



PROFESJONALNY GAME DEVELOPMENT Z **JAVASCRIPT** I **HTML5**

TWORZENIE GIER INTERNETOWYCH RECEPTURY

 **Helion**

Evan BURCHARD

Tytuł oryginału: The Web Game Developer's Cookbook: Using JavaScript and HTML5 to Develop Games

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-8042-9

Authorized translation from the English language edition, entitled: THE WEB GAME DEVELOPER'S COOKBOOK: USING JAVASCRIPT AND HTML5 TO DEVELOP GAMES; ISBN 0321898389; by Evan Burchard; published by Pearson Education, Inc, publishing as Addison Wesley.
Copyright © 2013 Pearson Education, Inc.

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 Pearson Education, Inc.

Polish language edition published by HELION S.A. Copyright © 2014.

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.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi 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/twgint.zip>

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

Printed in Poland.

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

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

Spis treści

Przedmowa	11
O autorze	13
Wstęp	15
Rozdział 1. Quiz	19
Receptura. Tworzenie zbioru pytań	20
Receptura. Ukrywanie i pokazywanie quizu	25
Receptura. Przywracanie pytań do widoku	26
Receptura. Lista zakupów	28
Receptura. Które odpowiedzi są poprawne	32
Podsumowanie	34
Rozdział 2. Fikcja interaktywna	37
Receptura. Stylizowane strony	38
Receptura. Zmienianie stron	41
Receptura. Dodanie schowka z obsługą funkcji przeciągania i upuszczania	44
Receptura. Dodawanie złożonych interakcji	50
Receptura. Okruszki	59
Receptura. Dramatyczne zakończenie	61
Podsumowanie	63
Rozdział 3. Impreza	65
Receptura. Tworzenie przykładowej gry przy użyciu silnika atom.js	66
Receptura. Rysowanie na kanwie	70
Receptura. Rysowanie dziur	72
Receptura. Rysowanie kreta	74
Receptura. Umieszczanie kretów w dziurach	77
Receptura. Dynamiczne pokazywanie kreta	79
Receptura. Bicie kretów	80
Pogrążanie się w rozpacz z powodu elementu <audio> HTML5	83
Podsumowanie	85

Rozdział 4. Puzzle	87
Receptura. Renderowanie przy użyciu biblioteki easel.js	88
Receptura. Renderowanie większej liczby obiektów	92
Receptura. Tworzenie par	94
Receptura. Dopasowywanie i usuwanie par	97
Receptura. Ukrywanie i przekraczanie obrazków	99
Receptura. Wygrywanie i przegrywanie	100
Receptura. Buforowanie i wydajność	104
Receptura. Dopasowywanie par zamiast duplikatów	106
Podsumowanie	110
Rozdział 5. Gry platformowe	113
Wprowadzenie do biblioteki melon.js	114
Receptura. Tworzenie mapy kafelkowej	114
Receptura. Uruchamianie gry	116
Receptura. Dodawanie postaci	119
Receptura. Budowa mapy kolizji	121
Receptura. Chodzenie i skakanie	121
Receptura. Ekran tytułowy	123
Receptura. Dodawanie przedmiotów do zbierania	125
Receptura. Wrogowie	126
Receptura. Zwiększanie mocy postaci	128
Receptura. Przegrywanie, wygrywanie oraz informacje	129
Podsumowanie	131
Rozdział 6. Bijatyki	133
Receptura. Podstawowe wiadomości o bibliotece game.js	134
Receptura. Wybieranie poszczególnych sprite'ów z zestawu	136
Receptura. Odbieranie danych od dwóch graczy	137
Receptura. Poruszanie się i zmienianie formy	141
Receptura. Przyjmowanie danych od obu graczy naraz	144
Receptura. Implementacja masek bitowych	146
Receptura. Maskowanie kolizji	150
Receptura. Niszczenie z wzajemnością	152
Podsumowanie	156
Rozdział 7. Strzelanka	159
Trochę podstawowych informacji o renderowaniu	160
Receptura. Wstęp do gameQuery	160
Receptura. Dodawanie wrogów	163
Receptura. Tworzenie pojazdu	167
Receptura. Kolizje z wrogami	169
Receptura. Strzelanie	170
Receptura. Uzupełnianie mocy	172
Podsumowanie	174

Rozdział 8. Gry FPS	177
Receptura. Wprowadzenie do biblioteki Jaws	178
Receptura. Tworzenie mapy dwuwymiarowej	179
Receptura. Dodawanie postaci gracza	182
Receptura. Raycasting widoku z góry	186
Receptura. Imitacja trójwymiarowości przy użyciu raycastingu	190
Receptura. Dodawanie kamery	192
Receptura. Uatrakcyjnianie świata pod względem wizualnym	196
Receptura. Dodawanie przyjaciół i wrogów	200
Podsumowanie	207
Rozdział 9. RPG	209
Receptura. Wprowadzenie do biblioteki enchant.js	210
Receptura. Tworzenie mapy	211
Receptura. Dodawanie gracza	214
Receptura. Dodawanie warstwy kolizji	217
Receptura. Ekran stanu	220
Receptura. Interakcja z postaciami w grze	223
Receptura. Tworzenie schowka	226
Receptura. Tworzenie sklepu	228
Receptura. Tworzenie interfejsu bitwy	235
Receptura. Zapisywanie gry przy użyciu API Local Storage HTML5	245
Podsumowanie	247
Rozdział 10. Gry RTS	249
Potrzebujemy serwera	250
Receptura. Instalacja i uruchamianie Node	251
Receptura. Synchronizacja przy użyciu biblioteki Socket.IO	254
Receptura. Tworzenie mapy izometrycznej przy użyciu silnika crafty.js	257
Receptura. Rysowanie jednostek	259
Receptura. Poruszanie jednostkami	263
Receptura. Sterowanie gracza i widoczność	265
Receptura. Kolizje dla destrukcji i sprawdzenia przeciwnika	270
Podsumowanie	275
Rozdział 11. Dalszy rozwój	277
Co się wydarzyło?	278
Co dalej?	278
Dodatek A Podstawy JavaScriptu	281
Główne typy API w JavaScriptcie	282
API rdzenne	282
Implementacja API	282
API bibliotek	282
Własne API	283
Instrukcje	283

Zmienne	283
Łącuchy	284
Liczby	284
Tablice	284
Funkcje	285
Obiekty	285
Instrukcje warunkowe	286
Pętle	286
Komentarze	287
Dodatek B Kontrola jakości	289
Przeglądarkowe narzędzia do diagnostyki błędów	290
Testowanie	291
Współpraca	292
Dodatek C Zasoby	295
Silniki gier	296
Edytory tekstu	297
Przeglądarki	297
Inne narzędzia	298
Tworzenie i wyszukiwanie sztuki	299
Dema i poradniki	299
Książki	300
Portale internetowe	300
Skorowidz	303

Quiz

Jest to rodzaj gry o bardzo prostych zasadach. Trzeba udzielić odpowiedzi na pytania i albo się je zna, albo się zgaduje. Jest to niezwykle popularny typ gry, można go spotkać m.in. w interaktywnych teleturniejach telewizyjnych. Nawet jeśli gra jest bardziej skomplikowana niż prosty zestaw pytań i odpowiedzi, to tak jak każdy program, działa w oparciu o pewną logikę. Gdy król spyta Cię, czy chcesz walczyć ze smokiem, i odpowiesz „tak”, to jest to mały prosty quiz. Przesadą byłoby stwierdzenie, że wpadnięcie w otchłań w platformówce albo strata wszystkich punktów mocy w grze RPG jest tym samym, czym udzielenie odpowiedzi w quizie, ale jednak programowanie zasad gry i ich konsekwencji w każdym gatunku wygląda podobnie.

Receptura. Tworzenie zbioru pytań

Biorąc pod uwagę fakt, że tę książkę mogą czytać początkujący, starałem się, aby objaśnienia w tym rozdziale były jak najprostsze. Dalsze rozdziały są bardziej skomplikowane. Natomiast materiał znajdujący się w tym rozdziale ma umożliwić zrozumienie tematu każdemu. Każdy od czegoś zaczyna i dla niektórych takim początkiem może być właśnie ten tekst. Jeśli treść tego rozdziału wydaje Ci się banalna, możesz go tylko przejrzeć albo w ogóle pominąć. W dalszych rozdziałach są opisane bardziej złożone i trudniejsze zagadnienia.

Opis gry w tym rozdziale pełni trzy funkcje. Po pierwsze, chcę na jego bazie objaśnić podstawy języków HTML, CSS i JavaScript. Z tych trzech technologii najważniejszy jest język JavaScript. Jeśli słabo znasz podstawy tego języka, przeczytaj poradnik znajdujący się w dodatku A. Po drugie, w opisywanych grach jest używanych wiele różnych bibliotek i chcę mieć pewność, że będziesz wiedzieć, skąd je brać. Po trzecie, chcę ustalić wygodne i powtarzalne zasady tworzenia, edytowania, zapisywania i otwierania plików będących podstawą tej książki.

Jeśli nie masz edytora tekstu, to musisz się w niego zaopatrzyć. Edytorów do tworzenia i edytowania plików HTML, CSS oraz JavaScript jest mnóstwo. Jeśli nie wiesz, który wybrać, możesz skorzystać z podpowiedzi zawartych w dodatku C.

Jeśli masz już edytor tekstu, uruchom go, utwórz w nim plik *quiz/start/index.html* i wklej do niego kod z listingu 1.1. Jeśli jeszcze nie pobrałeś plików z serwera FTP, poszukaj informacji o nich we wstępie.

