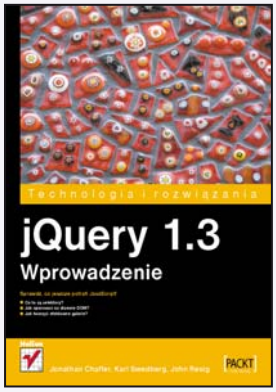
jQuery 1.3. Wprowadzenie



Autorzy: Jonathan Chaffer, Karl Swedberg, John Resig
Tłumaczenie: Anna Trojan
ISBN: 978-83-246-2641-0
Tytuł oryginału: Learning jQuery 1.3
Format: 170×230, stron: 424

Język JavaScript po blisko piętnastu latach na rynku dalej ma się dobrze. Interaktywne strony, interesujące efekty oraz technologia AJAX sprawiają, że wciąż jest bardzo atrakcyjnym narzędziem, a aplikacje internetowe z każdym rokiem coraz bardziej zaczynają przypominać te znane z codziennej pracy. Biblioteka jQuery pozwala na jeszcze więcej przy zdecydowanie mniejszym nakładzie pracy! Niemożliwe? Przekonaj się sam!

W trakcie lektury tej książki poznasz tajniki biblioteki jQuery oraz atuty, dzięki którym zyskała ona taką rzeszę fanów. Autorzy omawiają wszystkie zagadnienia związane z tą biblioteką. Na samym początku nauczysz się korzystać z selektorów oraz zdarzeń, aby następnie przejść do zaawansowanych tematów związanych z AJAX-em oraz edycją drzewa DOM. Rozdział poświęcony efektom specjalnym będzie jednym z tych, które pozwolą Ci wykrzesać jeszcze więcej porywających efektów z Twojego kodu. Olśniewające galerie, pokazy slajdów, interaktywne formularze są w zasięgu Twoich możliwości! Pod koniec książki dowiesz się, jak wykorzystywać dodatki oraz tworzyć własne. Trzymasz w rękach długo oczekiwaną na polskim rynku pozycję, poświęconą niezwykłej bibliotece!

- Przeznaczenie biblioteki jQuery
- Wykorzystanie selektorów
- Wykorzystanie mechanizmu zdarzeń
- Zdarzenia złożone
- Użycie efektów specjalnych
- Modyfikowanie arkuszy CSS „w locie”
- Edycja drzewa DOM
- Metody na łatwiejsze wykorzystanie technologii AJAX
- Operacje na tabelach
- Praca z formularzami
- Instalowanie oraz wykorzystanie dodatków
- Tworzenie własnych dodatków dla jQuery

Uzyskaj jeszcze więcej z języka JavaScript!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

1

ROZDZIAŁ

Początki

Dzisiejszy Internet to dynamiczne środowisko, a jego użytkownicy wysoko stawiają poprzeczkę, zarówno jeśli chodzi o styl, jak i o funkcje stron internetowych. By móc tworzyć interesujące i interaktywne witryny, programiści sięgają po biblioteki JavaScriptu, takie jak jQuery, które pomagają im automatyzować często wykonywane zadania i upraszczać te bardziej skomplikowane. Jednym z powodów, dla których biblioteka jQuery cieszy się takim powodzeniem, jest to, że jest w stanie wspomóc programistę w wielu różnych zadaniach.

Właściwie trudno jest zadecydować, od czego zacząć, ponieważ jQuery ma tyle różnych funkcji. Mimo to struktura biblioteki jest bardzo spójna i symetryczna — większość jej koncepcji zapożyczono z projektów takich, jak **HTML** i **kaskadowe arkusze stylów** (**CSS**). Sama struktura biblioteki ułatwia zaczęcie pracy projektantom z niewielkim doświadczeniem programistycznym, ponieważ wielu twórców stron internetowych ma o wiele większe doświadczenie z HTML i CSS niż z JavaScriptem. Już w pierwszym rozdziale książki napiszemy zresztą w jQuery działający program składający się z trzech wierszy kodu. Z drugiej strony również doświadczeni programiści skorzystają na tej konceptualnej spójności, o czym przekonamy się w dalszych, bardziej zaawansowanych rozdziałach.

Przyjrzyjmy się zatem temu, co może dla nas zrobić jQuery.

