jquery-2.1.4.min.js"></script>
```

Ładowanie kopii biblioteki przechowywanej na zdalnym serwerze

Drugi sposób to dołączenie kopii biblioteki jQuery z serwera Google. Podstawową zaletą tego rozwiązania jest duże prawdopodobieństwo, że odwiedzający stronę internetową będzie już mieć w pamięci podręcznej swojej przeglądarki kopię tego samego pliku biblioteki pobraną wcześniej z innej strony, co skróci czas ładowania.

Odwołanie do zdalnego pliku wygląda podobnie jak do pobranego:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
```

Plik testowy

Teraz, gdy środowisko testowe jest już skonfigurowane, w folderze *htdocs*, w obrębie instalacji *XAMPP-a*, należy utworzyć nowy folder *testing* zawierający plik *index.html*. Otwórz swój ulubiony edytor i umieść w nim następujący kod HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Testowanie jQuery</title>
  <meta charset="utf-8" />
</head>
<body>
```

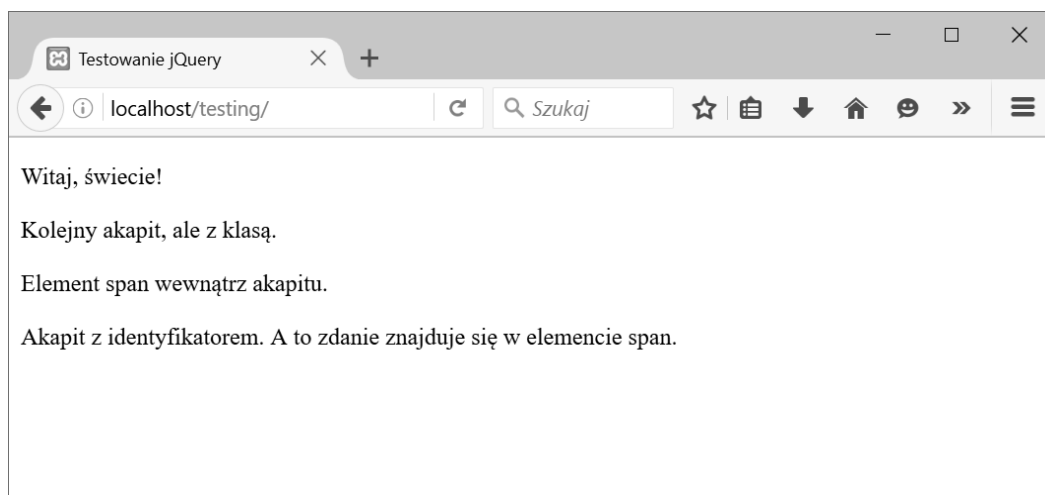
```

<p>Witaj, świecie!</p>
<p class="foo">Kolejny akapit, ale z klasą.</p>
<p><span>Element span wewnątrz akapitu.</span></p>
<p id="bar">Akapit z identyfikatorem.
  <span class="foo">A to zdanie znajduje się w elemencie span.</span>
</p>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js">
</script>
</body>
</html>

```

-
- **Uwaga** Ładowanie JavaScriptu tuż przed znacznikiem zamykającym </body> służy temu, aby skrypty nie blokowały ładowania innych elementów strony, na przykład obrazów. Zapobiega to też uruchomieniu skryptów przy nie w pełni załadowanych elementach strony, co mogłoby powodować nieoczekiwane działanie lub błędy JavaScript.
-

Zapisz ten plik i przejdź do adresu *http://localhost/testing/* w Firefoksie (zob. rysunek 1.3).



Rysunek 1.3. Plik testowy wyświetlony w Firefoksie

Plik testowy wykorzystamy do eksperymentów z podstawowymi operacjami jQuery.

Funkcja jQuery

Sercem biblioteki jQuery jest funkcja jQuery. Bez niej tworzenie kodu jQuery jest niemożliwe. Jednak w większości implementacji jQuery używa się skrótu `$()` zamiast `jQuery()`, dzięki czemu kod jest bardziej zwięzły.

Nie będziemy się tu zajmować mechanizmem działania tej funkcji. W zasadzie wystarczy tylko wiedzieć, że tworzy ona obiekt jQuery i oblicza wartość wyrażenia przekazanego jej w argumentach. Następnie ustala, jak powinna zareagować, i na tej podstawie odpowiednio się zmienia.

-
- **Ostrzeżenie** Niektóre inne biblioteki JavaScript również używają funkcji `$()`, co może powodować konflikty przy próbie korzystania z wielu bibliotek jednocześnie. jQuery udostępnia rozwiązanie tego problemu w postaci: `jQuery.noConflict()`. Więcej informacji o tej metodzie znajdziesz pod adresem <http://docs.jquery.com/Core/jquery.noConflict>.
-

Wybór elementów DOM za pomocą składni CSS

Fundamentem biblioteki jQuery jest jej niezwykle wydajny mechanizm selektorów. Pozostała część tego rozdziału poświęcona jest różnym sposobom wybierania za pomocą jQuery elementów z drzewa DOM (ang. *Document Object Model* — obiektowy model dokumentu).

-
- **Uwaga** DOM to zbiór obiektów i węzłów, które składają się na dokumenty HTML, XHTML, XML. Jest niezależny od platformy i języka. Oznacza to, że programiści mogą posługiwać się różnymi językami programowania (na przykład JavaScriptem) w różnych środowiskach (jak przeglądarki internetowe), aby uzyskać dostęp do informacji zapisanych w DOM i modyfikować je bez problemów ze zgodnością.
-

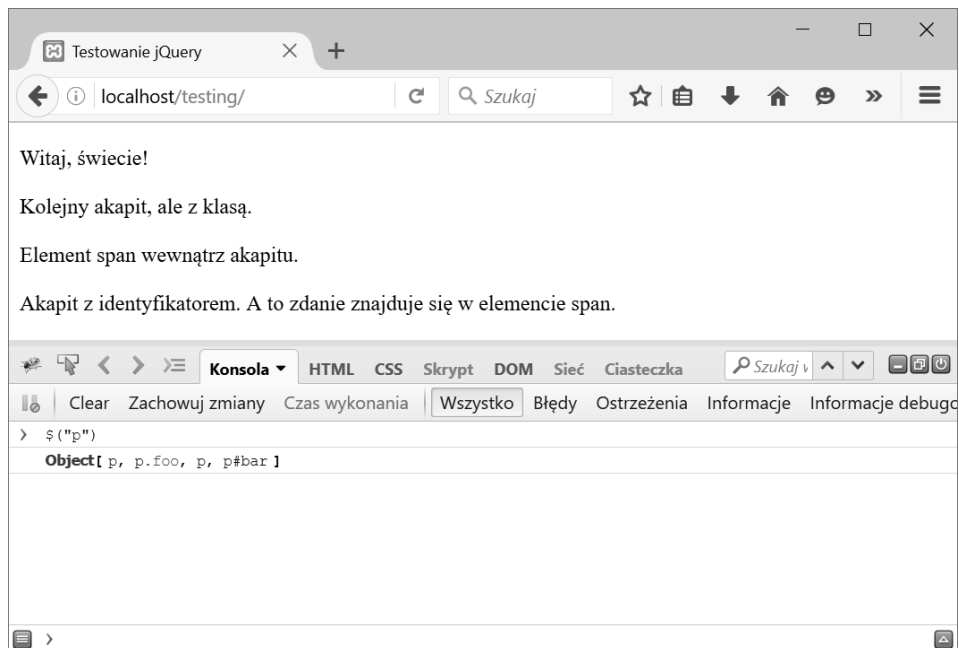
Jednym z największych wyróżników jQuery jest łatwość, z jaką programista może wybierać elementy w obrębie DOM. Pseudoselektory CSS¹ zwiększają znacznie potencjał biblioteki. Dają programiście dostęp do ściśle określonych wystąpień elementów w kodzie HTML. To bardzo duże ułatwienie, szczególnie dla tych, którzy mieli wcześniej styczność z CSS, ponieważ składnia jest prawie identyczna. W zasadzie, za pomocą tych samych konstrukcji CSS, z których zbudowane są reguły stylów, można wybierać elementy. Można przy tym korzystać z:

- podstawowych selektorów,
- hierarchii selektorów,
- filtrowania:
 - prostego,
 - na podstawie zawartości,
 - na podstawie widoczności,
 - na podstawie atrybutów,
 - na podstawie relacji rodzic-dziecko,
 - związanego z formularzami.

Podstawowe selektory

Podstawowe selektory pozwalają wybrać elementy o określonym typie znacznika, nazwie klasy lub identyfikatorze, a także dowolnej ich kombinacji. Wyświetl zawartość <http://localhost/testing/>, aktywuj okno Firebuga, a następnie kliknij zakładkę *Konsola* (zob. rysunek 1.4). Jeśli panel konsoli jest wyłączony, kliknij zakładkę *Konsola*, a następnie wybierz opcję *Włączony*. Z konsoli będziemy korzystać we wszystkich przykładach w tym rozdziale.

¹ www.w3schools.com/CSS/css_pseudo_classes.asp.



Rysunek 1.4. Konsola Firebuga po wykonaniu polecenia

-
- **Uwaga** Jeśli znasz CSS, możesz się ograniczyć do przejrzenia tej części rozdziału, ponieważ opisywane selektory zachowują się tak samo, jak ich odpowiedniki CSS.
-

Wybór elementów na podstawie typu znacznika

W celu wybrania elementu o określonym typie znacznika wystarczy jako selektora użyć nazwy znacznika (na przykład *p*, *div* czy *span*):

element

Aby wybrać wszystkie znaczniki akapitu (*<p>*) w dokumencie testowym, wpisz w wierszu u dołu konsoli:

```
$("p");
```

Naciśnij klawisz *Enter*, a kod zostanie uruchomiony. Konsola powinna teraz wyglądać tak, jak na rysunku 1.4:

```
> $("p");
Object[ p, p.foo, p, p#bar ]
```

Pierwszy wiersz świadczy o tym, że polecenie zostało wykonane, a drugi pokazuje otrzymany wynik. Dokument testowy zawiera cztery znaczniki akapitu: dwa bez klasy, identyfikatora ani atrybutów, jeden z klasą *foo* i jeden z identyfikatorem *bar* (o zastosowanej składni dowiesz się więcej w następnych rozdziałach). Po przekazaniu nazwy znacznika do funkcji jQuery wszystkie jego wystąpienia dodawane są do obiektu jQuery.

Wybór znaczników na podstawie nazwy klasy

Równie szybko, jak według typu znacznika, można wybrać elementy, którym przypisano określoną klasę lub klasy. Służy do tego nazwa klasy poprzedzona kropką (.):

.klasa

Zaznacz wszystkie elementy z klasą `foo`, uruchamiając w konsoli następujący kod:

```
$(".foo");
```

W konsoli pojawi się wynik:

```
> $(".foo");
Object[ p.foo, span.foo ]
```

Wynikowy zbiór zawiera zarówno znacznik akapitu, jak i element `span`, ponieważ oba są klasy `foo`.

Wybór elementów na podstawie identyfikatora

Aby wybrać element o określonej wartości atrybutu `id`, stosujemy taką samą konstrukcję, jak w CSS — identyfikator poprzedzony znakiem hash (#):

#id

Znajdź wszystkie elementy z identyfikatorem `bar`, wpisując:

```
$("#bar");
```

Tylko jeden akapit w dokumencie ma identyfikator `bar`, co widać w wyniku:

```
$("#bar");
Object[ p#bar ]
```

Łączenie selektorów w celu bardziej precyzyjnego wyboru

Czasami może zająć potrzeba wyizolowania spośród znaczników tylko tych, które posiadają określoną klasę. Łatwo to osiągnąć, łącząc w selektorze typ znacznika i klasę.

Aby wybrać tylko znaczniki akapitów z klasą `foo`, wydaj w konsoli instrukcję:

```
$("p.foo");
```

Wyświetlony wynik potwierdza, że element `span` został zignorowany, chociaż i on ma klasę `foo`:

```
> $("p.foo");
Object[ p.foo ]
```

Grupowanie selektorów

W sytuacji, która wymaga dostępu do wielu elementów jednocześnie, selektory można grupować.

Na przykład, jeśli chcemy wybrać każdy znacznik akapitu z klasą `foo` lub każdy element o identyfikatorze `bar`, użyjemy selektorów:

```
$("p.foo, #bar");
```

W zbiorze wynikowym znajdą się wtedy elementy, które odpowiadają *przynajmniej* jednemu selektorowi w cudzysłowie:

```
> $("p.foo,#bar");  
Object[ p.foo, p#bar ]
```

Hierarchia selektorów

Zdarza się, że możliwość wyboru na podstawie elementu, klasy czy identyfikatora nie wystarcza. Czasami potrzebujemy dostępu do elementów znajdujących się wewnątrz lub obok innego elementu albo następujących po nim, na przykład kiedy chcemy usunąć klasę `active` ze wszystkich elementów menu z wyjątkiem jednego, który właśnie został kliknięty, kiedy chcemy przechwycić wszystkie pozycje z wybranej listy nieuporządkowanej albo kiedy chcemy zmienić atrybuty elementu nadrzędnego przy wybranym elemencie formularza.

Wybór elementów potomnych

Do wybierania elementów potomnych, czyli elementów zawartych w innych elementach, stosuje się selektor przodka, spacje i selektor potomka, tak jak w przykładzie poniżej:

przodek potomek

Aby wybrać w dokumencie testowym elementy `span` będące potomkami, uruchom w konsoli Firebuga następujące polecenie:

```
$("body span");
```

Znalezione zostaną wszystkie elementy `span` wewnątrz znacznika `<body>`, mimo że znajdują się one również wewnątrz znaczników `p`:

```
> $("body span");  
Object[ span, span.foo ]
```

Wybór dzieci

Selektor dzieci można traktować jako szczególny przypadek selektora potomków. Wybiera on tylko elementy znajdujące się w hierarchii jeden poziom niżej. Aby znaleźć element dziecko, należy użyć kolejno elementu rodzica, znaku większości (`>`) i elementu dziecka:

rodzic>dziecko

Spróbuj wybrać z pliku testowego wszystkie elementy `span`, które są dziećmi elementu `body`, wpisując w konsoli:

```
$("body>span");
```

Ponieważ nie ma elementów `span` zawartych bezpośrednio w elemencie `body`, w konsoli wyświetli się następujący wynik:

```
> $("body>span");  
Object[ ]
```

Teraz zawęż zbiór elementów `span` do tych, które są dziećmi elementu `p`:

```
 $("p>span");
```

Wynik wygląda tak:

```
> $("p>span");
Object[ span, span.foo ]
```

Wybór następnego sąsiada

Czasami w skrypcie trzeba wybrać następny element drzewa DOM. Podaje się wtedy identyfikator elementu wyjściowego (w praktyce może to być dowolny selektor), po którym następuje znak plus (+) oraz selektor docelowy, tak jak w przykładzie poniżej:

poprzedni+następny

Spróbuj wpisać w konsoli następujące polecenie:

```
$(".foo+p");
```

Ponieważ istnieje tylko jeden element klasy foo, zwrócony zostanie tylko jeden element akapitu:

```
> $('p.foo+p');
Object[ p ]
```

Użyj teraz bardziej ogólnego zapytania i wybierz element akapitu następujący po jakimkolwiek innym elemencie akapitu:

```
$('p+p');
```

Kod dokumentu zawiera cztery akapity i po każdym z nich, z wyjątkiem ostatniego, jest jeszcze jeden akapit. Konsola wyświetli zatem w wyniku trzy elementy:

```
> $('p+p');
Object[ p.foo, p, p#bar ]
```

W zbiorze wynikowym znalazł się drugi, trzeci i czwarty akapit z kodu HTML.

Wybór rodzeństwa

Elementy rodzeństwo to wszystkie elementy znajdujące się w tym samym elemencie. Rodzeństwo wybiera się podobnie jak następnych sąsiadów. Różnica polega na tym, że selektor rodzeństwa dopasuje wszystkie elementy rodzeństwa następujące po elemencie wyjściowym, a nie tylko pierwszy.

Aby wybrać elementy rodzeństwa, należy podać selektor wyjściowego elementu, a po nim tyldę (~) oraz selektor rodzeństwa, tak jak w przykładzie poniżej:

brat-siostra

Aby dopasować całe rodzeństwo po akapicie z klasą foo, wykonaj następujące polecenie w konsoli:

```
$(".foo~p");
```

Wynikowy zbiór będzie wyglądać następująco:

```
> $(".foo~p");
Object[ p, p#bar ]
```

Proste filtrowanie

Filtry to jeszcze jeden wydajny sposób dostępu do elementów DOM. Zamiast polegać na typach elementów, ich klasach czy identyfikatorach, możemy je wyszukiwać na podstawie położenia, aktualnego stanu oraz innych kryteriów.

Podstawowym elementem składni filtra jest dwukropek (:), który poprzedza nazwę filtra:

```
:filtr
```

W przypadku niektórych filtrów w nawiasach może być przekazywany argument:

```
:filtr(parametr)
```

W dalszej części rozdziału omówimy najpopularniejsze i najbardziej przydatne filtry.

- **Uwaga** Aby jak najszybciej przejść do praktycznych przykładów, ograniczymy się do opisu najważniejszych filtrów. Pełną listę dostępnych filtrów można znaleźć w dokumentacji jQuery.

Wybór pierwszego lub ostatniego elementu

Do najbardziej powszechnych zastosowań filtrów należy ustalanie, czy dany element jest pierwszym albo ostatnim w zbiorze. Dzięki filtrom jest to bardzo proste zadanie. Wystarczy dołączyć do dowolnego selektora filtr `:first` lub `:last`, tak jak w przykładzie poniżej:

```
$("p:last");
```

Po uruchomieniu tego kodu w konsoli zostanie wyświetlony wynik:

```
> $("p:last");  
Object[ p#bar ]
```

Wybór elementów niepasujących do selektora

Najprostszym sposobem na znalezienie wszystkich elementów, które *nie* pasują do selektora, jest skorzystanie z filtra `:not()`. Dołącz ten filtr do selektora, podając jako argument inny selektor, a w wynikowym zbiorze znajdą się wszystkie elementy, które będą pasować do pierwszego selektora, ale nie do selektora przekazanego jako argument `:not()`.

Na przykład

```
$("p:not(.foo)");
```

zwróci następujący zbiór wynikowy:

```
> $("p:not(.foo)");  
Object[ p, p, p#bar ]
```

Wybór elementów parzystych i nieparzystych

Podobnie do `:first` i `:last`, filtry `:even` i `:odd` mają prostą składnię, zwracając, odpowiednio, parzyste i nieparzyste elementy zbioru wynikowego, co w kontekście ich nazw nie jest zaskoczeniem²:

```
$("p:odd");
```

Uruchamiając powyższy wiersz w konsoli, uzyskamy następujący wynik:

```
> $("p:odd");  
Object [ p.foo, p#bar ]
```

² *Even* to po angielsku parzysty, a *odd* — nieparzysty — *przyp. tłum.*

Wybór elementów na podstawie indeksu

Gdy chcemy wydobyć element o konkretnej pozycji w zbiorze, możemy się posłużyć filtrem `:eq()`, przekazując jako argument indeks żądanego elementu:

```
$("p:eq(3)");
```

Wynik będzie następujący:

```
> $("p:eq(3)");
Object[ p#bar ]
```

-
- **Uwaga Indeks** elementu odnosi się do jego pozycji wśród innych elementów zbioru. Liczenie w programowaniu zaczyna się od 0, więc pierwszy element ma indeks 0, drugi indeks 1 itd.
-

Filtrowanie na podstawie zawartości

Istnieją też filtry, które pozwalają wybierać elementy według ich zawartości. Takim kryterium może być na przykład określona zawartość tekstowa albo zawartość HTML.

Wybór elementów, które zawierają określony tekst

Aby wybrać tylko te elementy, które zawierają dany tekst, użyjemy filtra `:contains()`, przekazując szukany tekst jako parametr:

```
$("p:contains(Kolejny)");
```

Uruchamiając powyższy wiersz w konsoli, uzyskamy następujący wynik:

```
> $("p:contains(Kolejny)");
Object[ p.foo ]
```

-
- **Uwaga Wielkość** liter w ciągu przekazywanym do filtra `:contains()` ma znaczenie. W komentarzu pod opisem tego filtra w dokumentacji API jeden z uczestników dyskusji dodał wersję filtra `:contains()`, która nie rozróżnia wielkości liter. Więcej informacji odnośnie tego filtra znajdziesz pod adresem <http://api.jquery.com/contains-selector>.
-

Wybór elementów, które zawierają określony element

Jeśli trzeba wybrać tylko te elementy, które zawierają inny element, można skorzystać z filtra `:has()`. Działa on podobnie do `:contains()`, z tą różnicą, że zamiast ciągu znaków przyjmuje jako argument nazwę elementu:

```
$("p:has(span)");
```

Po wykonaniu tej instrukcji w konsoli zostanie wyświetlony wynik:

```
> $("p:has(span)");
Object[ p, p#bar ]
```

Jak widać, w zbiorze wynikowym znalazły się tylko elementy zawierające elementy span.

Wybór elementów, które są rodzicami

W przeciwieństwie do filtra `:empty`, filtr `:parent` spowoduje wybranie tylko tych elementów, które mają dzieci, przy czym dziećmi mogą być zarówno inne elementy, jak i węzły tekstowe.

Wybierz wszystkie akapity, które są rodzicami, wydając następujące polecenie:

```
$("p:parent");
```

Ponieważ każdy z akapitów w przykładowym dokumencie HTML zawiera tekst, a niektóre również inne elementy, w zbiorze wynikowym znajdują się wszystkie elementy p:

```
> $("p:parent");
Object[ p, p.foo, p, p#bar ]
```

Filtrowanie na podstawie widoczności

Filtry widoczności `:hidden` i `:visible` służą do wybierania, odpowiednio, ukrytych i widocznych elementów. Wybierz wszystkie widoczne akapity za pomocą polecenia:

```
$("p:visible");
```

Ponieważ żaden z elementów przykładowego dokumentu HTML nie jest aktualnie ukryty, konsola zwróci następujący zbiór wynikowy:

```
> $("p:visible");
Object[ p, p.foo, p, p#bar ]
```

Filtrowanie na podstawie atrybutów

Do wybierania elementów doskonale nadają się również atrybuty. **Atrybuty** to te składowe elementu, które dodatkowo go określają. Atrybutami są na przykład: `class`, `href`, `id` albo `title`. W przykładach poniżej posłużymy się atrybutem `class`.

-
- **Uwaga** Pamiętaj, że w środowisku produkcyjnym zaleca się stosowanie — tam, gdzie jest to możliwe — selektorów identyfikatora (`#id`) oraz klasy (`.class`) z racji ich większej wydajności. Poniższe przykłady służą tylko prezentacji możliwości filtrów atrybutów.
-

Wybór elementów na podstawie wartości atrybutu

Aby dopasować elementy, które zawierają dany atrybut i jego określoną wartość, należy ująć parę atrybut-wartość w nawiasy kwadratowe (`[]`):

```
[atribut=wartość]
```

Zaznacz wszystkie elementy z atrybutem `class` o wartości `foo`, wydając w konsoli instrukcję:

```
$("[class=foo]");
```

W wyniku operacji otrzymasz:

```
> $("[class=foo]");
Object[ p.foo, span.foo ]
```

Wybór elementów, które nie mają atrybutu lub jego określonej wartości

Aby — odwrotnie niż wcześniej — wybrać tylko te elementy, które *nie* odpowiadają parze atrybut-wartość, przed znakiem równości między atrybutem a wartością należy wstawić wykrzyknik (!):

```
[atrybut!=wartość]
```

Wybierz wszystkie akapity bez klasy foo za pomocą następującej instrukcji:

```
$("p[class!=foo]");
```

Oto wynik:

```
> $("p[class!=foo]");
Object[ p, p, p#bar ]
```

Filtrowanie na podstawie relacji rodzic-dziecko,

Filtry bazujące na relacji rodzic-dziecko w rzeczywistości są alternatywą dla filtrów :even, :odd i :eq(). Podstawowa różnica polega na tym, że w tej grupie filtrów indeksowanie zaczyna się od 1 zamiast od 0 (jak ma to miejsce w przypadku :eq()).

Wybór elementów dzieci na podstawie parzystości, indeksu lub równania

Jeden z najbardziej uniwersalnych filtrów, :nth-child(), akceptuje aż trzy różne rodzaje argumentów: słowo odd lub even, indeks oraz równanie.

Podobnie jak w przypadku pozostałych filtrów operujących na elementach dzieciach, tu też indeksowanie zaczyna się od 1 zamiast od 0 (a więc pierwszy element ma indeks 1, drugi element indeks 2 i tak dalej).

Po użyciu filtra :odd zbiór wynikowy zawierał akapity z klasą foo i identyfikatorem foo. Wybierz nieparzyste akapity za pomocą :nth-child(), aby zobaczyć, czy i tym razem będzie podobnie. Wpisz w konsoli:

```
$("p:nth-child(odd)");
```

Otrzymasz następujący wynik:

```
> $("p:nth-child(odd)");
Object[ p, p ]
```

Dane wyjściowe mogą być zaskoczeniem, ale pamiętaj, że różnica wynika z innego sposobu indeksowania elementów.

Wybór pierwszego lub ostatniego elementu

Chociaż bardzo podobne do :first i :last, :first-child i :last-child wyróżnia to, że zwrócony przez nie zbiór wynikowy może zawierać więcej niż jedno dopasowanie. Na przykład, aby znaleźć ostatni element span spośród wszystkich będących dziećmi akapitu, użyjemy:

```
$("p span:last");
```

co spowoduje wyświetlenie w konsoli wyniku:

```
> $("p span:last");
Object[ span.foo ]
```

Jeśli jednak chcemy znaleźć *wszystkie* elementy span będące ostatnim dzieckiem akapitu, skorzystamy z filtra `:last-child`:

```
$("p span:last-child");
```

Tutaj punktem odniesienia jest nie drzewo DOM, ale każdy rodzic, dlatego uzyskujemy inny wynik:

```
> $("p span:last-child");  
Object[ span, span.foo ]
```

Filtry związane z formularzami

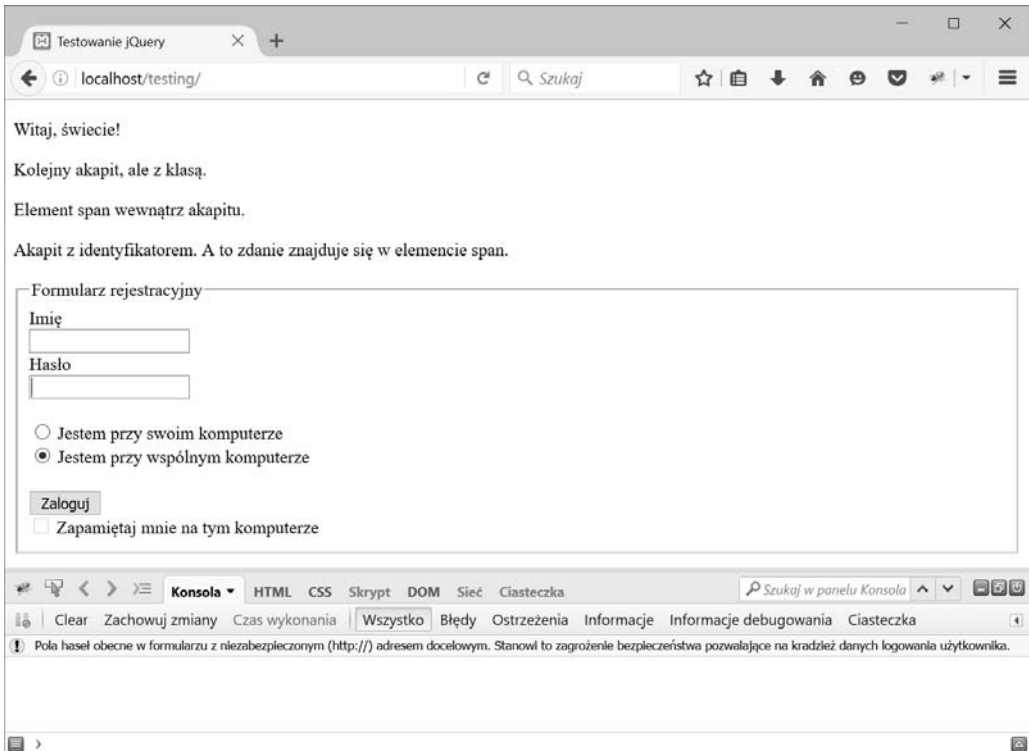
Formularze stanowią obecnie niezwykle ważną część stron WWW. Ich niebagatelna rola przyczyniła się do powstania szeregu filtrów nastawionych na obsługę formularzy.

Ponieważ w naszym przykładowym dokumencie HTML nie ma formularza, trzeba go dodać przed wykonaniem kolejnych ćwiczeń. Dodaj w pliku *index.html* między ostatnim znacznikiem akapitu i pierwszym znacznikiem skryptu następujący kod HTML:

```
<form action="#" method="post">  
  <fieldset>  
    <legend>Formularz rejestracyjny</legend>  
    <label for="name">Imię</label><br />  
    <input name="name" id="name" type="text" /><br />  
    <label for="password">Hasło</label><br />  
    <input name="password" id="password" type="password" /><br />  
    <label>  
      <input type="radio" name="loc" />  
      Jestem przy swoim komputerze  
    </label><br />  
    <label>  
      <input type="radio" name="loc" checked="checked" />  
      Jestem przy wspólnym komputerze  
    </label><br /><br />  
    <input type="submit" value="Zaloguj" /><br />  
    <label>  
      <input type="checkbox" name="notify" disabled="true" />  
      Zapamiętaj mnie na tym komputerze  
    </label><br />  
  </fieldset>  
</form>
```

Po zapisaniu zmian odśwież w przeglądarce stronę <http://localhost/testing/>, aby wyświetlić testowy formularz. Powinien on wyglądać tak, jak na rysunku 1.5.

-
- **Uwaga** Ponieważ strona zawiera pole hasła, a jest wyświetlana w trybie zwykłym (niezabezpieczonym), w konsoli może się pojawić ostrzeżenie o zagrożeniu, takie jak na rysunku 1.5. Konfigurowanie serwera Apache na komputerze lokalnym tak, aby strony były wyświetlane w bezpieczny sposób, może być zbyt daleko posuniętą ostrożnością, lecz aplikacje w środowisku produkcyjnym powinny używać bezpiecznego protokołu https. Więcej informacji uzyskasz pod adresem <https://httpd.apache.org/docs/2.4/ssl/>.
-



Rysunek 1.5. Wygląd formularza po zmianach w pliku index.html

Dopasowanie na podstawie typu elementu formularza

Najczęściej używane filtry związane z formularzami dopasowują typ elementu. Są to `:button`, `:checkbox`, `:file`, `:image`, `:input`, `:password`, `:radio`, `:submit` oraz `:text`.

Aby zaznaczyć wszystkie pola wyboru typu radio, wystarczy użyć następującego kodu:

```
$("input:radio");
```

Po jego uruchomieniu w konsoli zostanie wyświetlony wynik:

```
> $("input:radio");
Object[ input property value = "on" attribute value = "null", input property value = "on"
↪attribute value = "null" ]
```

Filtry te przydają się szczególnie dlatego, że wszystkie wymienione wyżej typy pól to elementy `input`. Wybór poszczególnych typów pól formularza bez tych filtrów byłby więc znacznie utrudniony.

Wybór włączonych lub wyłączonych elementów formularza

Kolejne dwa filtry pozwalają wybrać włączone lub wyłączone elementy formularza. Są to odpowiednio `:enabled` i `:disabled`. Aby zaznaczyć wszystkie wyłączone elementy formularza, należy użyć następującej instrukcji:

```
$(":disabled");
```

Po jej wykonaniu w konsoli zostanie wyświetlony wynik:

```
> $(":disabled");
Object[ input property value = "on" attribute value = "null" ]
```

Opcja *Zapamiętaj mnie na tym komputerze* jest wyłączona i dlatego znajduje się w zbiorze zwracanym przez filtr `:disabled`.

Wybór zaznaczonych lub wybranych elementów formularza

Pola wyboru typu radio i typu checkbox wejścia mają stan `checked`³, natomiast lista rozwijana (element `select`) ma stan `selected`⁴. Dostęp do elementów formularza będących w którymkolwiek z tych stanów zapewniają, odpowiednio, filtry `:checked` i `:selected`.

Aby wybrać w naszym przykładzie aktualnie zaznaczone pole wyboru typu radio, użyj następującego kodu:

```
$(":checked");
```

W konsoli wyświetlone zostanie pole wyboru typu radio, które jest aktualnie zaznaczone:

```
> $(":checked");
Object[ input property value = "on" attribute value = "null" ]
```

Podsumowanie

W tym rozdziale dowiedziałeś się, czym jest biblioteka jQuery i dlaczego została stworzona, a także poznałeś podstawy jej działania. Skonfigurowaliśmy środowisko, instalując XAMPP-a, Firefoksa i dodatek Firebug.

W tym momencie powinieneś z łatwością wybierać elementy z DOM przy użyciu mechanizmu selektorów jQuery. Być może uznasz ten rozdział za mało ekscytujący, ale to ważne, żebyś w pełni rozumiał, jak działa jQuery, zanim skoczysz na głębszą wodę.

W następnym rozdziale omówimy operacje na drzewie DOM przy użyciu wbudowanych metod jQuery (między innymi: poruszanie się w jego obrębie, dostęp do elementów oraz ich modyfikację).

³ gdy są zaznaczone — *przyp. tłum.*

⁴ wskazujący wybraną pozycję — *przyp. tłum.*

Skorowidz

A

- adres URL, 253
- AJAX, 75, 226, 229, 243
- algorytmy szyfrujące, 190
- aliasy, 315
- animacja, 65
- aplikacja Kalendarz wydarzeń, 113
- arkusz stylów CSS, 160, 217
- atak typu blokada usługi, 191
- atrybuty, 28
 - CSS, 55

B

- baza danych, 113
- bezpieczeństwo, 191
- biblioteki
 - jQuery, 15
 - JavaScript, 15
- budowa skryptów, 33

C

- cele stopniowego ulepszania, 216
- chronione metody i właściwości, 100
- ciąg zaburzający, 190, 191
- CSS, 143, 160, 217
- CSS3, 144

D

- definiowanie
 - klasy, 185
 - metod klasy, 88
 - właściwości klasy, 87

- dekodowanie znaków, 253
- deserializacja danych, 251
- destruktory, 90
- DocBlock, 105
- dodatek Firebug, 18
- dodawanie
 - funkcji, 315
 - funkcji do obiektu, 316, 320
 - wydarzenia, 251, 259
- dołączenie
 - arkusza stylów, 218
 - biblioteki jQuery, 216
 - pliku konfiguracyjnego, 217
- DOM, Document Object Model, 21, 34
- dopasowanie
 - do wzorca, 298
 - formatu daty, 303
- dostęp
 - do domyślnych opcji, 320
 - do narzędzi administratora, 204
 - do oryginalnych metod, 98
 - do stron administracyjnych, 209
 - do właściwości, 99
- drzewo DOM, 21, 34
- duplikaty wydarzeń, 270
- dziedziczenie, 95
- dzielenie całkowite, 335

E

- edycja
 - kalendarza, 241
 - wydarzeń, 155, 265

edytor poleceń, 44
 efekty specjalne, 233
 elementy
 DOM, 21
 rodzeństwo, 25

F

filtr
 button, 31
 checkbox, 31
 filtr
 contains, 27
 empty, 28
 eq, 27, 29
 even, 26, 29
 file, 31
 first, 26
 has, 27
 image, 31
 input, 31
 last, 26
 not, 26
 nth-child, 29
 odd, 26, 29
 parent, 28
 password, 31
 radio, 31
 submit, 31
 text, 31
 filtrowanie, 25, 29
 na podstawie atrybutów, 28
 na podstawie widoczności, 28
 na podstawie zawartości, 27
 filtry związane z formularzami, 30
 Firebug, 17
 Firefox, 17
 foldery publiczne i niepubliczne, 115
 formularz, 30, 155
 administratora, 175
 do tworzenia wydarzeń, 163
 logowania, 183
 potwierdzenia, 180
 wylogowania, 201
 frameworki JavaScript, 15
 funkcja
 \$.ajax(), 76, 226, 229, 242
 \$.ajaxSetup(), 77
 \$.get(), 79

\$.getJSON(), 80
 \$.getScript(), 80
 \$.post(), 79
 fx.addevent, 255
 fx.initModal(), 224
 jQuery.noConflict(), 315
 preg_match(), 305
 funkcja session_destroy(), 201
 sleep(), 194
 validDate(), 311
 funkcje
 narzędziowe, 223, 225
 skrótów, 190

G

generowanie kodu HTML, 148, 150
 granice słowa, 298
 grupowanie, 292
 selektorów, 23

H

hasło, 190
 hierarchia
 folderów aplikacji, 114
 selektorów, 24

I

identyfikator
 miesiąca, 256
 wydarzenia, 261, 268
 informacje o wydarzeniu, 153, 226
 inicjalizacja wtyczki, 326
 inspektor elementów, 45
 instalacja
 Firebuga, 17
 Firefoksa, 17
 XAMPP, 18
 interfejs użytkownika, 215

J

JavaScript, 15
 jQuery, 13, 15
 metody, 33
 operacje, 33

K

Kalendarz wydarzeń, 113

- dodawanie wydarzeń, 155, 251
- edycja, 241
- edycja wydarzeń, 155, 265
- interfejs użytkownika, 215
- planowanie aplikacji, 113
- tworzenie aplikacji, 117
- usuwanie wydarzeń, 175
- wylogowywanie, 199
- zapis wydarzeń, 162

klasa, 86

- active, 220
- Admin, 185
- Calendar, 120, 256, 305
- Event, 131

klasy znaków, 294, 297

kod

- HTML kalendarza, 134
- HTML przycisku usuwania, 176
- obiektowy, 107
- proceduralny, 107

komentarze DocBlock, 105

komunikat o błędzie, 307, 309, 314

konfiguracja

- środowiska testowego, 17, 18
- właściwości obiektu, 126

konsola Firebuga, 22

konstruktor, 90

- klasy Calendar, 122

kontrola

- dostępu, 209
- poprawności danych, 287
- poprawności daty, 301, 310
- uprawnień użytkownika, 183, 210

konwersja do łańcucha znaków, 93

L

literały obiektowe, 225

logowanie, 183

Ł

ładowanie

- dokumentu, 71
- kopii biblioteki, 19

łańcuch

- zapytania, 221, 248
- znaków, 93

łączenie

- selektorów, 23
- wywołań, 322

M

mechanizm dziedziczenia, 95

metoda

- .add(), 40
- .addClass(), 60
- .after(), 48
- .andSelf(), 41
- .animate(), 67
- .append(), 45
- .appendTo(), 47
- .attr(), 55
- .before(), 48
- .bind(), 72
- .children(), 36
- .closest(), 37
- .contents(), 42
- .css(), 56
- .data(), 59
- .delay(), 69
- .detach(), 54
- .die(), 73
- .each(), 63, 324
- .end(), 42
- .eq(), 34
- .error(), 70
- .fadeIn(), 65
- .fadeOut(), 65
- .fadeTo(), 65
- .filter(), 35
- .find(), 37
- .first(), 35
- .getDay(), 258
- .has(), 35
- .hasClass(), 61
- .height(), 61
- .hide(), 65
- .html(), 57
- .innerHeight(), 61
- .innerWidth(), 61
- .insertAfter(), 49
- .insertBefore(), 49

metoda

.is(), 36
 .last(), 35
 .live(), 73
 .load(), 80
 .map(), 63
 .next(), 38
 .nextAll(), 38
 .nextUntil(), 38
 .not(), 35
 .one(), 74
 .outerHeight(), 61
 .outerWidth(), 61
 .parent(), 39
 .parents(), 40
 .parentsUntil(), 40
 .prepend(), 45
 .prependTo(), 47
 .prev(), 39
 .prevAll(), 39
 .prevUntil(), 39
 .ready(), 71
 .remove(), 54
 .removeAttr(), 56
 .removeClass(), 60
 .scroll(), 70
 .show(), 65
 .siblings(), 39
 .slice(), 36
 .slideDown(), 66
 .slideToggle(), 66
 .slideUp(), 66
 .stop(), 69
 .text(), 57
 .toggle(), 74
 .toggleClass(), 60
 .trigger(), 75
 .unbind(), 72
 .unload(), 71
 .unwrap(), 50
 .val(), 58
 .width(), 61
 .wrap(), 49
 .wrapAll(), 52
 .wrapInner(), 52
 _adminEntryOptions(), 171
 _adminGeneralOptions(), 205
 _createEventObj(), 132, 149
 _getSaltedHash(), 193
 _loadEventById(), 149

_loadEventData(), 132
 _validDate(), 305
 buildCalendar(), 134
 displayEvent(), 150, 151, 172
 processForm(), 162, 163, 262, 307
 processLogout(), 201

metody

jQuery, 33
 klasy, 88
 prywatne, 102
 publiczne, 99
 statyczne, 104
 rozszerzające, 315

modyfikacja

atrybutów CSS, 55
 elementów, 324
 pliku init.js, 318
 pliku stopki, 318
 rdzenia aplikacji, 195, 203
 środowiska programistycznego, 116

modyfikatory, 289, 291

N

nagłówkek, 146
 narzędzia administratora, 204
 nawiasy
 klamrowe, 298
 kwadratowe, 28
 negacja klasy, 297

O

obiekt, 86
 Date, 256
 obsługa
 formularza potwierdzenia, 180
 wysyłania formularza, 275, 280
 zdarzenia click, 220
 zdarzeń, 70, 72, 75, 224
 zadań AJAX, 75, 226, 243
 odczyt
 atrybutów CSS, 55
 łańcucha zapytania, 221
 odnośnik, 220
 odwołania wsteczne, 292