Listing 1.1. Plik index.html — struktura dokumentu HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quiz</title>
    <link rel="stylesheet" type="text/css" href="main.css">
  </head>
  <body>
    <h1>Quiz</h1>
    <div id="quiz">
      </div>
  </body>
</html>
```



UWAGA

Akronim HTML pochodzi od angielskich słów *Hypertext Markup Language* (język znakowania hipertekstu). Nazwa ta powstała w dawnych czasach, gdy **łącza** nazywano także **hiperłączeniami** i istniały jeszcze inne technologie pozwalające na przemieszczanie się między dokumentami. Hipertekst to zwykły tekst zawierający łącza. Znaczniki to specjalne fragmenty tekstu otaczające zwykły tekst w celu określenia jego funkcji. Krótko mówiąc, HTML to zestaw wytycznych składniowych określających sposób łączenia różnych rodzajów tekstu w celu uzyskania wzajemnie powiązanych stron będących plikami z rozszerzeniem *.html*.

Znacznik HTML to tekst znajdujący się w <nawiasie trójkątym>. Natomiast **element** HTML to para znaczników, <otwierający> i <zamykający>, wraz z tym, co znajduje się między nimi (zwróć uwagę na znak / w znaczniku zamykającym).

Tworzenie dokumentu zaczyna się od zadeklarowania jego typu przy użyciu deklaracji DOCTYPE. Dzięki temu przeglądarka internetowa „wie”, że otrzymała do przetworzenia dokument HTML. Przeglądarki mogą też otwierać dokumenty innych typów, od plików XML przez pliki dźwiękowe po obrazy graficzne. Dlatego też zadeklarowanie typu dokumentu jako normalnej strony internetowej jest bardzo ważne. Pewnie zastanawia Cię, co się stanie, jeśli tego nie zrobisz. W zależności od przeglądarki skutki mogą być zarówno mało znaczące, jak i straszne. Bądź co bądź tak naprawdę nigdy nie wiadomo, co się stanie, i dlatego najlepiej jest nie zapominać o tym drobiazgu na początku dokumentu.

Na drugim miejscu znajduje się znacznik `<html>`. Jest to globalny kontener dokumentu, który zazwyczaj zawiera dwa elementy: `head` i `body`, jak widać na powyższym listingu. Zwróć uwagę, że wszystkie wymienione elementy składają się ze znacznika otwierającego i zamykającego z ukośnikiem (`/`). Znacznik zamykający pozwala umieścić w elemencie inny element.

W elemencie `head`, ogólnie rzecz biorąc, wpisuje się informacje, które są ważne dla przeglądarki internetowej, ale nie mają bezpośredniego wpływu na to, co użytkownik widzi na ekranie. Element `meta` ma wiele zastosowań. W tym przypadku został użyty do określenia sposobu kodowania dokumentu. Jeśli nie określi się kodowania, to znaki niewchodzące w skład standardowego (i mocno ograniczonego) zestawu będą traktowane w nieprzewidywalny sposób. Jeśli będziesz pisać np. tylko po angielsku, to pewnie nie napotkasz trudności, ale z typowo polskimi literami typu `ś`, `ć` możesz mieć problemy. Nawet w konsoli JavaScript (a także narzędziu Firebug i narzędziach dla webmasterów przeglądarki Chrome) jest wyświetlane powiadomienie, że brakuje tego składnika. Mimo to w tej książce w większości przypadków ten element jest opuszczony, aby można było skupić się na tym, co jest nowego w każdym rozdziale.

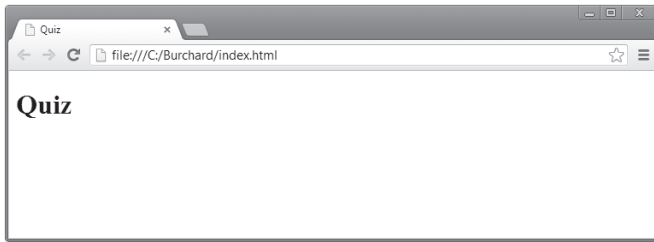
Treść elementu `title` jest wyświetlana w górnej części okna przeglądarki, na górnej belce, karcie albo na obu. Wykorzystują ją też aplikacje do tworzenia skrótów i zakładek jako „zwięzły opis strony”.

Dalej znajduje się element `link` z atrybutem `rel` ustawionym na `stylesheet`, atrybutem `type` informującym, że dołączany jest arkusz stylów CSS, oraz atrybutem `href` zawierającym ścieżkę do tego arkusza. W tym przypadku ścieżką jest nazwa pliku, co oznacza, że ten plik znajduje się w tym samym folderze co plik `index.html`. Element ten jest często używany do dołączania do stron arkuszy stylów i oprócz ścieżki praktycznie zawsze wygląda tak samo. W przypadku elementu `link`, jak również `meta` należy zwrócić uwagę na brak znacznika zamykającego (np. `</link>`). Elementy te nie są kontenerami i nie muszą mieć zamknięcia.

W elemencie `body` znajdują się dwa zagnieżdżone elementy. Pierwszy z nich to nagłówek `h1`, którego zawartość domyślnie jest wyświetlana powiększoną czcionką. Drugi to element `div` pełniący rolę kontenera oznakowanych informacji. W przedstawionym przykładzie element `div` ma atrybut `id`, który obok atrybutu `class` i samej nazwy elementu stanowi jeden z trzech najczęściej używanych sposobów wybierania elementów do formatowania za pomocą CSS (np. w celu zmiany koloru albo rozmiaru tekstu) i przetwarzania za pomocą JavaScriptu (np. gdy element zostanie kliknięty, strona ma zostać przekręcona do góry nogami).

Na razie w elemencie `div` jeszcze niczego nie ma, ale zanim coś do niego dodamy, sprawdzimy, czy wszystko działa tak, jak powinno. Zapisz plik `index.html` i uruchom przeglądarkę internetową. Aby otworzyć w niej swój plik, wpisz w pasku adresu jego adres, przeciągnij plik na pasek adresu przeglądarki albo kliknij go dwukrotnie myszą.

Gdy plik zostanie otwarty, w oknie przeglądarki powinien pojawić się widok pokazany na rysunku 1.1. Zwróć uwagę, że tytuł karty został ustawiony na *Quiz*, ponieważ to słowo wpisaliśmy w elemencie `title`.



Rysunek 1.1. Plik HTML otwarty w przeglądarce Chrome

Jeśli nie masz przeglądarki Chrome ani Firefoksa, to powinieneś je sobie teraz obie zainstalować. Ukazują różne problemy, jakie mogą występować podczas tworzenia gier przy użyciu HTML5, i obu będziesz często używać. Przeglądarki te nie są w tej książce traktowane jako idealne zamienniki.

Teraz dodamy trochę pytań w elemencie `div`. Widać je na listingu 1.2, gdzie zostały wyróżnione pogrubieniem. Jest to dość długi fragment kodu, ale znajduje się w nim wiele powtarzających się części. Jeśli nie masz chęci tego wszystkiego przepisywać, możesz to skopiować z pliku `quiz/po_recepturze1/index.html`.

Listing 1.2. Pytania quizu

```

<div id="quiz">
  <div id="question1">
    <div class="question">Który z tych typów plików nie jest używany do tworzenia
    ↪ stron internetowych?</div>
    <input type="radio" name="question1" value="a"/>
    <label>.html</label>
    <input type="radio" name="question1" value="b"/>
    <label>.exe</label>
    <input type="radio" name="question1" value="c"/>
    <label>.js</label>
    <input type="radio" name="question1" value="d"/>
    <label>.css</label>
  </div>
  <br />
  <div id="question2">
    <div class="question">Która para znaków jest używana do oznaczania obiektów
    ↪ JavaScript?</div>
    <input type="radio" name="question2" value="a"/>
    <label>[]</label>
    <input type="radio" name="question2" value="b"/>
    <label>;;</label>
    <input type="radio" name="question2" value="c"/>
    <label>{}</label>
    <input type="radio" name="question2" value="d"/>
    <label>()</label>
  </div>
  <br />
  <div id="question3">
    <div class="question">Krety są...</div>
    <input type="radio" name="question3" value="a"/>
    <label>wszystkożerne</label>
    <input type="radio" name="question3" value="b"/>
    <label>urocze</label>
    <input type="radio" name="question3" value="c"/>

```

```

<label>obrzydliwe</label>
<input type="radio" name="question3" value="d"/>
<label>wszystkie powyższe</label>
</div>
<br />
<div id="question4">
  <div class="question">Japoński znak "ぢ" wymawia się...</div>
  <input type="radio" name="question4" value="a"/>
  <label>ka</label>
  <input type="radio" name="question4" value="b"/>
  <label>ko</label>
  <input type="radio" name="question4" value="c"/>
  <label>ke</label>
  <input type="radio" name="question4" value="d"/>
  <label>ki</label>
</div>
<br />
<div id="question5">
  <div class="question">Stała grawitacji na Ziemi w przybliżeniu
  ↳wynosi...</div>
  <input type="radio" name="question5" value="a"/>
  <label>10 m/s2</label>
  <input type="radio" name="question5" value="b"/>
  <label>0,809 m/s2</label>
  <input type="radio" name="question5" value="c"/>
  <label>9,81 m/s2</label>
  <input type="radio" name="question5" value="d"/>
  <label>84,4 m/s2</label>
</div>
<br />
<div id="question6">
  <div class="question">Jak wygląda dziesiętna liczba 45 w systemie
  ↳dwójkowym?</div>
  <input type="radio" name="question6" value="a"/>
  <label>101101</label>
  <input type="radio" name="question6" value="b"/>
  <label>110011</label>
  <input type="radio" name="question6" value="c"/>
  <label>011101</label>
  <input type="radio" name="question6" value="d"/>
  <label>101011</label>
</div>
<br />
<div id="question7">
  <div class="question">4 << 2 = ...</div>
  <input type="radio" name="question7" value="a"/>
  <label>16</label>
  <input type="radio" name="question7" value="b"/>
  <label>4</label>
  <input type="radio" name="question7" value="c"/>
  <label>2</label>
  <input type="radio" name="question7" value="d"/>
  <label>8</label>
</div>
<br />
<div id="question8">
  <div class="question">Jak obliczyć długość przeciwprostokątnej trójkąta
  ↳prostokątnego, mając podane długości jego przyprostokątnych? </div>

```

```


<label>pi*promień^2</label>

<label>korzystając z twierdzenia Pitagorasa</label>

<label>używając kalkulatora</label>

<label>sin(bok1 + bok2)</label>
</div>
<br />
<div id="question9">
  <div class="question">Prawda czy fałsz: aby gra była coś warta, musi zmieniać
  ↳klatki z prędkością przynajmniej 60 na sekundę.</div>
  <input type="radio" name="question9" value="a"/>
  <label>prawda</label>
  <input type="radio" name="question9" value="b"/>
  <label>fałsz</label>
</div>
<br />
<div id="question10">
  <div class="question">Dzięki serwerowi można...</div>
  <input type="radio" name="question10" value="a"/>
  <label>ukryć swój kod</label>
  <input type="radio" name="question10" value="b"/>
  <label>utworzyć świetną grę</label>
  <input type="radio" name="question10" value="c"/>
  <label>umożliwić graczom wspólną grę</label>
  <input type="radio" name="question10" value="d"/>
  <label>wszystkie powyższe</label>
</div>
</div>

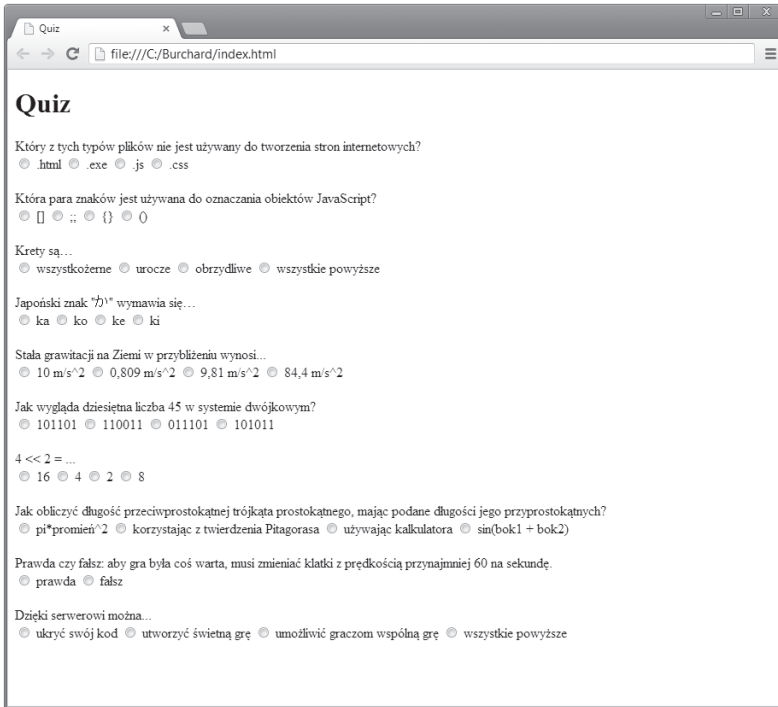
```

...

Wszystkie pytania w tym quizie mają taką samą ogólną strukturę. Różnią się natomiast numerami, treścią oraz możliwościami do wyboru. Przyjmijmy, że interesuje nas tylko pierwsze pytanie. Znajduje się ono w elemencie `div` o identyfikatorze (`id`) `question1`. Identyfikator jest niepowtarzalny i można go będzie później użyć do różnych celów. Ten element `div` zawiera samo pytanie i cały blok odpowiedzi. W nim jest zagnieżdżony kolejny element `div` zawierający tylko samo pytanie. Ma on przypisaną klasę (`class`) `question`. Przypomnę, że za pomocą klasy, podobnie jak nazwy elementu i identyfikatora, można się później odwoływać do elementu. Najważniejszą różnicą między klasą a identyfikatorem jest to, że identyfikator nie może powtarzać się na stronie, a liczba klas jest nieograniczona.

Dalej znajduje się element `input` z trzema atrybutami. Atrybut `type="radio"` oznacza, że został utworzony przycisk radiowy. Jeśli nie wiesz, jak on wygląda, spójrz na rysunek 1.2. Drugi atrybut to `name`. Każda odpowiedź w zestawie musi mieć inną wartość tego atrybutu. Atrybut `value` określa, co jest przesyłane jako wartość elementu po zatwierdzeniu formularza HTML. Podobnie przesyłana jest zawartość pola tekstowego. My nie będziemy zatwierdzać formularzy, ale będziemy korzystać z tych wartości do sprawdzania odpowiedzi przy użyciu JavaScriptu. Poznałeś już elementy wymagające i niewymagające znacznika zamykającego. Natomiast omawiany element `input` ma zakończenie `/>` oznaczające, że sam się zamyka.

Elementy `label` służą do oznaczania tekstu znajdującego się poza elementami `input`. Ich głównym zadaniem jest przeniesienie fokusu na odpowiednie pola wejściowe w reakcji na ich klik-



Rysunek 1.2. Pytania i odpowiedzi quizu

nięcie. Nie ma tego w przykładowym kodzie, ale jeśli chcesz, możesz nadać każdej odpowiedzi niepowtarzalny identyfikator, np. `id="question-10-answer-b"`, oraz użyć go w atrybucie `for` odpowiedniej etykiety, np. `<label for="question-10-answer-b">`.

Między każdą parą pytań znajduje się znacznik `
`, w którym ukośnik oznacza, że jest to samozamykający się element. Element `br` służy do rozsuwania elementów w pionie. Wysokość tej pustej przestrzeni jest zależna od przeglądarki i dlatego jeśli układ elementów jest ważny (w większości przypadków jest, ale tutaj akurat nie), należy zamiast tego elementu używać własności CSS.

Jeśli wszystko poszło zgodnie z planem, to po otwarciu pliku w przeglądarce powinieneś zobaczyć widok pokazany na rysunku 1.2.

Receptura. Ukrywanie i pokazywanie quizu

W grach często występują różnego rodzaju blokady, np. zablokowane postaci, niedostępne plansze albo ukryte poziomy. Tutaj mamy zablokowane pytania. Może się wydawać, że to przestarzałe techniki, ale to tylko złudzenie. Nie chciałbyś przechodzić wszystkich plansz gry Mario naraz, prawda? To samo dotyczy quizu. Gdyby zawierał 100 pytań, to lepiej byłoby nie wyświetlać ich wszystkich jednocześnie.

Jak można zablokować treść? Jest wiele możliwości, wśród których można wymienić np. umieszczenie jej grupami na różnych stronach. Jednak dla uproszczenia w tym przypadku użyjemy kodu CSS, aby ukryć część treści strony. W związku z tym w folderze zawierającym plik `index.html` musimy utworzyć plik o nazwie `main.css` i zawartości pokazanej na listingu 1.3.

Listing 1.3. Zawartość pliku `main.css` ukrywająca treść strony