Co robi jQuery

Biblioteka jQuery udostępnia ogólny poziom abstrakcji służący do tworzenia skryptów na potrzeby często spotykanych zadań, dzięki czemu przydatna jest w niemal każdej sytuacji wymagającej napisania skryptu. Rozszerzalna natura biblioteki oznacza, że nigdy nie udałoby nam się opisać wszystkich jej możliwych zastosowań i funkcji w jednej książce, ponieważ dodatki ciągle są tworzone, dokładając nowe możliwości. Najważniejsze funkcje jądra biblioteki są jednak następujące:

- **Uzyskanie dostępu do elementów dokumentu.** Bez biblioteki JavaScriptu przejście drzewa **DOM** (ang. *Document Object Model*) i lokalizacja wybranych części struktury dokumentu HTML wymaga napisania wielu wierszy kodu. jQuery oferuje rozbudowany i wydajny mechanizm selektorów służący do pobierania fragmentu dokumentu, który ma być przejrzany lub zmodyfikowany.
- **Modyfikowanie wyglądu strony internetowej.** CSS oferuje metodę wpływania na sposób wyświetlania dokumentu, jednak na niewiele się przydaje, kiedy nie wszystkie przeglądarki obsługują te same standardy. Dzięki jQuery programiści są w stanie wypełnić tę lukę, polegając na obsłudze tych samych standardów we wszystkich przeglądarkach. Dodatkowo jQuery jest w stanie modyfikować klasy lub pojedyncze właściwości stylu zastosowane do części dokumentu nawet po wygenerowaniu strony.
- **Zmiana zawartości dokumentu.** jQuery nie ogranicza się jednak do zmian czysto kosmetycznych — jest w stanie zmodyfikować zawartość samego dokumentu za pomocą kilku naciśnięć klawiatury. Można zmieniać tekst, wstawiać lub podmieniać obrazki, zmieniać kolejność list lub przepisać bądź rozszerzyć całą strukturę HTML — wszystko za pomocą jednego, łatwego w użyciu **API** (ang. *Application Programming Interface*).
- **Reagowanie na interakcję z użytkownikiem.** Nawet najbardziej wyszukane działania nie są szczególnie przydatne, jeśli nie możemy kontrolować tego, kiedy następują. Biblioteka jQuery oferuje elegancki sposób przechwytywania różnego rodzaju zdarzeń — takich jak kliknięcie odnośnika przez użytkownika — bez konieczności zaśmiecania kodu HTML programami obsługi zdarzeń. Jednocześnie API służące do obsługi zdarzeń likwiduje niezgodności między przeglądarkami, które są dla programistów internetowych prawdziwą katorgą.
- **Animacja zmian wprowadzanych do dokumentu.** By efektywnie implementować tego typu działania interaktywne, projektant musi dać użytkownikowi jakieś wizualne informacje zwrotne. Biblioteka jQuery umożliwia to, udostępniając wiele efektów (takich jak blaknięcie czy animowane przejścia), a także służąc jako narzędzie do tworzenia nowych efektów.
- **Pobieranie informacji z serwera bez odświeżania strony.** Ten wzorzec kodu znany jest jako **Ajax** (od *Asynchronous JavaScript and XML*) i wspomaga programistów w tworzeniu bogatych w możliwości oraz reagujących na działanie użytkownika stron. Biblioteka jQuery likwiduje z tego procesu część związaną z różnicami między poszczególnymi przeglądarkami, pozwalając programistom na skupienie się na funkcjonalności po stronie serwera.
- **Uproszczenie często spotykanych zadań wykonywanych za pomocą JavaScriptu.** Oprócz wszystkich możliwości jQuery związanych z obsługą dokumentów biblioteka ta udostępnia także ulepszenia dla podstawowych konstrukcji języka JavaScript, takich jak iteracja i modyfikacja tablic.