 odwołanie do jQuery, 19
 okno modalne, 219, 222, 225
 opcje administratora, 172, 205

operator

<=>, 337

*, 298

??. 336

+, 298

operatory powtórzenia, 298

P

pakiet XAMPP, 18

parametryzacja funkcji, 316

PHP, 83

PHP 7

deklarowanie typów, 331

dzielenie całkowite, 335

nowe wyjątki, 333

rozpakowywanie argumentów, 335

stałe tablicowe, 333

phpMyAdmin, 184

planowanie aplikacji, 113

plik

admin.css, 160, 173

ajax.css, 217

ajax.inc.php, 249, 278

confirmdelete.php, 180

CSS, 143

do obsługi formularza potwierdzenia, 180

do przetwarzania formularza, 165

index.php, 125, 135

init.js, 217

konfiguracyjny aplikacji, 124

konfiguracyjny bazy danych, 123

konfiguracyjny JavaScript, 216

login.php, 186

process.inc.php, 195

testowy, 19

view.php, 173

z formularzem, 159

z wtyczką, 326

pliki

niepubliczne, 115

publiczne, 114

płynne wyświetlanie okna, 235

pobieranie danych wydarzenia, 229

podświetlenie liter, 296

pole action, 266

połączenie z bazą danych, 119

potwierdzenie usunięcia wydarzenia, 176, 274

predefiniowane klasy znaków, 297

programowanie

obiektowe, 85, 109

proceduralne, 108, 226

projektowanie bazy danych, 117

prywatne metody i właściwości, 102

przedwczesne wykonanie skryptu, 217

przesłanianie dziedziczonych właściwości i metod,
97

przetwarzanie

formularza, 165

wydarzenia, 149

zbiorów wynikowych, 62

przycisk

Anuluj, 246

do tworzenia wydarzeń, 167

edycji, 170, 175

usuwania, 176, 177

Wyloguj, 201, 202

Zamknij, 231

publiczne metody i właściwości, 99, 322

R

relacja rodzic-dziecko, 29

rozpakowywanie argumentów, 335

rozszerzanie biblioteki jQuery, 315

S

scalanie opcji, 317

selektor, 21

dzieci, 24

separatory, 289

serializacja danych, 247

składnia Perla, 287

skrypt, 33

sprawdzanie

danych logowania, 186

połączenia, 123

poprawności łańcucha daty, 310

stałe tablicowe, 333

statyczne właściwości i metody, 104

stopka, 146

stopniowe ulepszanie aplikacji, 215

strefa czasowa, 257

strona administratora, 160

struktura

bazy danych, 113

klasy, 86

style CSS, 143
 stylizacja strony wydarzenia, 173
 szkielet aplikacji, 113
 szyfrowanie haseł, 190, 193

Ś

środowisko
 lokalne, 116
 produkcyjne, 117
 testowe, 17

T

tabela użytkowników, 183
 tablica, 132
 obiektów wydarzeń, 130
 tekst dodany dynamicznie, 47
 testowanie
 opcji administratora, 194
 szyfrowania hasła, 193
 wyrażeń regularnych, 289
 tęczowe tablice, 190, 191
 token, 157
 tworzenie
 elementów DOM, 42
 kalendarza, 117, 136
 wtyczki, 320
 wydarzeń, 155, 167, 241

U

uprawnienia użytkownika, 183, 210
 usuwanie wydarzenia, 175, 210, 269, 279

W

walidacja, 287
 daty, 301, 305, 310, 317
 warstwa maskująca, 235
 wczytywanie danych wydarzenia, 128
 własne aliasy, 315
 właściwości klasy, 87
 Calendar, 121
 wstawianie elementów DOM, 45
 wtyczka, 315
 validDate(), 320
 wtyczki własne, 320

wybór

 dzieci, 24
 elementów
 DOM, 21
 formularza, 31, 32
 na podstawie identyfikatora, 23
 na podstawie indeksu, 27
 na podstawie typu znacznika, 22
 na podstawie wartości atrybutu, 28
 niepasujących do selektora, 26
 parzystych, 26
 potomnych, 24
 następnego sąsiada, 25
 ostatniego elementu, 26
 rodzeństwa, 25
 znaczników na podstawie nazwy klasy, 23
 wydarzenia, 128
 edycja, 155
 przycisk administratora, 171
 tworzenie, 155, 167
 usuwanie, 175
 zapis, 162, 246
 wygaszanie okna, 233
 wygląd kalendarza, 143
 wyjątki, 333
 wykrywanie początku łańcucha, 298
 wylogowywanie z aplikacji, 199
 wyrażenia regularne, 221, 287
 alternatywy, 299
 dopasowanie formatu daty, 303
 granice słowa, 298
 grupowanie, 292
 klasy znaków, 294
 modyfikatory, 291
 odwołania wsteczne, 292
 operatory powtórzenia, 298
 składnia, 287
 walidacja daty, 302, 305
 wyrażenia opcjonalne, 299
 złożone, 299
 wysyłanie
 formularza, 275, 280
 żądań AJAX, 79
 wyszukiwanie danych, 293
 wyświetlanie
 formularza, 242
 formularza logowania, 183
 opcji administratora, 205
 opisów wydarzeń, 148, 151

wydarzeń, 140
komunikatu o błędzie, 307
okna potwierdzenia, 274
wzorzec, 289, 299, 303

X

XAMPP, 18

Z

zabezpieczenie formularza, 157
zalety jQuery, 15
zamykanie okna, 246
zapis
 identyfikatora wydarzenia, 267
 wydarzeń, 162, 246
zapytania AJAX, 75
zastępowanie tekstu, 289
zdarzenia, 72, 75
 przeglądarki, 70
zdarzenie click, 220

znak

@, 106
dwukropka, 26
hash, 23, 289
kropki, 23
plusa, 25
tyldy, 25
ukośnika, 289
większości, 24
wykrzyknika, 29
zapytania, 299

Ż

żądania AJAX, 226, 243

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

PHP i JavaScript często idą w parze, ponieważ ich połączenie pozwala na łatwe budowanie elastycznych i dynamicznych aplikacji internetowych. Szczególnie interesujące możliwości drzemą w połączeniu PHP w wersji 7 z lekką, wszechstronną biblioteką jQuery. API jQuery jest proste w użyciu i może być obsługiwane przez różne przeglądarki. Dzięki temu możliwe są coraz bardziej złożone interakcje z użytkownikiem oraz obsługa dużych implementacji aplikacji, zwłaszcza że PHP 7 w porównaniu z wcześniejszymi wersjami może pochwalić się lepszą wydajnością i mniejszym zużyciem pamięci.

Niniejsza książka jest przeznaczona dla średnio zaawansowanych programistów, którzy chcą tworzyć lepsze aplikacje z wykorzystaniem technologii PHP i jQuery. Na przykładzie budowy kompletnej aplikacji WWW pokazano tu mnóstwo zaawansowanych technik PHP i metod doskonalenia aplikacji z frameworkiem jQuery. Niejako przy okazji zaprezentowano też sporo nowych możliwości PHP 7, takich jak określanie typów argumentów funkcji, stałe tablicowe, nowe typy wyjątków i stosowanie kilku bardzo przydatnych operatorów. Przedstawione tu z pewnością wzbogacą warsztat każdego programisty WWW!

W tej książce znajdziesz najważniejsze informacje o:

- jQuery — o budowie skryptów, typowych operacjach i metodach
- elementach DOM, atrybutach CSS i zapytaniach AJAX
- programowaniu obiektowym w PHP 7
- optymalizacji interfejsu użytkownika za pomocą jQuery
- rozszerzaniu biblioteki jQuery

Keith Wald — jest doktorem fizyki. Wykłada na Uniwersytecie Kalifornijskim. Jest doświadczonym projektantem stron WWW, programistą PHP, JavaScriptu, Pythona i Perla. Przez wiele lat projektował układy scalone, a obecnie zajmuje się rozwijaniem oprogramowania wbudowanego.

Jason Lengstorf — pochodzi z Montany. Jest projektantem i programistą stron WWW. Tworzy oprogramowanie do zarządzania treścią z wykorzystaniem PHP, MySQL i AJAX-a. Jego hobby to gra w golfa, podróże i degustacja piwa.



księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

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

Sprawdź najnowsze promocje:
• <http://helion.pl/promocje>
Książki najchętniej czytane:
• <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
• <http://helion.pl/nowości>



ISBN 978-83-283-3035-1



Informatyka w najlepszym wydaniu

cena: 67,00 zł