```
#quiz{
  display:none;
}
body{
  margin-left:50px;
}
```

Tekst `#quiz` oznacza, że jest to reguła CSS odnosząca się do wszystkiego, co znajduje się w kontenerze, np. `div`, o identyfikatorze (`id`) `quiz`. Użyta tu deklaracja `display: none` powoduje ukrycie całej zawartości elementu `div` o identyfikatorze `quiz`. Gdybyśmy chcieli zdefiniować właściwości elementu o identyfikatorze `inny-quiz`, to użylibyśmy selektora `#inny-quiz`. A gdyby interesował nas element przypisany do klasy `quiz`, to zamiast znaku `#` użylibyśmy kropki, np. `.quiz`.

Przed selektorami elementów nie ma żadnych dodatkowych znaków, a więc aby odnieść się do elementu `body`, nie trzeba używać kropki ani krzyżyka. Zastosowana w tym przykładzie deklaracja `margin-left:50px;` przesuwa całą stronę nieco w prawo. Przyjrzyjmy się dokładniej strukturze tych dwóch bloków formatujących. Każdy z nich zawiera selektor, otwarcie klamry, deklaracje stylistyczne oraz zamknięcie klamry. Deklaracja stylu składa się z nazwy atrybutu po lewej, dwukropka, wartości tego atrybutu po prawej oraz średnika oznaczającego koniec wiersza.

Początkującym składnia ta może sprawiać drobne problemy, zwłaszcza gdy się ją połączy z elementami HTML oraz ich identyfikatorami, klasami i innymi atrybutami. Pocięszę Cię, że znasz już podstawy języków HTML i CSS. Później poznasz jeszcze inne atrybuty i selektory, ale najważniejsze podstawy już znasz. Pewnie nie raz popełnisz jakiś błąd, np. użyjesz krzyżyka zamiast kropki albo odwrotnie, zapomnisz o końcowym średniku lub zamknięciu klamry itp. Nie przejmuj się jednak. Są to powszechnie występujące błędy, których nie potrafią się ustrzec nawet zawodowcy. Jeśli coś nie będzie działać, zastanów się przez chwilę i dokładnie przeczytaj napisany przez siebie kod.

Jeśli teraz zapiszesz pliki i otworzysz plik `index.html` w przeglądarce, to zobaczysz, że strona wygląda tak jak na rysunku 1.1, chociaż jest odrobinę wcięta w prawo.

Receptura. Przywracanie pytań do widoku

Wszystkie pytania zniknęły i trzeba je jako przywrócić do widoku. Możesz to zrobić, dodając pakiety użyte w każdym z kolejnych rozdziałów. Każdy pakiet spowoduje pojawienie się jednego pytania.

Zanim zaczniemy ładować pakiety, musimy sprawdzić, czy w ogóle możemy ładować kod JavaScript. Prawie na samym dole pliku `index.html` wpisz kod wyróżniony pogrubieniem na listingu 1.4.

Listing 1.4. Ładowanie pierwszego zewnętrznego pliku JavaScript

```
...
  <script src="game.js"></script>
</body>
</html>
```

W ten sposób ładujemy na stronę plik JavaScript o nazwie `game.js`. Oczywiście musimy też go utworzyć. Utwórz plik o nazwie `game.js` w tym samym folderze, w którym znajdują się pliki `main.css` i `index.html`, oraz wpisz w nim kod widoczny na listingu 1.5.

Listing 1.5. Zawartość pliku `game.js`

```
alert('Witaj, świecie');
console.log('Witaj, świecie');
```

Kod ten drukuje informacje w dwóch miejscach. Pierwsze z nich stanie się oczywiste po otwarciu pliku `index.html`, ponieważ jest to wyskakujące okienko alertu. Natomiast instrukcja `console.log` powoduje wysłanie tekstu do konsoli JavaScript, która jest niezbędnym narzędziem dla każdego webmastera. Jeśli potrzebujesz pomocy na temat podstaw obsługi konsoli JavaScript, zajrzyj do dodatku B „Kontrola jakości”.

Teraz dodamy bibliotekę jQuery. Najprościej jest w tym celu wejść na stronę <http://jquery.com> i pobrać bibliotekę na swój dysk w dowolny sposób. Ja po prostu kliknąłem największy i najbardziej efektowny przycisk znajdujący się na stronie, aby przejść na stronę zawierającą kod tej biblioteki. Następnie go skopiowałem i wkleiłem do utworzonego własnoręcznie pliku o nazwie `jquery.js`. Na koniec zapisałem ten plik.

Na stronie jQuery można też pobrać plik biblioteki w tradycyjny sposób. Nieważne, jak ją zdobędziesz, pamiętaj tylko, aby umieścić ją w odpowiednim folderze na swoim dysku (w tym samym, w którym znajdują się pliki `index.html`, `main.css` i `game.js`).

Po umieszczeniu pliku w odpowiednim miejscu na dole pliku `index.html` dodaj kod wyróżniony pogrubieniem na listingu 1.6. Upewnij się, że nazwa pliku jest taka sama jak nazwa wpisana w tym kodzie.

Listing 1.6. Dodanie biblioteki jQuery do pliku `index.html`

```
...
  <script src="jquery.js"></script>
  <script src="game.js"></script>
</body>
</html>
```

Jeśli swojemu plikowi nadałeś inną nazwę niż `jquery.js`, pamiętaj, żeby zmienić ją także w powyższym kodzie.

Teraz przydałoby się nieco dostosować arkusz stylów. Wcześniej zadziałałoby trochę zbyt agresywnie. Teraz to zmienimy i zamiast ukrywać wszystkie pytania naraz, schowamy każde z nich osobno przy użyciu kodu widocznego na listingu 1.7.

Listing 1.7. Ukrywanie pytań, nie całego quizu

```
body{
  margin-left:50px;
}
#question1, #question2, #question3, #question4, #question5,
#question6, #question7, #question8, #question9, #question10{
  display:none;
}
```

Został usunięty selektor `#quiz`, a w jego miejsce wstawiliśmy listę rozdzielanych przecinkami selektorów identyfikatorów pytań. Można też było przypisać wszystkim pytaniom wspólną klasę i ukryć je wszystkie przy użyciu selektora kropki. Ale warto wiedzieć, że można też tworzyć takie listy selektorów jak powyższa.

Po ukryciu pytań za pomocą CSS możemy je odblokować przy użyciu jQuery. W tym celu musimy zmienić kod znajdujący się w pliku `game.js` na pokazany na listingu 1.8. Należy nim zastąpić poprzednią zawartość tego pliku.

Listing 1.8. Kod powodujący wyświetlenie pierwszego pytania, jeżeli jest załadowana biblioteka jQuery