Dlaczego jQuery działa dobrze

Wraz ze wzrostem zainteresowania dynamicznym HTML pojawiło się mnóstwo bibliotek i platform opartych na JavaScripcie. Niektóre z nich są wyspecjalizowane i skupiają się tylko na jednym bądź dwóch z wymienionych wyżej zadań. Inne starają się skatalogować wszystkie możliwe działania i animacje, udostępniając je jako gotowy pakiet. By zachować szeroką gamę możliwości przedstawionych powyżej, ale też niewielki rozmiar, jQuery przyjmuje kilka strategii:

- **Wykorzystanie znajomości CSS.** Opierając mechanizm lokalizacji elementów strony na **selektorach CSS**, jQuery dziedziczy zwięzły, a jednocześnie czytelny sposób wyrażania struktury dokumentu. Biblioteka jQuery staje się punktem wyjścia dla projektantów, którzy chcą dodawać do swych stron różnorodne działania, gdyż znajomość składni CSS jest podstawą zawodowego tworzenia witryn internetowych.
- **Obsługa dodatków.** W celu uniknięcia przeładowania funkcjami jQuery przenosi specjalne zastosowania do **dodatków**. Metoda tworzenia nowych dodatków jest prosta i dobrze udokumentowana, co pobudziło rozwój dużej liczby pomysłowych i przydatnych modułów. Nawet w podstawowym pliku jQuery większość opcji wewnętrznie korzysta z architektury dodatków, dzięki czemu można je w miarę potrzeby usunąć, uzyskując jeszcze mniejszą bibliotekę.
- **Abstrakcja niezgodności między przeglądarkami.** Tragedią programistów stron internetowych jest to, że każda przeglądarka ma swój własny zbiór odchyleń od opublikowanych standardów. Znaczna część procesu tworzenia każdej aplikacji internetowej przypada na obsługę tych samych funkcji w inny sposób dla każdej z platform. Choć ciągle ewoluujący świat przeglądarek sprawia, że w przypadku bardziej zaawansowanych funkcji stworzenie podstaw kodu całkowicie neutralnych dla przeglądarki jest niemożliwe, jQuery dodaje **poziom abstrakcji** normalizujący często wykonywane zadania, zmniejszający wielkość kodu i znacznie go upraszczający.
- **Praca ze zbiorami.** Kiedy nakazujemy jQuery odnalezienie wszystkich elementów klasy `collapsible`, a następnie ukrycie ich, nie ma potrzeby wykonywania pętli po każdym zwracanym elemencie. Zamiast tego metody takie jak `.hide()` zaprojektowano w taki sposób, by automatycznie działały na zbiorze obiektów, a nie tylko na pojedynczych obiektach. Technika ta, zwana **niejawną iteracją**, oznacza, że wiele konstrukcji pętli staje się zbędnych, co znacznie skraca kod.
- **Zezwolenie na wiele działań w jednym wierszu.** By uniknąć nadmiernego korzystania z tymczasowych zmiennych czy bezsensownego powtarzania, jQuery w większości metod wykorzystuje wzorzec programowania zwany **łańcuchem**. Oznacza to, że wynikiem większości działań na obiekcie jest sam obiekt, gotowy do zastosowania na nim kolejnych działań.

Strategie te pozwalają na utrzymanie niewielkiego rozmiaru pakietu jQuery — po skompresowaniu jest to poniżej 20 KB — udostępniając jednocześnie techniki umożliwiające zachowanie zwięzłości własnego kodu korzystającego z biblioteki.

Elegancja biblioteki po części jest cechą samego projektu jQuery, a po części efektem procesu ewolucyjnego pobudzanego przez aktywną społeczność, która wytworzyła się wokół projektu. Użytkownicy jQuery aktywnie omawiają nie tylko rozwój dodatków, ale także ulepszenia samego jądra biblioteki. W dodatku A przedstawimy wiele zasobów dostępnych dla programistów korzystających z jQuery.

Pomimo skali wysiłków niezbędnych do stworzenia tak elastycznego i rozbudowanego systemu produkt końcowy jest darmowy i dostępny dla każdego. Projekt ten dostępny jest na podwójnej licencji: **GNU Public License** (dzięki czemu można go dołączyć do wielu innych projektów open source) oraz **MIT License** (by ułatwić zastosowanie jQuery w oprogramowaniu własnościowym).