```
if(jQuery){
  $("#question1").show();
};
```

Znajdująca się w pierwszym wierszu instrukcja warunkowa sprawdza, czy jest załadowana biblioteka jQuery. Jeśli tak, następuje wykonanie drugiego wiersza kodu. W tym wierszu jest użyta funkcja jQuery `$`, której przekazujemy selektor CSS `#question1` w cudzysłowie i nawiasie. Następnie wykonujemy funkcję `show` w celu zamiany deklaracji `display:none` pierwszego pytania na `display:block`.

Jeśli teraz zapiszesz pliki i otworzysz stronę `index.html` w przeglądarce, zobaczysz, że pojawiło się pierwsze pytanie.

Receptura. Lista zakupów

W tej recepturze zaimportujemy na naszą stronę jeszcze dziewięć dodatkowych plików. Pewnie zastanawiasz się, dlaczego warunkiem wyświetlenia pytań ma być załadowanie jakichś plików. Wielu osobom może się wydawać, że pobieranie plików na dysk, a następnie dołączanie ich do innych plików jest bez sensu. Jednak umiejętność korzystania z kodu napisanego przez inne osoby jest bardzo ważna. Niewiele projektów tworzy się zupełnie od podstaw, a nauczanie się tworzenia gier poprzez „stawanie na ramionach olbrzymów” jest naprawdę warte zachodu. Ponadto w tej części rozdziału zrobisz przegląd, jakiego rodzaju plików będziesz używać w dalszych częściach kursu.

Jeżeli wiesz, jak się dołącza pliki JavaScript do systemu, i dobrze znasz metody kontroli wersji, to pozostałe podrozdziały będą dla Ciebie jedynie powtórką. Możesz je tylko przejrzeć, a nawet pominąć, jeśli chcesz.

Mając załatwioną sprawę z najważniejszą w tym rozdziale biblioteką, jQuery, możemy udać się na dalsze zakupy. Jeśli masz ochotę na małą przygodę, to możesz wszystkie biblioteki pobrać z ich stron, których adresy znajdziesz w dodatku C „Zasoby”. Ale możesz też je wszystkie znaleźć w folderze `po_recepturze4` w katalogu plików do tego rozdziału. Nie zapomnij tylko umieścić wszystkich plików w tym samym folderze, w którym znajduje się plik `index.html`.

Po zdobyciu wszystkich plików w ten czy inny sposób Twój system plików powinien wyglądać tak jak na rysunku 1.3.

Teraz możesz rozpocząć dołączanie plików JavaScript do strony, dodając wiersze pogrubione na listingu 1.9 do pliku `index.html`.

ARTYKUŁY SPOŻYWCZE

1. **jquery.js**: ten plik już masz. Jest używany w kilku innych rozdziałach do wybierania elementów na stronie i manipulowania nimi.
2. **impress.js**: w rozdziale 3. „Impreza” tego narzędzia do tworzenia prezentacji (podobnego do PowerPointa, ale w JavaScriptcie) użyjemy do zarządzania „stronami” interaktywnej gry.
3. **atom.js**: jest to jeden z najmniejszych dostępnych silników gier (zawiera tylko 203 nieskompresowane wiersze kodu CoffeeScript). Skryptu tego użyjemy do budowy gry imprezowej.
4. **easel.js**: skrypt udostępniający udoskonalony interfejs do API kanwy, którego będziemy używać przy rysowaniu puzzli.
5. **melon.js**: tego silnika użyjemy do budowy platformówki w rozdziale 5.
6. **yabble.js**: w grze symulującej walkę wykorzystamy tę bibliotekę do załadowania silnika *game.js* (nie mylić z plikiem *game.js* użytym w tym rozdziale i innych).
7. **jquery.gamequery.js**: wtyczka do jQuery będąca silnikiem gier. Użyjemy jej do utworzenia strzelanki, w której poruszamy się z boku ekranu.
8. **jaws.js**: tego wszechstronnego silnika gier (i staroświeckiej trygonometrii) użyjemy do budowy typowej gry FPS.
9. **enchant.js**: japoński silnik gier o bogatej funkcjonalności i doskonałej obsłudze urządzeń mobilnych. Użyjemy go do budowy gry RPG w rozdziale 9. „RPG”.
10. **crafty.js**: rozbudowany i bardzo dobrze obsługiwany silnik gier, którego użyjemy do budowy gry RTS (gdybym miał wybrać jeden silnik, który zabrałbym na bezludną wyspę, to możliwe, że wybrałbym właśnie ten).



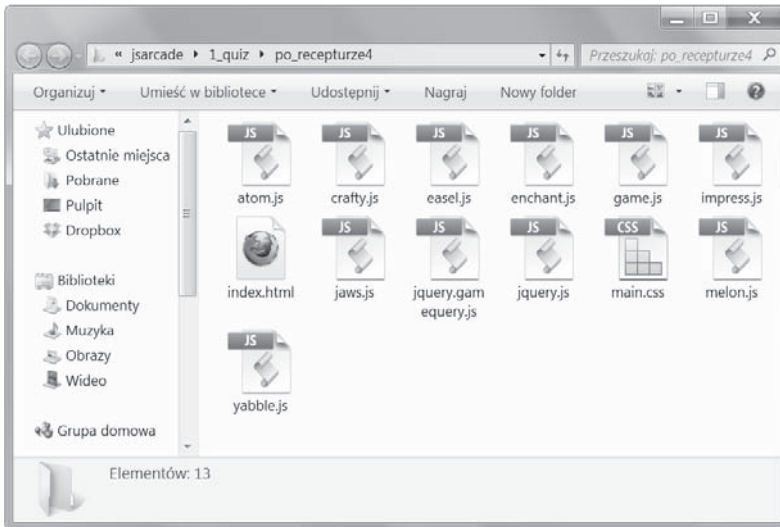
UWAGA

Jeśli przeczytałeś dodatek C, to zapewne zauważyłeś, że wszystkie wymienione pliki można też znaleźć w serwisie GitHub. Pliki z tego serwisu można pobierać na trzy sposoby. Po pierwsze, można pobrać cały projekt w formacie ZIP, wypakować pliki i użyć tych, które są potrzebne.

Po drugie, można przejrzeć zawartość projektu, kliknąć wybrany plik, skopiować jego zawartość, a następnie wkleić ją do nowo utworzonego pliku na własnym dysku. Może się wydawać, że to dużo zachodu, ale w istocie ta metoda jest naprawdę szybka.

Trzecia możliwość jest nieco bardziej skomplikowana, ale może zaowocować ułatwieniem pracy w przyszłości. Polega ona na zainstalowaniu programu Git w komputerze, pobraniu (sklonowaniu) projektu oraz przejściu do folderu tego rozdziału w celu pobrania plików. Możesz pracować bezpośrednio w tym folderze albo skopiować z niego potrzebne pliki.

Git to system kontroli wersji umożliwiający śledzenie zmian w plikach. Natomiast GitHub to portal internetowy, w którym osoby używające programu Git (wielu programistów z różnych krajów) mogą przechowywać własne projekty i znajdować projekty innych osób. Publicznie udostępnione projekty można przechowywać za darmo. Gorąco polecam skorzystanie z tej możliwości. Najlepszy poradnik instalacji programu Git znajduje się pod adresem help.github.com/articles/set-up-git.



Rysunek 1.3. Katalog zawierający wszystkie potrzebne pliki JavaScript

Listing 1.9. Dodawanie plików JavaScript do strony index.html

```

<script src="jquery.js"></script>
<script src="impress.js"></script>
<!-- To jest potrzebne do działania skryptu atom.js -->
<canvas></canvas>
<script src="atom.js"></script>
<script src="easel.js"></script>
<script src="melon.js"></script>
<script src="yabble.js"></script>
<script src="jquery.gamequery.js"></script>
<script src="jaws.js"></script>
<script src="enchant.js"></script>
<script src="crafty.js"></script>
<script src="game.js"></script>
</body>
</html>

```

Upewnij się, że nazwy plików dołączanych do strony zgadzają się z nazwami wpisanymi w elementach `script`. Pliki JavaScript do stron HTML zazwyczaj dołączają się właśnie przy użyciu elementu `script`. W całym tym kodzie znajduje się tylko jedna nietypowa rzecz — element `canvas` umieszczony między skryptem `atom.js` a komentarzem `<!-- -->`. Element ten jest potrzebny do działania biblioteki `atom.js`. Większość silników gier uruchamia się poprzez wywołanie funkcji inicjującej albo wskazanie konkretnego elementu `canvas`, który ma zostać wykorzystany. Jednak skrypt `atom.js` automatycznie szuka elementu `canvas` natychmiast, gdy tylko zostanie dołączony do strony. Zamiast z nim walczyć (tzn. edytować plik `atom.js`), lepiej jest dać mu to, czego chce. Znaki `<!-- -->` oznaczają komentarz HTML. Komentarze służą do wpisywania na stronie notatek przeznaczonych dla nas samych lub innych osób, które są ignorowane przez przeglądarki. Należy jednak pamiętać, że może je przeczytać każdy użytkownik, który zajrzy do kodu źródłowego strony. Jeśli nie wiesz, o co mi chodzi, przeczytaj dodatek B.

Teraz w pliku `game.js` przywrócimy do widoku pozostałe pytania naszego quizu. W tym celu należy dodać wiersze oznaczone na listingu 1.10 pogrubieniem.

Listing 1.10. Przywrócenie pozostałych pytań do widoku

```

if(jQuery){
  $("#question1").show();
};
if(impress){
  $("#question2").show();
};
if(atom){
  $("#question3").show();
};
if(createjs){
  $("#question4").show();
};
if(me){
  $("#question5").show();
};
if(require){
  $("#question6").show();
};
if($.playground){
  $("#question7").show();
};
if(jaws){
  $("#question8").show();
};
if(enchant){
  $("#question9").show();
};
if(Crafty){
  $("#question10").show();
};

```

Efekt dodania każdej z tych instrukcji jest od razu widoczny w postaci pojawienia się nowego obiektu na stronie. Jedynym wyjątkiem w tym bloku kodu jest test `playground` dotyczący pytania 7. `gameQuery` to rozszerzenie jQuery, a więc jego funkcje bazują na funkcjach tej biblioteki. Nie ma własnego rdzennego obiektu i dlatego trzeba sprawdzić dostępność funkcji `playground` w obiekcie `jQuery`.

**OSTRZEŻENIE**

TO NIE JEST TWÓJ KOD. Programiści piszący kod zazwyczaj chcą mieć odrobinę kontroli nad sposobem jego używania przez innych. Kontrolę tę sprawują poprzez dołączenie do kodu licencji. Nie oznacza to, że takich programów nie można albo nie należy używać. Niektóre licencje zabraniają tylko używać kodu w celach komercyjnych, inne wymagają podania gdzieś nazwiska autora skryptu, a jeszcze inne są tylko po to, aby skrypt mógł być zawsze używany. Szczegółowy opis kwestii licencjonowania oprogramowania wykracza poza zakres tej książki, ale jeśli przeczytasz licencje użytych w niej bibliotek lub poczytasz o licencjach Creative Commons, GPL, BSD i MIT, to będziesz się orientować, jak inni zapatrują się na kwestię otwartości oprogramowania. To samo dotyczy obrazów, plików dźwiękowych i innych typów treści.

Jeśli teraz zapiszesz plik *index.html* i otworzysz go w przeglądarce internetowej, to zobaczysz cały quiz, chociaż nie będzie on reagował na kliknięcia. Powodem tego jest rozciągnięcie na powierzchni całej strony elementu *canvas*, który jak niewidoczna płachta przykrywa wszystko, blokując dostęp do elementów strony. Problem ten rozwiążemy, dodając prostą regułę CSS (pogrubiony kod na listingu 1.11).

Listing 1.11. Kod CSS ukrywający element *canvas*

```
body{
  margin-left:50px;
}
#question1, #question2, #question3, #question4, #question5,
#question6, #question7, #question8, #question9, #question10{
display:none;
}
canvas{
  display:none;
}
```

Receptura. Które odpowiedzi są poprawne

Poprawne odpowiedzi można by było oznaczyć, dodając do nich klasę *correct*, ale to zbyt proste rozwiązanie zarówno pod względem implementacji, jak i ryzyka podejrzenia odpowiedzi przez użytkownika. Wszystko, co znajduje się w tych plikach, nawet komentarze, jest widoczne dla użytkownika, który jeśli nie będzie znał odpowiedzi, będzie mógł ją podejrzeć. Aby trochę utrudnić oszukiwanie osobom znającym się na programowaniu i uniemożliwić tym, które się nie znają, do sprawdzania odpowiedzi można użyć słabej funkcji mieszającej.

Funkcja mieszająca to funkcja pobierająca wartość i przekształcająca ją w inną wartość. Jej zaletą w tym przypadku jest łatwość, z jaką można odkryć pierwotną wartość, mając wynik mieszania.

Zanim ją napiszemy, najpierw utworzymy styl informujący w widoczny sposób, że wszystkie odpowiedzi są poprawne. Styl ten, zapisany w pliku *main.css*, jest pokazany na listingu 1.12 i wyróżniony pogrubieniem.

Listing 1.12. Styl włączany, gdy użytkownik poprawnie odpowie na wszystkie pytania

```
body{
  margin-left:50px;
}
#question1, #question2, #question3, #question4, #question5,
#question6, #question7, #question8, #question9, #question10{
display:none;
}
canvas{
  display:none;
}
.correct{
  background-color:#24399f;
  color:white;
}
```

Dodana reguła definiuje niebieskie tło i biały tekst dla elementów należących do klasy `correct`. Klasę tę można dodać do quizu, gdy użytkownik poprawnie odpowie na wszystkie pytania. W przedszkolu albo gdzieś indziej może słyzałeś o kolorze białym, ale kolor o nazwie `#24399f` raczej rzadko pojawia się w codziennych konwersacjach, nawet wśród absolwentów większości kierunków technicznych. Jest to definicja koloru w formacie RGB (ang. *red*, *green*, *blue* — czerwony, zielony, niebieski). Dwie pierwsze cyfry określają wartość czerwieni, następane dwie — wartość zieleni, a ostatnie dwie — ilość niebieskiego.

Ale chwileczkę, ostatnia cyfra to litera *f*. Litera to przecież nie cyfra. W istocie w dziesiętnym systemie liczbowym nie ma takiej cyfry. Ale gdybyśmy używali systemu dziesiętnego, to mielibyśmy do dyspozycji tylko 100 (0 – 9 i 0 – 9, czyli 10·10) wartości dla każdej z barw składowych. Ktoś uznał, że to za mało jak na sieć, i dlatego używamy systemu szesnastkowego, w którym dla każdej barwy RGB jest dostępnych 256 (16·16) odcieni. Istnieje też ograniczony zbiór nazw kolorów i można np. napisać `white` albo `#ffffff` oraz `black` albo `#000000`. Przy okazji ktoś inny kiedyś pomyślał sobie, że tych cyfr czasami jest za dużo, i dlatego powtarzające się cyfry można zredukować do trzech, np. kolor czarny można zapisać jako `#000`, a biały — `#fff`.

Po dodaniu kodu CSS pozostaje jeszcze zmienić coś w pliku `index.html`. Znacznik otwierający `<body>` zamień na znacznik oznaczony pogrubieniem na listingu 1.13.

Listing 1.13. Dodanie procedury obsługi kliknięcia do elementu `body` w pliku `index.html`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quiz</title>
    <link rel="stylesheet" type="text/css" href="main.css">
  </head>
  <body onclick="checkAnswers();">
```

Zamiast zwykłego znacznika `<body>` mamy teraz znacznik z atrybutem `onclick` zawierającym łańcuch kodu JavaScript w cudzysłowie. Jeśli dziwi Cię słowo „łańcuch”, przeczytaj dodatek A i dopiero potem wróć do tego miejsca. Łańcuch znajdujący się w tym atrybucie `onclick` powoduje wywołanie funkcji `checkAnswers` za każdym razem, gdy zostanie kliknięty jakiś element na stronie. Zwróć uwagę na nawias, który oznacza, że jest to wywołanie funkcji. Gdyby go nie było, po prostu odnosilibyśmy się do funkcji, ale byśmy jej nie wywoływali.

Na listingu 1.14 znajduje się ostatni przykład kodu prezentowany w tym rozdziale. Jest to treść opisaną powyżej funkcji. Pogrubiony kod z tego listingu można umieścić na początku pliku `game.js`, między testem obecności biblioteki jQuery a instrukcją powodującą wyświetlenie pierwszego pytania.

Listing 1.14. Sprawdzenie odpowiedzi

```
if(jQuery){
  var checkAnswers = function(){
    var answerString = "";
    var answers = $("":checked");
    answers.each(function(i) {
      answerString = answerString + answers[i].value;
    });
    $("":checked").each(function(i) {
```

```

        var answerString = answerString + answers[i].value;
    });
    checkIfCorrect(answerString);
};
var checkIfCorrect = function(theString){
    if(parseInt(theString, 16) === 811124566973){
        $("body").addClass("correct");
        $("h1").text("Wygrałeś!");
        $("canvas").show();
    }
};
$("#question1").show();
};
...

```

W pogrubionym kodzie znajdują się definicje dwóch funkcji. Pierwsza ma nazwę `checkAnswers` i tworzy pusty łańcuch, do którego będziemy dodawać kolejne odpowiedzi, gdy użytkownik będzie klikał przyciski radiowe. Po zakończeniu tej pętli zostaje wywołana druga funkcja, `checkIfCorrect`, porównująca otrzymany łańcuch z długą liczbą. Skąd wzięła się ta liczba?

Przypomnij sobie szesnastkowe wartości kolorów CSS. Można w nich używać cyfr od 0 do f. To oznacza, że litery a – d, będące odpowiedziami w naszym quizie, mogą być traktowane jak cyfry systemu szesnastkowego (można je traktować jak liczby 10 – 13). Połączyłem je w jeden łańcuch, który następnie zamieniłem na format dziesiętny.

Jeśli wynik porównania jest pozytywny, dodajemy do elementu `body` klasę `correct`, co powoduje zmianę koloru tła i tekstu. Dodatkowo następuje zamiana tekstu elementu `h1` z `Quiz` na `Wygrałeś!`. Na zakończenie wykorzystujemy ukryty wcześniej element `canvas` do zablokowania możliwości używania myszy na ekranie. Normalnie w celu zablokowania możliwości używania elementów formularza na stronie posłużylibyśmy się funkcją `jQuery.disable`, ale dzięki tej sztuczce znaleźliśmy zastosowanie dla elementu `canvas`, który w innym przypadku byłby całkiem bezużyteczny. Ponadto element ten można by było wykorzystać jako bazę gry opartej na silniku `atom.js` i wówczas udzielenie poprawnych odpowiedzi w quizie byłoby warunkiem rozpoczęcia gry.

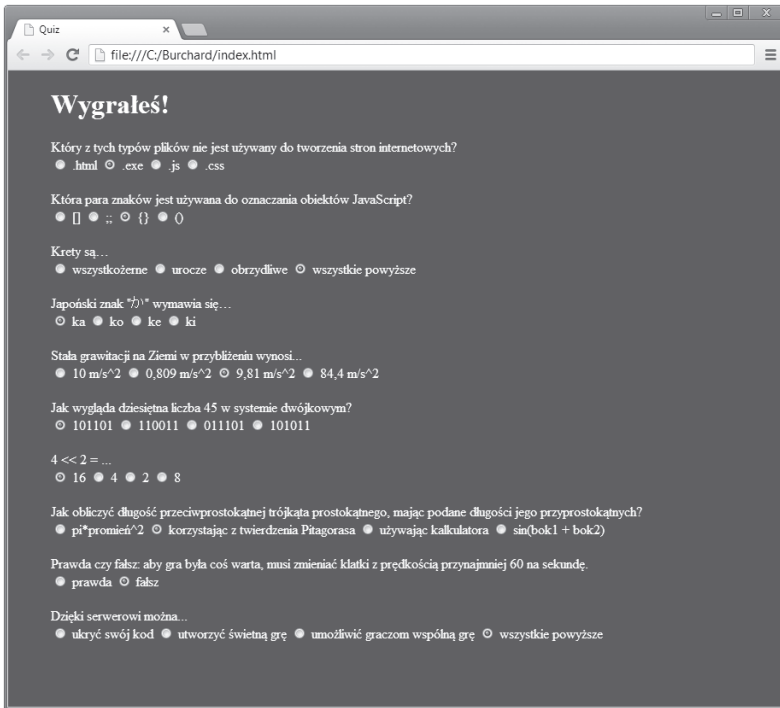
Po wykonaniu wszystkich opisanych czynności oraz zapisaniu wszystkich plików i otwarciu w przeglądarce pliku `index.html` powinieneś zobaczyć stronę pokazaną na rysunku 1.4.

Podsumowanie

W tym rozdziale utworzyliśmy prosty quiz z pytaniami mającymi związek z każdym rozdziałem tej książki. Wszystkie pytania były zablokowane i aby je odblokować, trzeba było dołączyć do strony pliki JavaScript, które będą używane w różnych rozdziałach. Do sprawdzania odpowiedzi użyliśmy prostej funkcji mieszającej zamieniającej wartości szesnastkowe reprezentujące odpowiedzi na długą wartość dziesiętną.

Podczas tworzenia tej gry poznałeś podstawy technologii HTML, CSS, jQuery, Git oraz dowiedziałeś się o istnieniu niektórych licencji na oprogramowanie. Ponadto poznałeś nazwy silników gier i innych bibliotek, których będziemy używać we wszystkich pozostałych rozdziałach.

Jeśli chcesz trochę poćwiczyć, możesz spróbować utworzyć drugą stronę z pytaniami, która będzie odblokowywana dopiero wtedy, gdy użytkownik udzieli prawidłowych odpowiedzi na pierwszej. W rozdziale 2. znajduje się opis jednej z możliwości wyświetlania bardziej dynamicznych



Rysunek 1.4. Wszystkie poprawne odpowiedzi z informacją o wygranej

informacji, a więc możesz w nim poszukać inspiracji. Ponadto na zakończenie gry przykryliśmy wszystko elementem canvas. Możesz umieścić na nim inną grę. Skrypt *atom.js* kontroluje go i czeka, aż wrócisz po lekturze rozdziału 3., aby coś na nim dodać, gdy już będziesz wiedział, jak to zrobić.

Jeśli treść tego rozdziału była dla Ciebie trudna do zrozumienia, przestuduj go jeszcze raz wraz z dodatkiem A. A jeśli nie znalazłeś w nim nic nowego, to nie przejmuj się. Od rozdziału 2. zaczynamy prawdziwą zabawę, a w rozdziale 7. idziemy już na całość.

Skorowidz

A

- activeMole, 81
- aktualizowanie graczy, 144
- anchor, 205
- API, 282
 - bibliotek, 282
 - implementacja, 282
 - localStorage, 245
 - rdzenne, 282
 - sieciowe, 250
 - własne, 283
- argument, 285
- atom.js, 29, 65
 - dostęp do elementu canvas, 70
 - podstawowy plik, 66
 - tworzenie przykładowej gry, 66
- atrybut
 - canChange, 144
 - class, 21
 - data-x, 42
 - fillStyle, 71
 - href, 21
 - id, 21
 - moleOffset, 78
 - name, 24
 - onclick, 33
 - onload, 89
 - player, 266
 - reallySuperDead, 274
 - rel, 21
 - room, 266
 - type, 21
 - type=, 24
 - value, 24
- audiocontext.play(noteOrFrequency), 83

B

- backbone.js, 66
- Bejeweled, 106
- biblioteka
 - Akihabara, 296
 - Atom, 296
 - atom.js, 30, 66
 - Crafty, 296
 - crafty.js, 250
 - cechy, 275
 - rysowanie tablicy izometrycznej, 257
 - wykrywanie kolizji, 271
 - dokumentacja dodatków, 92
- easel.js, 88
 - buforowanie, 104
 - renderowanie, 88
- enchant.js, 210
 - API, 247
 - cechy, 247
 - dokumentacja, 216
 - obiekt Group, 213
 - praca na urządzeniach przenośnych, 220
 - wiązanie klawiszy, 220
 - właściwości, 211
- filtr, 195
- game.js, 134
 - blit, 135
- Game.js, 296
- gameQuery, 160
 - dokumentacja, 164
 - funkcje, 175
 - interfejs playground, 163
- impress.js
 - dodawanie złożonych interakcji, 51

biblioteka
 Jaws, 178
 jQuery
 dodawanie do pliku, 27
 pobieranie, 27
 Raptorize, 61
 selektory, 166
 silnik gier, 29
 melonJS, 114
 kontekst renderowania kanwy, 124
 Melon Engine, 118
 narzędzia, 131
 przestrzeń nazw, 134
 warstwy kolizji, 121
 zapisywanie mapy, 116
 Node, 250
 instalacja i uruchamianie, 251
 pygame, 134
 Socket.IO, 250
 pobieranie, 255
 pokoje, 267
 synchronizacja, 254
 zalety wykorzystania, 178
 bijatyka, 133
 blit, 135
 definicje form, 155
 definicje nazw, 139
 definicje zmiennych pomocniczych, 148
 dodanie tekstu do gry, 143
 implementacja masek bitowych, 146
 koniec gry, 155
 maskowanie kolizji, 150
 narzędzia mask, 151
 niszczenie z wzajemnością, 152
 obsługa naciśnięć klawiszy, 139
 odbieranie danych od dwóch graczy, 137
 początkowy plik, 134
 poruszanie się, 141
 powiększanie, 135
 przesunięcie obiektów graczy, 155
 przyjmowanie danych od obu graczy naraz, 144
 rejestrowanie ciosów, 153
 silnik gry, 29
 sprite'y
 sprite'y, 135
 tworzenie obiektów graczy, 143
 wybieranie z zestawu, 136
 zmiana rozmiaru, 150
 zmiana sposobu obsługi klawiszy, 142
 zmienianie formy, 141
 block image transfer, 135
 blokady, 25
 blokowanie
 treści, 25
 używania elementów formularza, 34
 blokowe przesyłanie obrazu, 135

błąd składniowy, 49
 Box 2D Web, 300
 buforowanie, 104
 wyłączanie, 105
 bug, 289
 button, 220

C

caching, 104
 Can I use, 300
 Chrome, 297
 Chrono Trigger, 43
 class, 24
 closure, 50
 CoffeeScript, 66, 298
 dziedziczenie, 70
 konwersja na JavaScript, 66
 przykład kodu, 67
 utrudnione znajdowanie błędów, 66
 component-entity system, 257
 compositing, 135
 crafty.js, 29
 silnik gry, 257
 Crockford Douglas, 38
 cross-site scripting, 173
 CSS, 161
 definiowanie formatu stron, 39
 formatowanie, 21
 funkcja przeciągania przedmiotów, 45
 nawigacja między stronami, 39
 określanie kolorów, 197
 reset, 41
 ukrycie elementu canvas, 32
 ukrywanie części strony, 25
 wygląd w przeglądarkach, 41

D

Daily JS, 300
 dane
 w formacie JSON, 54
 debugowanie
 buforowanego systemu, 104
 definiowanie
 schowka, 46
 tytułu pliku HTML, 67
 deklaracja
 display
 block, 28
 none, 26
 DOCTYPE, 21
 html, 67
 margin-left
 50px, 26
 stylu, 26

distance, 191
 dodatek
 Firebug, 290
 DOM, 161
 dostępność dokumentu dla czytelników, 39
 Double Fine Adventure, 38
 dragDrop.js, 46
 modyfikacja, 52