Historia projektu jQuery

Niniejsza książka omawia funkcjonalność i składnię **jQuery 1.3.x**, najnowszą wersję dostępną w czasie jej pisania. Założenia leżące u podstaw biblioteki — udostępnienie łatwych sposobów odnajdywania elementów na stronie internetowej oraz modyfikowania ich — nie zmieniły się w trakcie jej rozwijania, choć zmienione zostały szczegóły składni i dostępnych możliwości. Poniższy krótki przegląd historii projektu opisuje najważniejsze zmiany pomiędzy wersjami.

- **Publiczne ogłoszenie prac:** John Resig po raz pierwszy wspomniał o poprawkach do biblioteki Prototype (części „Behaviour") w sierpniu 2005 roku. Nowa biblioteka została oficjalnie opublikowana pod nazwą **jQuery** 14 stycznia 2006 roku.
- **jQuery 1.0 (sierpień 2006):** już pierwsze wydanie biblioteki miało rozbudowaną obsługę selektorów CSS, zdarzeń i interakcji opartych na Ajaksie.
- **jQuery 1.1 (styczeń 2007):** to wydanie znacznie usprawniło API. Wiele rzadko wykorzystywanych metod zostało połączonych, co zmniejszyło całkowitą liczbę metod, które trzeba opanować i udokumentować.
- **jQuery 1.1.3 (lipiec 2007):** to pomniejsze wydanie zawierało ogromną poprawę szybkości działania silnika selektorów jQuery. Od tej wersji wydajność jQuery wypada bardzo korzystnie na tle podobnych bibliotek JavaScriptu, takich jak Prototype, Mootools oraz Dojo.
- **jQuery 1.2 (wrzesień 2007):** w wydaniu tym usunięto składnię **XPath** służącą do wybierania elementów, gdyż była ona powtarzalna w stosunku do składni CSS. Dostosowywanie efektów do własnych potrzeb jest od tego wydania bardziej elastyczne; tworzenie dodatków stało się łatwiejsze dzięki wprowadzeniu **zdarzeń z przestrzeni nazw**.
- **jQuery UI (wrzesień 2007):** ogłoszono powstanie nowego zbioru dodatków, który miał zastąpić popularny, jednak starzejący się dodatek Interface. Obejmował on bogatą kolekcję gotowych widżetów, a także zbiór narzędzi służących do budowania bardziej zaawansowanych elementów, takich jak interfejsy oparte na zasadzie „przeciągnij i upuść".

- **jQuery 1.2.6 (maj 2008):** funkcjonalność popularnego dodatku Brandona Aarona o nazwie **Dimensions** została dołączona do jądra biblioteki.
- **jQuery 1.3 (styczeń 2009):** poważna przebudowa silnika selektorów (**Sizzle**) ogromnie poprawiła wydajność biblioteki. Oficjalnie obsługiwana jest również **delegacja zdarzeń**.

Informacje o starszych wersjach jQuery można znaleźć na stronie internetowej projektu pod adresem: *http://docs.jquery.com/History_of_jQuery*.

Nasza pierwsza strona z jQuery

Skoro omówiliśmy już możliwości udostępniane przez jQuery, możemy teraz sprawdzić, jak można wykorzystać tę bibliotekę w praktyce.

Pobranie jQuery

Oficjalna strona internetowa jQuery (*http://jquery.com/*) jest zawsze najlepszym adresem, pod którym można znaleźć aktualny kod i informacje dotyczące biblioteki. Na początek potrzebna jest nam kopia jQuery, którą można pobrać bezpośrednio na stronie głównej tej witryny. W każdym momencie dostępnych może być kilka wersji jQuery — dla nas jako programistów stron internetowych najbardziej odpowiednia będzie najnowsza nieskompresowana wersja biblioteki. W środowisku produkcyjnym można ją zastąpić wersją skompresowaną.

Nie jest wymagana żadna instalacja. By użyć jQuery, wystarczy umieścić plik na naszej stronie internetowej, w publicznie dostępnym miejscu. Ponieważ JavaScript jest językiem interpretowanym, nie musimy martwić się o fazę kompilacji. Zawsze gdy będziemy potrzebować jQuery na stronie internetowej, będziemy się po prostu odnosili do lokalizacji pliku z dokumentu HTML.

Utworzenie dokumentu HTML

Większość przykładów wykorzystujących jQuery składa się z trzech elementów: samego dokumentu HTML, plików CSS nadających mu styl i plików JavaScriptu pozwalających wykonywać działania. W naszym pierwszym przykładzie wykorzystamy stronę z fragmentem książki[1]; do części kodu przypisano kilka klas.