E

Easel, 29,88, 296
 edytor map kafelkowych, 114
 edytor tekstu, 297
 wybór, 20
 ekran
 PlayScreen, 117
 element
 body, 21
 gradient tła, 41
 procedura obsługi kliknięcia, 33
 canvas, 30, 32, 67, 70, 90, 179, 181
 blokowanie myszy, 34
 dwuwymiarowy kontekst rysunkowy, 71
 game.js, 134
 znajdowanie wpliku HTML, 70
 dino, 54
 div, 21, 38
 dodawanie pytań quizu, 22
 head, 21
 HTML, 20
 input, 24
 inventory-box, 47
 label, 24
 link, 21
 meta, 21
 minimap
 arkusz stylów, 182
 dodanie do pliku, 182
 myAudio, 84
 playerBody
 dodanie symbolu, 172
 replay, 103
 screenshot, 195
 script, 61, 89, 134
 title, 21
 z identyfikatorem liczbowym, 51
 Emacs, 297
 Enchant, 29, 296
 entities.js, 120
 funkcja gameOver, 125
 ładowanie pliku, 120
 entity, 120
 Etsy, 299
 etykieta stanu, 222

F

fikcja interaktywna, 37
 dinozaura, 61
 dodanie
 kontenerów przedmiotów, 44
 schowka, 44
 stron historii, 38
 złożonych interakcji, 50
 dramatyczne zakończenie, 61
 formatowanie wnętrza slajdów, 43
 kod stron, 39
 nawigacja okruszkowa, 59
 obsługa interakcji, 46
 przechowywanie i pobieranie elementów, 48
 rozpoczęcie gry od nowa, 43
 slajdy, 38
 strona decyzyjna, 41
 strona zakończenia gry, 42
 fikcja literacka
 style okruszków, 60
 Filtr, 298
 funkcja
 update
 wywołanie dla graczy, 146
 Firebug, 298
 Firefox, 297
 folder
 gotowe, 17
 po_recepturze, 17
 for, 94
 fora dla programistów, 293
 forEach, 46
 porównanie z pętlą for, 46
 foreground, 114
 format
 .tmx, 114
 Base64, 116
 fps, 211
 funkcja, 285
 add, 49
 addChild, 213
 addChildAt, 108
 addCombatants, 243
 addItem, 56
 apply, 204
 arctan, 203
 areaMap, 263
 attack, 237
 beginPath, 71
 blank, 190, 192
 budowanie tablicy, 95
 call, 46
 callDino, 62
 camera.takePicture, 194
 canPlayType, 84

- funkcja
 - canvas.drawSliver, 191
 - canvas.init, 190
 - castRay
 - kolorowanie ścian, 198
 - castRays, 186
 - przeniesienie, 203
 - changeForm, 144
 - modyfikacja, 153
 - checkAnswers, 33, 34
 - checkIfCorrect, 34
 - clearInventory, 57
 - clearStatus, 226, 228
 - console.log, 253
 - containsBlock, 185
 - clearStatus
 - wywołanie, 226
 - deleteItem, 56
 - dino.draw, 205
 - disable, 34
 - displayStatus, 222
 - modyfikacja, 230
 - modyfikacja wyświetlania informacji, 237
 - przełączanie widoczności informacji, 226
 - doJump, 123
 - doWalk, 123
 - draw, 76, 124, 188, 189
 - modyfikacja, 83, 190, 204
 - drawHoles, 74
 - drawItemsForSale, 235
 - drawSliver
 - kanwy, 191
 - drawSquare, 93
 - modyfikacja, 91, 96
 - drawTextTile, 108
 - drawWhiskers, 76
 - dropItemInto, 62
 - end, 164, 165
 - eval, 173
 - zastąpienie, 173
 - facing, 224
 - facing.Square, 224
 - fillRect, 185
 - findTextNode, 56
 - floor, 185
 - focusViewport, 217
 - forEach, 47
 - function Eval, 173
 - game.onload
 - modyfikacja, 225, 231
 - game.slide, 55
 - gameOver, 110, 124
 - modyfikacja, 129
 - gameOver(), 102, 103
 - gameTick, 140
 - modyfikacja, 152, 155
 - getElementById, 90
 - getInventory, 56
 - getPlayerStatus, 244
 - getRandomPlacement, 96, 109
 - graphics.beginFill, 91
 - handleDragOver, 48
 - handleDragStart, 47
 - handleDrop, 48
 - handleEvent, 142
 - handleOnPress, 98, 109
 - aktualizacja bufora, 105
 - modyfikacja, 100, 103
 - hideInventory, 226, 228
 - hitStrength, 239, 240
 - hitTest, 220
 - init, 89, 118
 - deklaracje zmiennych, 92
 - dodawanie kwadratów, 102
 - modyfikacja, 122
 - modyfikacja pętli, 96
 - obiektu minimap, 181
 - pętla for, 108
 - renderowanie kwadratów, 94
 - renderowanie par, 107
 - wiązanie z oknem, 89
 - jsApp.onload, 118
 - JSON.parse, 247
 - JSON.stringify, 246
 - keydown, 171
 - lineTo, 76, 189
 - listen, 253
 - load, 150
 - loaded, 118
 - lost, 240
 - main, 135, 139, 140
 - renderowanie sprite'ów, 137
 - makeHoles, 77
 - markToDestroy, 274
 - Math.atan, 203
 - Math.floor, 91, 266
 - Math.round, 173
 - mieszająca, 32
 - move, 184
 - moveBy, 217
 - moveTo, 76, 189
 - moveUnit, 264
 - onDestroyEvent, 130
 - onHit, 272
 - onload, 118
 - dodanie własności coins i totalCoins, 130
 - onResetEvent, 124
 - instrukcje dla gracza, 129
 - parseInt, 247
 - pause, 241
 - placeUnits, 260
 - modyfikacja, 267
 - obsługa klikania i ruchu, 264
 - obsługa kolizji, 271

Player
 zapisywanie informacji o graczu, 152
 Player draw
 modyfikacja, 142
 player.displayStatus, 222
 player.draw, 183
 player.move, 217
 modyfikacja, 222
 preload, 118, 150
 przeciągania i upuszczania, 44
 push, 49, 96
 pushScene, 230
 randomColor, 91, 93
 registerCallbacks, 163
 registerHit, 153, 155
 remove, 49
 render, 195
 replay, 103, 110
 odświeżenie strony, 106
 reset, 195
 response.end, 253
 run, 67
 Run, 241
 samowykonująca, 52
 scaleUp, 136
 setBattle, 238
 setInterval, 269
 setMaps, 213
 dodanie warstwy kolizji, 217
 setPlacementArray, 95, 109
 setPlayer, 216, 222
 setShopping, 231
 setStage, 213
 setText, 56
 setTimeout, 234
 setup, 179
 modyfikacja, 190
 obiekt palette, 196
 shoppingFunds, 232
 show, 28
 showInventory, 226, 228
 modyfikacja, 230
 splice, 49
 sprite.draw, 206
 standardowa, 90, 92
 start, 179
 startGame, 163
 state.change, 118
 takePicture, 195
 text, 166
 tick, 102, 110
 toDataURL, 195
 uncache, 105
 unitsWithLimitedData, 269
 update
 modyfikacja, 79, 122
 obiektu Player, 154

 obsługa animacji podczas ruchu, 120
 poruszanie graczem, 183
 updateEnemyPositions, 269
 window.OnReady, 118
 window.open, 196
 with_key, 81, 82
 won, 240
 wywołanie, 33
 zmiany rozmiaru ekranu, 66
 zmienianie form, 141
 funkcje
 trygonometryczne w grze, 185

G

game jam, 15
 game.css, 179, 210
 game.js, 26, 179, 210
 dodanie własności screen, 58
 funkcja game.slide, 55
 funkcja placeUnits, 262
 kod kliencki Socket.IO, 256
 kod wiązania klawiszy, 220
 kolizje, 270
 nasłuchiwanie wiadomości place units, 262
 obsługa ruchu gracza, 214
 procedura obsługi kliknięć kafelków, 263
 silnik gry
 wykrywanie kolizji, 150
 uruchamianie aparatu, 194
 window.onload, 211
 GameQuery, 296
 Gedit, 297
 Gimp, 299
 magiczna różdżka, 150
 Git, 29, 252
 GitHub, 29, 253, 282
 współpraca, 292
 globalna przestrzeń nazw, 118
 gniazda sieciowe, 255
 gra
 FPS, 177
 dodawanie kamery, 192
 dodawanie postaci gracza, 182
 dodawanie przyjaciół i wrogów, 200
 imitacja trójwymiarowości, 190
 kierunek patrzenia, 185
 konfiguracja raycastera, 186
 ładowanie dinozaura, 200
 podstawowy plik HTML, 178
 poruszanie postacią, 184
 raycasting widoku z góry, 186
 rejestrwanie danych wejściowych, 183
 rysowanie kolorów i odcieni, 198
 rzucanie promieni, 187

- silnik gry, 29
- style elementów aparatu fotograficznego, 193
- tworzenie mapy dwuwymiarowej, 179
- uatrakcyjnianie świata, 196
- umieszczanie gracza na mapie, 183
- włączenie sepii, 206
- zasoby, 299
- platformowa, 113
 - automatyczne resetowanie, 124
 - budowa mapy kolizji, 121
 - chodzenie i skakanie, 121
 - definicja wygranej, 130
 - dodanie kontenerów na wiadomości
 - i instrukcje, 129
 - dodawanie postaci, 119
 - dodawanie przedmiotów do zbierania, 125
 - dodawanie ziemi, 121
 - edycja mapy, 115
 - ekran tytułowy, 123
 - gameOver, 129
 - informacje, 129
 - inicjowanie aplikacji, 118
 - jednostka EnemyEntity, 127
 - obsługa ruchu gracza, 122
 - obsługa stanu MENU, 124
 - przegrywanie i wygrywanie, 129
 - przycisk dodawania obiektu, 119
 - resetowanie monet, 130
 - silnik gry, 117
 - tworzenie mapy kafelkowej, 114
 - uruchamianie, 116
 - wiązanie klawiszy ruchu, 122
 - wrogowie, 126
 - youWin, 130
 - zakończenie gry, 125
 - załadowanie zasobów, 118
 - zapis danych mapy, 116
 - zasoby, 299
 - zwiększanie mocy postaci, 128
- ROG
 - rysowanie kota, 232
- RPG, 209
 - atakowanie, 240
 - atakowanie i przechodzenie poziomów, 237
 - budowa sceny, 244
 - dodawanie gracza, 214
 - dodawanie gracza i wroga, 242
 - dodawanie warstwy kolizji, 217
 - dodawanie włączęgi, 235
 - działania wojenne, 240
 - ekran stanu, 220
 - etykieta na status gracza, 238
 - funkcja obsługi danych wejściowych, 222
 - interakcja z postaciami, 223
 - magazyn lokalny, 246
 - mówiący kot, 228
 - obsługa początku bitwy, 243
 - odczytywanie danych z magazynu
 - lokalnego, 246
 - odejmowanie punktów zdrowia, 239
 - określanie sprite'a przed graczem, 224
 - opuszczanie sceny bitwy, 244
 - otwieranie sklepu, 230
 - pętla bitwy, 243
 - plik index.html, 210
 - poruszanie gracza, 217
 - procedura obsługi zdarzeń sklepu, 233
 - przeglądarka Chrome, 219
 - przegranie bitwy, 239
 - przygotowanie bitwy, 238
 - przygotowywanie danych
 - do wyświetlania, 221
 - rozmawianie z postaciami z gry, 224
 - rysowanie produktów w sklepie, 232
 - skrótły atrybutów, 222
 - sprite'y przedmiotów, 226
 - stan gracza, 221
 - turowa, 210
 - tworzenie interfejsu bitwy, 235
 - tworzenie mapy, 211
 - tworzenie sklepu, 228
 - ukrywanie etykiety, 222
 - uruchamianie sklepu, 231
 - usunięcie zawartości schowka, 229
 - widoczność informacji o stanie gracza, 226
 - worzenie schowka, 226
 - wygrana w bitwie, 239
 - wykrywanie kolizji, 219
 - wyświetlanie danych, 221
 - wyświetlanie i ukrywanie schowka, 227
 - wyświetlanie opcji walki, 242
 - wywołania funkcji i przypisania własności
 - w sklepie, 235
 - zakup produktu, 234
 - zapisywanie, 245
 - zasoby, 300
- RTS, 249
 - dodawanie sprite'ów, 258
 - informacja o zmianach pozycji, 268
 - kolizje dla destrukcji i sprawdzenia
 - przeciwnika, 270
 - obsługa kliknięć kafelków, 263
 - plik index.html, 255
 - poruszanie jednostkami, 263
 - procedura obsługi połączenia, 266
 - procedura obsługi wiadomości initialize
 - player, 273
 - procedura obsługi wiadomości place units, 273
 - rysowanie jednostek, 259
 - sterowanie gracza, 265
 - tworzenie mapy izometrycznej, 257
 - ustawienie kafelków, 258