[1] W przykładzie wykorzystano fragment książki *Po drugiej stronie lustra* Lewisa Carrolla w tłumaczeniu Roberta Stillera — *przyp. tłum.*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">

<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Po drugiej stronie lustra</title>
   <link rel="stylesheet" href="alice.css" type="text/css" media="screen" />
   <script src="jquery.js" type="text/javascript"></script>
   <script src="alice.js" type="text/javascript"></script>
</head>

<body>
   <h1>Po drugiej stronie lustra</h1>
   <div class="author">autor: Lewis Carroll</div>
   <div class="chapter" id="chapter-1">
      <h2 class="chapter-title">1. Dom odbity w lustrze</h2>
      <p>Na stole koło Alicji leżała książka, więc kiedy tak siedziała,
      obserwując Białego Króla (bo wciąż trochę się o niego lękała i trzymała
      w pogotowiu atrament, żeby go oblać, gdyby jeszcze raz zemdlał),
      przewracała sobie kartki w poszukiwaniu czegoś, co by mogła przeczytać:
      <span class="spoken">"bo wszystko to w jakimś nieznanym języku!"</span>
      powiedziała do siebie.</p>
      <p>Brzmiało to tak.</p>
      <div class="poem">
         <h3 class="poem-title">IKAŁORBAŻ</h3>
         <div class="poem-stanza">
            <div>ywtąrks eikbilś ,ywałtzsurm sazc łyB</div>
            <div>,yłpyriwś cązcreiw hcazław aN</div>
            <div>ywtąlgorob anc od ełgzim A</div>
            <div>.yłpyzrgz yruzczsiwś enmodz I</div>
         </div>
      </div>
      <p>Zastanawiała się nad tym przez jakiś czas, aż błysnęła jej genialna
      myśl. <span class="spoken">"Przecież to jest, oczywiście, Lustrzana
      Książka! i jak na nią popatrzę w Lustrze, to słowa znów będą takie,
      jak trzeba."</span></p>
      <p>I odczytała następujący wiersz:</p>
      <div class="poem">
         <h3 class="poem-title">ŻABROŁAKI</h3>
         <div class="poem-stanza">
            <div>Był czas mrusztławy, ślibkie skrątwy</div>
            <div>Na ważach wiercząc świrypły,</div>
            <div>A mizgłe do cna boroglątwy</div>
            <div>I zdomne świszczury zgrzypły.</div>
         </div>
      </div>
   </div>
</body>
</html>
```

Układ plików na serwerze nie ma znaczenia. Odwołania z jednego pliku do drugiego należy jedynie dostosować tak, by pasowały do wybranego przez nas sposobu organizacji. W większości przykładów w książce w odwołaniach do plików wykorzystamy ścieżki względne (*../images/foo.png*), a nie bezwzględne (*/images/foo.png*). Pozwala to na lokalne uruchamianie kodu bez konieczności posiadania serwera WWW.

Natychmiast po zwykłym nagłówku HTML ładowany jest arkusz stylów. W tym przykładzie jest on wyjątkowo spartański.

```
body {
   font: 62.5% Arial, Verdana, sans-serif;
}
h1 {
   font-size: 2.5em;
   margin-bottom: 0;
}
h2 {
   font-size: 1.3em;
   margin-bottom: .5em;
}
h3 {
   font-size: 1.1em;
   margin-bottom: 0;
}
.poem {
   margin: 0 2em;
}
.highlight {
   font-style: italic;
   border: 1px solid #888;
   padding: 0.5em;
   margin: 0.5em 0;
   background-color: #ffc;
}
```

Po odwołaniu do arkusza stylów dołączone zostają pliki JavaScriptu. Istotne jest, by element `script` z biblioteką jQuery znajdował się *przed* elementem `script` dla naszych własnych skryptów. W przeciwnym razie biblioteka jQuery nie będzie dostępna, kiedy nasz kod będzie się próbował do niej odwołać.

W pozostałej części książki drukowane będą jedynie istotne części plików HTML i CSS. Pełne pliki dostępne są na stronie internetowej książki pod adresem: *http://helion.pl/ksiazki/jquer1.htm*.

Teraz nasza strona wygląda tak:

Po drugiej stronie lustra

autor: Lewis Carroll

1. Dom odbity w lustrze

Na stole koło Alicji leżała książka, więc kiedy tak siedziała, obserwując Białego Króla (bo wciąż trochę się o niego lękała i trzymała w pogotowiu atrament, żeby go oblać, gdyby jeszcze raz zemdlał), przewracała sobie kartki w poszukiwaniu czegoś, co by mogła przeczytać: "bo wszystko to w jakimś nieznanym języku!" powiedziała do siebie.

Brzmiało to tak.

IKAŁORBAŻ
ywtąrks eikbilś ,ywałtzsurm sazc łyB
,ułpyriwś cązoreiw hcazław aN
ywtąlgorob anc od ełgzim A
.ułpyzrgz yruzczsiwś enmodz I

Zastanawiała się nad tym przez jakiś czas, aż błysnęła jej genialna myśl. "Przecież to jest, oczywiście, Lustrzana Książka! i jak na nią popatrzę w Lustrze, to słowa znów będą takie, jak trzeba."

I odczytała następujący wiersz:

ŻABROŁAKI
Był czas mrusztławy, ślibkie skrątwy
Na wałzach wiercząc świrypły,
A mizgłe do cna boroglątwy
I zdomne świszczury zgrzypły.

Rysunek 1.1.

Wykorzystamy teraz jQuery do nadania nowego stylu tekstowi wiersza.

> Przykład ten ma na celu zademonstrowanie prostego zastosowania jQuery. W rzeczywistości ten typ stylizacji tekstu można wykonać za pomocą samego CSS.

Dodanie jQuery

Nasz kod umieścimy w drugim — obecnie pustym — pliku JavaScriptu, dołączonym do dokumentu HTML za pomocą kodu `<script src="alice.js" type="text/javascript"></script>`. W tym przykładzie wystarczą nam trzy wiersze kodu:

```
$(document).ready(function() {
   $('.poem-stanza').addClass('highlight');
});
```

Odnalezienie tekstu wiersza

Podstawowym działaniem jQuery jest wybranie części dokumentu. Wykonywane jest to za pomocą konstrukcji `$()`. Zazwyczaj przyjmuje ona jako parametr łańcuch znaków, który może zawierać dowolne wyrażenie selektora CSS. W tym przypadku chcemy odnaleźć wszystkie części dokumentu, do których przypisano klasę `poem-stanza`, dlatego selektor jest bardzo prosty. W książce opiszemy jednak również o wiele bardziej wyszukane możliwości. Różne sposoby odnajdywania części dokumentu omówimy w rozdziale 2.

Funkcja `$()` jest tak naprawdę fabryką dla **obiektu jQuery** będącego podstawowym budulcem, z jakim będziemy od teraz pracować. Obiekt jQuery zawiera zero lub większą liczbę elementów drzewa DOM i pozwala nam wchodzić z nimi w różnego rodzaju interakcje. W tym przypadku chcemy zmodyfikować wygląd części strony, co osiągniemy, zmieniając klasy przypisane do tekstu wiersza.

Wstawienie nowej klasy

Metoda `.addClass()`, tak jak większość metod jQuery, ma opisową nazwę (ang. *add class* — „dodaj klasę"). Dodaje ona klasę CSS do wybranej części strony. Jej jedynym parametrem jest nazwa klasy, jaką należy dodać. Metoda ta oraz jej odpowiednik `.removeClass()` pozwalają nam łatwo zaobserwować działanie jQuery, kiedy będziemy badać różne dostępne wyrażenia selektorów. Na razie nasz przykład po prostu dodaje klasę `highlight`, którą nasz arkusz stylów definiuje jako tekst z obramowaniem napisany kursywą.

Warto zauważyć, że w celu dodania klasy do wszystkich zwrotek wiersza (elementów `poem-stanza`) nie jest konieczne wykonywanie jakiejkolwiek iteracji. Tak jak wspominaliśmy, jQuery wykorzystuje w metodach takich jak `.addClass()` **iterację niejawną**, dzięki czemu jedno wywołanie funkcji wystarczy do zmodyfikowania wszystkich wybranych części dokumentu.

Wykonanie kodu

Połączenie `$()` i `.addClass()` wystarczy nam do osiągnięcia celu, jakim jest zmiana wyglądu tekstu wiersza. Jeśli jednak ten wiersz kodu wstawimy po prostu w nagłówku dokumentu, nie przyniesie to żadnego efektu. Kod w JavaScripcie wykonywany jest w momencie napotkania go przez przeglądarkę, a w czasie gdy przetwarzany jest nagłówek, nie ma jeszcze żadnego kodu HTML, któremu można by nadać styl. Musimy zatem opóźnić wykonanie kodu do czasu, gdy będzie dla nas dostępne drzewo DOM.

Tradycyjnym mechanizmem opóźniania wykonywania kodu w JavaScripcie jest wywołanie kodu wewnątrz **programu obsługi zdarzeń** (ang. *event handler*). Większość programów obsługi zdarzeń dostępna jest dla zdarzeń inicjowanych przez użytkownika, takich jak kliknięcia myszą i naciśnięcia klawiszy. Gdybyśmy nie mieli dostępu do jQuery, musielibyśmy polegać na programie obsługi zdarzeń `onload`, który wywoływany jest po wygenerowaniu strony (ze wszystkimi obrazkami). By wywołać kod ze zdarzenia `onload`, umieścilibyśmy go wewnątrz funkcji:

```
function highlightPoemStanzas() {
   $('.poem-stanza').addClass('highlight');
}
```

Następnie dołączylibyśmy tę funkcję do zdarzenia, modyfikując element `<body>` dokumentu HTML, tak by się do niej odwoływał:

```
<body onload="highlightPoemStanzas();">
```

W ten sposób nasz kod zostałby wykonany po całkowitym załadowaniu strony.

To rozwiązanie ma jednak swoje wady. Zmodyfikowaliśmy sam kod HTML, by wywołać zmianę zachowania. Tak ścisłe powiązanie struktury i funkcji zaśmieca kod, wymagając nieraz powtarzania wywołań tej samej funkcji na wielu różnych stronach lub — w przypadku zdarzeń takich jak kliknięcie myszą — przy każdym pojawieniu się elementu na stronie. Dodanie nowych działań wymaga wtedy wprowadzania zmian w wielu miejscach, co zwiększa szansę na popełnienie błędu i komplikuje równoległą pracę nad wieloma dokumentami projektantom i programistom.

By uniknąć tych problemów, jQuery pozwala zaplanować wywołanie funkcji na czas po załadowaniu DOM — bez czekania na obrazki — dzięki konstrukcji `$(document).ready()`. Z funkcją zdefiniowaną jak powyżej możemy zapisać:

```
$(document).ready(highlightPoemStanzas);
```

Technika ta nie wymaga wprowadzania jakichkolwiek modyfikacji do kodu HTML. Zamiast tego działanie jest dołączane w całości za pomocą pliku JavaScriptu. W rozdziale 3. dowiemy się, jak można odpowiadać na inne typy zdarzeń, rozdzielając ich efekty od struktury dokumentu HTML.

To wcielenie kodu jest jeszcze trochę nieoptymalne, ponieważ funkcja `highlightPoemStanzas()` definiowana jest tylko po to, by skorzystać z niej natychmiast — i to tylko raz. Oznacza to, że wykorzystaliśmy identyfikator w globalnej przestrzeni nazw funkcji, zyskując niewiele, a teraz musimy pamiętać, by z niego już więcej nie korzystać. JavaScript, podobnie jak inne języki programowania, zawiera sposób obejścia tego braku optymalności zwany **funkcjami anonimowymi** (czasami nazywany również **funkcjami lambda**). Za pomocą funkcji anonimowych możemy pisać kod w sposób, który został zaprezentowany na początku:

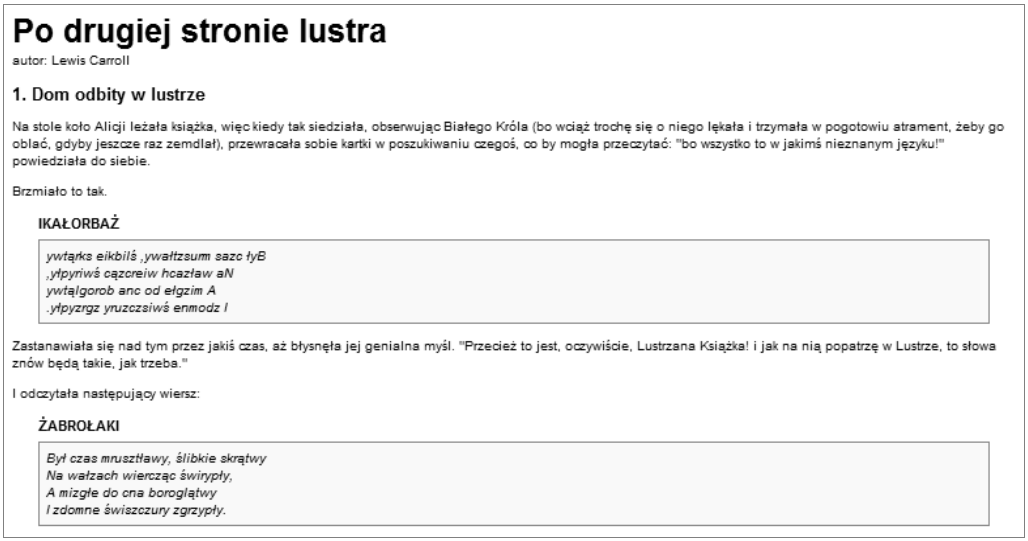
```
$(document).ready(function() {
   $('.poem-stanza').addClass('highlight');
});
```

Wykorzystując słowo kluczowe `function` bez nazwy funkcji, definiujemy funkcję dokładnie tam, gdzie jest potrzebna, a nie wcześniej. Usuwa to niepotrzebne rozbudowanie kodu i pozwala zredukować go do trzech wierszy JavaScriptu. Skrót ten jest szczególnie wygodny w przypadku kodu jQuery, gdyż wiele metod przyjmuje jako argument funkcję, a funkcje takie rzadko można wykorzystać ponownie.

Kiedy składnię tę wykorzystujemy do zdefiniowania funkcji anonimowej wewnątrz ciała innej funkcji, można utworzyć **domknięcie** (ang. *closure*). Jest to bardziej zaawansowana koncepcja o szerokich możliwościach, jednak powinno się ją dobrze zrozumieć, gdy intensywnie wykorzystuje się definicje zagnieżdżonych funkcji, gdyż może ona mieć niezamierzone konsekwencje i implikacje w zakresie wykorzystania pamięci. Kwestię tę omawiamy w całości w dodatku C.

Gotowy produkt

Kiedy nasz kod w JavaScripcie jest gotowy, strona wygląda tak:



Rysunek 1.2.

Zwrotki wierszy napisane są teraz kursywą i umieszczone są w ramce, zgodnie z arkuszem stylów *alice.css*, z powodu wstawienia klasy `highlight` za pomocą kodu w JavaScripcie.

Podsumowanie

Wiemy już, dlaczego programista mógłby wybrać korzystanie z biblioteki JavaScriptu, zamiast pisać cały kod od podstaw — nawet w przypadku najprostszych zadań. Widzieliśmy także, co sprawia, że jQuery sprawdza się jako doskonała biblioteka, której zastosowanie może być lepszym rozwiązaniem od innych opcji. Wiemy także mniej więcej, jakiego typu zadania jQuery może uprościć.

W niniejszym rozdziale nauczyliśmy się, jak udostępnić jQuery kodowi w JavaScripcie na naszej stronie internetowej, jak wykorzystać funkcję fabryczną `$()` do zlokalizowania części strony z przypisaną określoną klasą, jak wywoływać metodę `.addClass()` w celu nadania tej części strony dodatkowego stylu, a także jak wywołać `$(document).ready()`, co spowoduje, że kod ten zostanie wywołany po załadowaniu strony.

Prosty przykład, którym się posłużyliśmy, demonstruje sposób działania jQuery, jednak nie jest szczególnie przydatny w praktyce. W kolejnym rozdziale rozbudujemy ten kod, zapoznając się z wyszukanym językiem selektorów jQuery i znajdując praktyczne zastosowania dla tej techniki.