- warunek pierwszego kliknięcia, 268
- widoczność, 265
- wysyłanie jednostek miejsc do klienta, 259
- typu, 38
- grupa
 - battle, 238
 - shop, 231
- guard, 82

H

- halfAngularWidth, 206
- Harvest Moon, 38
- hipertekst, 20
- hitbox, 150
- HTML, 20
 - otwieranie pliku w przeglądarce, 21
 - struktura dokumentu, 20
- HTML5 Audio, 301
- HTML5 Game Development, 301
- HTML5 Rocks, 301
- httpserver.js, 253
- Hypertext Markup Language, 20

I

- id, 24
- identyfikator, 24
 - impress, 39
 - player_inventory, 45
- if else, 54
- importowanie
 - plików
 - na stronę, 28
 - zestawu kafelków, 114
- Impress, 298
- impress.js, 29, 38
 - okruszki, 59
- impreza, 65
 - bicie kretów, 80
 - dynamiczne pokazywanie kreta, 79
 - rysowanie dziur, 72
 - rysowanie kreta, 74
 - rysowanie na kanwie, 70
 - rysowanie tła, 71
 - skrypt, 29
 - sprawdzenie trafienia, 82
 - umieszczanie kretów w dziurach, 77
 - ustawienie stanu aktywności dziur, 80
 - zapisywanie wyników, 81
- inicjowanie obiektu inwentarza, 49
- initialize player, 269
- Inkscape, 299
- instrukcja
 - bind.this, 118
 - console.log, 27

- console.log(mójObiekt), 290
- console.log(toCzegoNieRozumiem), 73
- game.constructor, 70
- this.message, 232
- this.nazwaWłasności, 74
- warunkowa, 48
- instrukcje, 283
 - warunkowe, 286
- interakcje z obiektami, 50
- interfejs
 - książka, 38
 - programistyczny, 282
- interpreter, 284, 290
 - komentarze, 285
- interpretery, 66

J

- JavaScript
 - benchmarking, 104
 - definiowanie własności obiektów, 98
 - dodawanie plików, 30
 - dołączanie plików do systemu, 28
 - funkcja, 285
 - główne typy API, 282
 - gra platformowa, 116
 - instrukcje, 283
 - instrukcje warunkowe, 286
 - interpreter, 253
 - język przeglądarkowy, 254
 - komentarze, 287
 - konwersja z CoffeeScript, 68
 - lista numerów klawiszy, 169
 - ładowanie kodu, 26
 - ładowanie skryptu, 89
 - łańcuchy, 284
 - metody API, 49
 - nawiasy, 49
 - notacjaWielbłędzia, 283
 - obiekt, 285
 - ogólna budowa API, 283
 - określanie dostępności zmiennych, 69
 - określanie kolorów, 197
 - operatory, 172
 - opisowe nazwy zmiennych i funkcji, 93
 - pętla, 286
 - przecinki, 117
 - przykładowa gra, 68
 - tablica, 284
 - unobtrusive, 89
 - wartość zwrotna, 49
 - wczytywanie plików, 39
 - wzorce, 89
 - zmienna, 283
- Jaws, 29, 296
- jednostka gracza, 120

jQuery, 298
 jquery.gamequery.js, 29
 jquery.js, 29
 js2coffee.org, 66
 jsfiddle.net, 293
 JSLint, 292
 jsperf.com, 104

K

kafelek
 kolizji, 121
 solid, 121
 kanały IRC, 293
 kanwa, 160
 z dwuwymiarowym kontekstem, 160, 161
 z trójwymiarowym kontekstem, 160, 161
 katalog start, 17
 klasa, 24
 Bitmap, 111
 BitmapAnimation, 111
 correct, 32, 33
 empty, 46
 enemy, 166
 event-text, 51
 inventory-box, 46
 itemable, 45
 item-container, 45
 playerMissiles, 171
 question, 24
 slide, 39, 41
 slide-text, 42
 SpriteSheets, 111
 step, 39
 kod
 błędy programistyczne, 289
 elementy ułatwiające zrozumienie, 93
 oznaczenie, 16
 testowanie, 291
 komentarz, 285, 287
 komponent
 DOM, 258
 grass, 259
 komunikaty diagnostyczne, 290
 konsola, 290
 konstruktor, 138
 Enemy, 164
 obiektu, 69
 Player, 141
 height, 167
 width, 167
 z identyfikatorem formy, 156
 kontekst renderowania kanwy, 124
 kontekst trójwymiarowy, 71
 kontrola jakości, 289
 konwencje typograficzne, 16

konwersja
 kąta na stopnie, 205
 kształt, 108

L

Legend of Zelda, 88
 licencjonowanie oprogramowania, 31
 liczby, 146, 284
 binarne, 147
 dziesiętne, 147
 listy mailingowe, 292
 literał
 game.keys, 80
 local storage, 245
 losowanie kolorów, 91
 Lufia 2, 88

Ł

ładowanie
 zewnętrznego pliku JavaScript, 26
 ładowanie obrazu ekranu jako zasobu, 124
 łańcuchy, 284
 wywołań, 49

M

magazyn lokalny
 relacyjny, 247
 Magic wand, 150
 main.css, 39
 przeciąganie przedmiotów, 45
 ukrywanie treści strony, 26
 main.js
 dodanie butów do puli jednostek, 128
 dodanie monet do puli jednostek, 125
 dodanie wroga do puli jednostek, 126
 dodawanie modułu czcionek, 141
 Maniac Mansion, 38
 map.js, 211
 mapa, 211
 maper kodu, 66
 maski bitowe, 146
 obsługa zdarzeń, 149
 maszyna stanów, 244
 mechanizm
 broadcast, 267
 Melon, 297
 melon.js, 29
 silnik gry, 117
 menedżer pakietów
 npm, 252
 menedżery pakietów, 252
 message, 54

metoda

- add, 49
- addChild(), 90
- addEventListener, 46
- addGroup, 165
- addItem, 57
- addSprite, 165
- arc, 71
- attachEvent, 46
- beginFill(), 91
- beginStroke(), 91
- bitowa, 147
- context.fillText, 74
- context.font, 74
- currentSlide, 56
- deleteltem, 57
- draw, 71
 - bez skalowania, 150
 - modyfikacja, 78
 - uproszczenie, 72
- Draw
 - w pętli, 67
- drawHoles, 73
- drawSquare(), 90
- dropltemInto, 54
- fill, 71
- fillRect, 71
- fillStyle, 71
- game.bop.with_key, 83
- game.drawBackground, 72
- game.screen.draw, 55
- game.slide.SetText, 54
- game.update, 81
- get, 49, 54
- graphics.setStrokeStyle(), 90
- item, 47
- items, 54
- łączenie wywołań w łańcuchy, 165
- Object.create, 69, 70
- prywatna, 56
- publiczna, 49
- querySelectorAll, 46
- rect(), 91
- remove, 49
- rysowanie figur, 72
- stage.update(), 90
- stroke, 76
- update
 - bitowa, 148
 - w pętli, 67
- Minecraft, 38
- Modernizr, 298
- modularyzacja, 50
- module pattern, 50
- Mozilla Developer Network, 282
- Myst, 207

N

- nagłówek
 - h1, 21
- narzędzia
 - przydatne podczas tworzenia gier, 298
- Node, 298
- node package manager, 252
- notacjaWielbłądzia, 283
- Notepad++, 297
- NPM, 298

O

- obiekt, 285
 - bat, 54
 - battle.menu, 238
 - BootsEntity, 128
 - bop, 81
 - buforowanie, 104
 - camera, 194
 - canvas
 - definiowanie, 191
 - modyfikowanie, 197
 - CoinEntity, 126
 - Crafty, 258
 - dino, 200, 205
 - sprite jaws, 205
 - eksperymentowanie w konsoli, 181
 - Enemy, 166
 - EnemyEntity, 126
 - definiowanie, 127
 - forms, 142
 - game, 52, 67
 - wymiary sprite'a, 213
 - Game, 66, 67, 211
 - game.hole, 74
 - Graphics, 92
 - greeter, 225
 - Group, 213
 - hole
 - dotatkowy kod rysowania, 77
 - imgSize, 137
 - inventory, 49
 - inventoryObject, 48
 - kanwy, 191
 - map, 213
 - dodanie danych kolizji, 217
 - mapujący metody publiczne na prywatne, 54
 - minima
 - funkcja draw, 181
 - minimap, 181
 - funkcja draw, 204
 - mole, 75
 - NodeList, 46
 - npc, 225

- obiekt
 - opis przeglądarki, 70
 - palette, 196
 - player, 183, 216
 - atakowanie i przechodzenie poziomów, 237
 - Player, 137
 - atrybut mask, 148
 - funkcja update, 144
 - rejestr naciśnięć klawiszy, 144
 - rejestrwanie danych wejściowych, 145
 - PlayerEntity
 - dodawanie, 121
 - playerInventory, 56
 - potomny
 - tworzenie, 69
 - raycaster, 186
 - modyfikacja, 190
 - modyfikowanie, 201
 - rect, 137
 - reprezentujący kreta, 75
 - Stage, 90
 - surface, 137
 - Ticker, 102
 - tile, 98
 - tileClicked, 98
 - TitleScreen, 123
 - tworzenie
 - konwencje, 138
 - window, 89
 - object, 54
 - obsługa
 - padów do gier, 220
 - raycastingu, 178
 - zdarzeń klawiatury i myszy, 66
 - obszar widoku, 219
 - odblokowanie pytań, 28
 - odtwarzanie dźwięków, 83
 - w przeglądarkach, 84
 - okruszki, 59
 - implementacja, 59
 - Open Game Art, 299
 - open source, 282
 - operacje na bitach, 147
 - operatory bitowe, 147
 - oznaczanie poprawnych odpowiedzi
 - tworzenie stylu, 32
- P**
- pakiet npm, 252, 255
 - parallax scrolling, 131
 - parametr, 285
 - alignment, 262
 - context, 124
 - dt, 79
 - formIndex, 152
 - itemNode, 54
 - message, 56
 - slideld, 56
 - source-overlay, 105
 - target, 54
 - perspektywa izometryczna, 178
 - pętla, 286
 - for, 233
 - w stylu funkcyjnym, 46
 - w stylu proceduralnym, 46
 - pętle, 94
 - Pickle, 299
 - Piętnastka, 88
 - Pixel Joint, 299
 - playground, 162
 - pliki
 - index.html, 17
 - źródłowe, 17
 - pobieranie danych od graczy, 144
 - pobieranie losowego elementu, 97
 - poła kolizyjne, 150
 - polecenie
 - node, 253
 - poradniki, 299
 - procedura
 - dragenter, 48
 - dragleave, 48
 - enterframe, 234
 - onPress, 108
 - procedura nasłuchowa dla przycisku, 221
 - programowanie
 - niskopoziomowe, 88
 - wysokopoziomowe, 88
 - programowanie funkcyjne, 47
 - programy działające po stronie serwera, 250
 - projektowanie gier
 - kierunek badań, 279
 - prototyp obiektu, 69
 - przechowywanie danych, 284
 - przeglądarki, 297
 - bufory, 104
 - przeniesienie
 - fokusu, 24
 - przywracanie do widoku, 26
 - punkty wstrzymania, 291
 - puzzle, 87
 - aktualizacja bufora, 105
 - buforowanie i wydajność, 104
 - dopasowywanie i usuwanie par, 97
 - dopasowywanie par zamiast duplikatów, 106
 - inicjowanie bufora, 104
 - logika wygranej i przegranej, 103
 - Memory, 88, 99
 - obsługa
 - kliknięć, 97
 - przechowywanie czasu gry, 100

- skrypt, 29
- tworzenie kwadratów, 92
- tworzenie par, 94
- ukrycie koloru kwadratów, 99
- ukrywanie i przekręcanie obrazków, 99
- wstępny plik HTML, 88
- wygrywanie i przegrywanie, 100
- wyłączanie buforowania, 105

Q

- quiz, 19
 - dodawanie pytań, 22
 - lista zakupów, 28
 - oznaczanie poprawnych odpowiedzi, 32
 - plik index.html, 20
 - przywracanie pytań do widoku, 26
 - przywrócenie pytań do widoku, 31
 - reagowanie na kliknięcia, 32
 - sprawdzanie odpowiedzi, 24
 - sprawdzenie odpowiedzi, 33
 - styl poprawnych odpowiedzi, 32
 - ukrywanie i pokazywanie, 25
 - ukrywanie pytań, 27
 - wynik porównania odpowiedzi, 34
 - wyświetlenie
 - pierwszego pytania, 28
 - zablokowane pytania, 25
 - zbiór pytań, 20
- QUnit, 291

R

- Raptorize, 298
- ray casting, 177
- raycasting, 178
 - imitacja trójwymiarowości, 190
 - widoku z góry, 186
- receptury, 17
- refactoring, 73
- refaktoryzacja, 73
 - kodu, 137
- renderowanie, 160
 - grafiki, 88
 - kolorów
 - śródliniowo, 106
 - kontrolowanie, 219
 - kwadratów, 94
 - na kanwie, 160
 - technologie, 161
 - przeглядarkowe, 160
 - większej liczby obiektów, 92
- requestAnimationFrame
 - normalizacja, 66
- reset CSS, 41
- resources.js, 117

- dodanie sprite'a monet, 125
- dodanie wroga, 126
- dodanie zasobu boots, 128
- dodawanie gracza, 119
- dodawanie postaci gry, 120
- RGB, 33
- rootScene, 213
- Ruby on Rails, 66
- rysowanie
 - dziur, 72
 - funkcje bibliotek, 76
 - kształtów, 90
 - na elemencie canvas, 70
 - na kanwie, 70, 88
 - na ścieżce, 71
 - sumy trafień, 83
 - tła, 71
 - wykorzystanie obiektów graficznych, 76
 - wyniku, 81
- rzutowanie izometryczne, 178

S

- scena
 - battleScene, 236, 245
- schowek
 - zapełnianie, 49
- screen, 58
- screen.js
 - dodanie obiektu PlayScreen, 117
 - wiązanie klawiszy ruchu, 121
- screens.js, 117
 - ekran tytułowy, 123
 - instrukcje dla gracza, 129
 - usunięcie starych wiadomości, 129
- selektor
 - body, 41
- server.js
 - obsługa połączenia, 266
 - określenie położenia jednostek, 260
- serwer, 250
 - aktualizowanie zmian, 269
 - automatyczne przyjmowanie zmian, 260
 - kod serwerowy, 251, 254
 - komputer użytkownika, 251
 - protokół komunikacyjny, 250
 - Socket.IO, 256
 - zapisywanie zmian, 259
- sessionStorage, 247
- shade, 199
- Shadowgate, 38
- silniki gier, 29
- silnik
 - wykrywanie kolizji, 150
- silnik gry, 296
 - pojęcia i terminy, 228

silniki gier
 uruchamianie, 30
 składanie, 135
 skrypt
 yabble.js, 134
 słowo kluczowe
 super, 67
 this, 73
 var, 118, 283
 Socket.io, 298
 Socket.IO
 procedura nasłuchująca, 262
 sprawdzenie
 dopasowania kwadratów, 99
 sprite, 76
 Sprite Database, 299
 spritesheet, 114
 stage, 213
 statusLabel, 222
 Stratego, 250
 strażnik, 82
 struktura DOM, 160
 obiekty game i stage, 213
 strzelanka, 159
 dodanie
 gracza do planszy, 167
 nowej warstwy, 164
 wrogów, 163
 dynamiczne dodawanie wrogów, 165
 formatowanie pocisków, 172
 kod sterowania pojazdem, 168
 kolizje z pociskami
 udoskonalenie obsługi, 173
 wykrywanie, 170
 kolizje z wrogami, 169
 obsługa kolizji, 169
 początkowy kod HTML, 160
 podstawowe elementy gry, 162
 prędkość pocisku, 170
 przeglądanie funkcji, 174
 silnik gry, 29
 strzelanie, 170
 style statku kosmicznego, 168
 style wrogów, 166
 tworzenie pocisków, 171
 tworzenie pojazdu, 167
 uzupełnianie mocy, 172
 warstwa pocisku, 170
 zasoby, 299
 zmienne statku kosmicznego, 167
 subject, 54
 Sublime Text, 297
 Surface, 216
 SVG, 161
 system kontroli wersji, 252

Ś

ścieżka, 21

T

tablica, 284
 backgroundSlivers, 203
 do przechowywania slajdów, 52
 enemyUnits, 268
 flashcards, 107
 foregroundSlivers, 203
 game.holes, 77
 game.items
 szczegóły przedmiotów, 230
 indeksy, 284
 jednowymiarowa, 108
 map, 181
 maskCache, 151
 budowa, 151
 numberOfTiles, 96
 placementArray, 94
 spriteRoles, 225
 squares
 dodawanie kwadratów, 102
 stepsTaken, 56
 surfaceCache, 137
 textiles, 106
 units, 268
 visibleItems, 228
 walls, 197
 yLocations, 262
 technika
 parallax scrolling, 178
 ray tracing, 178
 rzutowanie izometryczne, 178
 test
 playground, 31
 kodu, 291
 niskopoziomowe, 291
 wysokopoziomowe, 291
 wydajności, 292
 three.js, 301
 TIGSource, 301
 Tiled, 114, 298
 dodawanie postaci, 119
 tworzenie
 mapy kafelkowej, 114
 nowego poziomu, 115
 pozycji startowej, 119
 warstwa
 boots, 128
 coin, 125
 enemy, 126
 kaflekowa, 121
 tileset, 114

tłumaczenie strony na wybrany język, 39
 trueSprite, 265
 tryb pełnoekranowy przeglądarki
 ustawianie, 219
 twierdzenie Pitagorasa, 199
 tworzenie
 dokumentu HTML, 21
 grafiki, 299
 obiektów potomnych, 69
 obiektu z szablonu, 69
 stron internetowych, 89

U

ukrycie pytań, 28
 undefined, 49
 unit, 265
 unitClicked, 264
 usługi sieciowe, 250
 ustawienie
 tła pod tekstem, 108
 ustawienie stanu aktywności, 80

V

viewport, 120, 219
 Vim, 297

W

warstwa
 enemies, 164
 dodawanie sprite'ów, 165
 player
 dodawanie sprite'ów, 167
 pocisku, 170
 warstwa collision, 121
 wartość
 adjustedDistance, 191
 bias, 258
 brightness, 199
 totalCoins, 130
 WebSocket, 255
 wiązanie klawiszy, 80
 wiązanie przycisku
 z funkcją, 234
 wiązanie zdarzeń myszy, 71
 wiersze kodu, 283
 własność
 active, 80
 alive, 272
 collisionData, 220
 color, 272
 constructor, 69
 dino.show, 203
 direction, 216

effects, 54
 frame, 216
 game.things, 53
 hiding, 272
 isMoving, 217
 itemSelected, 232
 nadpisywanie, 98
 name, 54
 node, 164
 opacity, 41
 prototype, 69
 screen, 58
 shades, 197
 spriteOffset, 216
 startingX, 216
 startingY, 216
 type, 272
 walk, 216
 współpraca, 292
 wydajność aplikacji na platformie, 104
 wykrywanie klawiszy, 81
 wypełnianie tła, 109
 wyszukiwanie
 grafiki, 299
 wyświetlanie błędów w konsoli, 39
 wywołanie
 e.preventDefault(), 48

X

XSS, 173

Y

Yabble, 298
 yabble.js, 29
 YAGNI, 89

Z

zamknięcie, 50
 zapisywanie danych
 po stronie klienta, 247
 zasoby, 295
 książki, 300
 portale internetowe, 300
 zdarzenie
 enter, 234
 onload
 wiązanie, 89
 zmiana
 łącza do slajdów, 51
 sposobu odnoszenia do elementów, 51
 zmiana kodu podczas pracy, 269
 zmienianie stron, 41

- zmienna, 283
 - activeGame, 155
 - adjustedAngle, 206
 - angle, 185
 - angleBetweenRays, 186
 - angleInDegrees, 205
 - battle.over, 239
 - color, 96
 - columns, 93
 - controllable, 268
 - counter, 58
 - currentMoleTime, 79
 - definiowanie, 93
 - definiowanie jako niezdefiniowana, 98
 - direction, 185
 - distance, 188
 - draggingObject, 47
 - dX, 188
 - dY, 188
 - enemyHeight, 164
 - enemySpawnRate, 164, 165
 - enemyWidth, 164
 - expMax, 237
 - filtered, 195
 - foregroundData, 212
 - game, 68
 - Game, 67, 68, 69
 - highlight
 - usunięcie, 100
 - hit, 154
 - initialWallColors, 197
 - items, 57
 - mapData, 212
 - max_rgb_color_value, 92
 - maxDistance, 203
 - movementSpeed, 185
 - moveStep, 185
 - numberOfTiles, 95, 96
 - pairIndex, 108
 - parallax, 162
 - percentageDistance, 203
 - placement, 96
 - PlayerEntity, 120
 - potentialWidth, 205
 - rayNumber, 187
 - rows, 93
 - speed, 164, 185
 - squareOutline, 92
 - squarePadding, 93
 - squareSide, 92
 - textTiles, 107
 - tileClicked, 97
 - total, 81
 - turnSpeed, 185
 - twoPi, 188
 - unitInfo, 268
 - units, 266
 - ustawienie na obiekt, 98
 - value, 164
 - wallType, 198
 - wallX, 188
 - wallY, 188
 - xHit, 188
 - yHit, 188
- zmienne
 - globalne, 283
- zmniejszenie szybkości ruchu jednostek, 264
- znacznik, 20
 - <html>, 21
 - HTML, 20
 - viewport, 219
- znaczniki
 - przeglądarki Safari, 219

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**


Odkryj możliwości JavaScriptu i HTML5 w zakresie tworzenia gier!

Popularność języków JavaScript i HTML5 bije wszelkie rekordy. Nikogo to nie dziwi, bo potencjał, jaki drzemie w tych technologiach, jest niewyobrażalny. Efekty, które do tej pory wydawały się nie do osiągnięcia w przeglądarce bez wykorzystania technologii Flash lub apletów Java, teraz można uzyskać bez większego problemu! Dostrzegli to producenci gier i deweloperzy. Zaczęli tworzyć coraz bardziej wymyślne aplikacje i gry działające w środowisku przeglądarki internetowej. Dołącz do tego grona!

Dzięki tej książce to zadanie będzie zdecydowanie łatwiejsze. Należy ona do cieszącej się dużą popularnością serii „Receptury”. W środku znajdziesz szereg artykułów zaczynających się od postawienia pytania, jak wykonać pewne zadanie. Następnym krokiem jest odpowiedź ze szczegółowym omówieniem. W trakcie lektury dowiesz się, jak rysować na kanwie, budować interakcję z użytkownikiem czy stylizować stronę. Ciekawa jest już sama organizacja książki — poszczególne rozdziały zawierają receptury, które pozwolą Ci zbudować konkretne rodzaje gier. Wśród nich znajdziesz platformówkę, strzelankę, RPG oraz strategię czasu rzeczywistego (RTS). Ta książka to doskonały podręcznik zarówno dla osób rozpoczynających przygodę z tworzeniem własnych gier, jak i dla tych, które już się tym zajmują.

SPRAWDŹ:

- jak rysować na kanwie
- jak korzystać z masek bitowych
- jak wykryć kolizję
- jak zbudować niesamowitą strzelankę, platformówkę lub strategię

 Addison-Wesley
Pearson Education

helion.pl
księgarnia
internetowa

Nr katalogowy: 16550



Helion

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

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



KOD KORZYŚCI

ISBN 978-83-246-8042-9



9 788324 680429

Cena: 49,00 zł

Informatyka w najlepszym wydaniu