

Zdobądź nowych użytkowników smartfonów!

Rusz głową!

Mobile Web



Zbuduj raz,
uruchamiaj
wszędzie



Udzielaj większego wsparcia
(swoim użytkownikom)



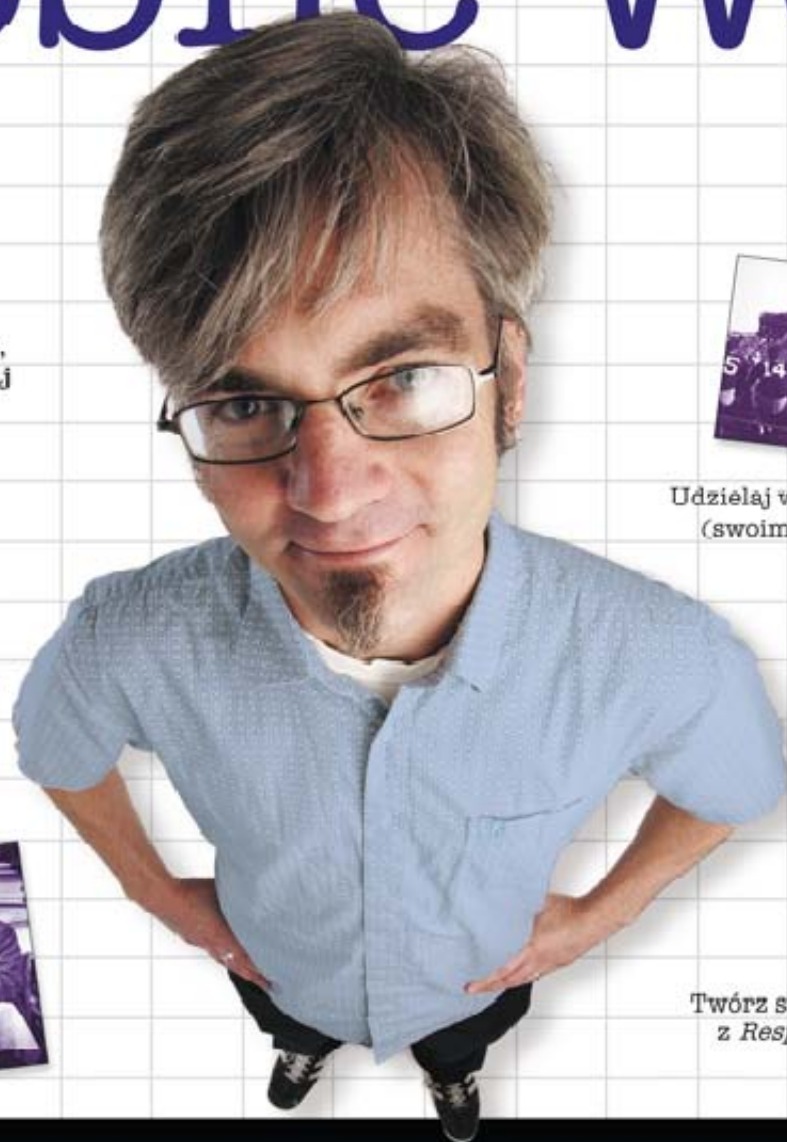
Znajdź drogę
z geolokalizacją



Skieruj swoje
strony na
małоекranową
dieta



Twórz swoje projekty zgodnie
z *Responsive Web Design*



O'REILLY®

Lyza Danger Gardner, Jason Grigsby



Tytuł oryginału: Head First Mobile Web

Tłumaczenie: Aleksander Lamża

ISBN: 978-83-246-4864-1

© 2013 Helion S.A.

Authorized Polish translation of the English edition of **Head First Mobile Web, 1st Edition**
9781449302665 © 2012 Cloud Floor, Inc.

This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

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

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

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

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

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/mobweb>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści (skrótowy)

Wprowadzenie	xxi
1. Wprowadzenie do mobilnych technologii webowych. <i>Wrażliwe projekty, czyli Responsive Web Design</i>	1
2. RWD na poważnie. <i>Koncepcja Mobile First w podejściu Responsive Web Design</i>	43
3. Oddzielna witryna mobilna. <i>Stawiamy czoła niezupełnie sprzyjającym okolicznościom</i>	91
4. Komu wsparcie, komu? <i>Które urzędnicy powinny być obsługiwane?</i>	137
5. Bazy i klasy urzędzeń. <i>Zapoznaj się z grupą</i>	151
6. Framework dla mobilnych aplikacji internetowych. <i>Tartanator</i>	217
7. Mobilne aplikacje w prawdziwym świecie. <i>Wyjątkowe mobilne aplikacje internetowe</i>	267
8. Tworzenie hybrydowych aplikacji mobilnych z PhoneGap. <i>Ustrzel tartan! — w stronę natywności</i>	313
9. Podejście „future friendly”. <i>Odnajdywanie (jakiegoś) sensu w chaosie</i>	357
A Dodatek Ścinki. <i>Sześć najważniejszych spraw (o których nie mówiliśmy)</i>	373
B Dodatek Postaw swój serwer. <i>Gdzieś trzeba zacząć</i>	387
C Dodatek Instalowanie WURFL. <i>Jak wywęszyć urzędnika?</i>	397
D Dodatek Instalowanie SDK i narzędzi dla Androida. <i>Zadbaj o środowisko</i>	403
Skorowidz	417

Spis treści (z prawdziwego zdarzenia)



Wprowadzenie

Twój mózg kontra technologie mobilne. Starasz się czegoś nauczyć, a mózg robi Ci „przystługę”, za wszelką cenę odciągając Twoją uwagę od nauki. Myśli: „Lepiej wyjdź i zajmij się czymś ciekawszym — wypatruj krwiożerczych bestii albo sprawdź, czy kiedy podpalisz swoje BlackBerry Bold, włączy się alarm pożarowy”. Jak w takim razie oszukać mózg, by uznał, że Twoje życie zależy od znajomości technologii mobilnych?

Dla kogo jest ta książka?	xxii
Wiemy, co sobie myślisz	xxiii
Wiemy też, co sobie myśli Twój mózg	xxiii
Metapoznanie — myślenie o myśleniu	xxv
Zespół korektorów merytorycznych	xxx
Podziękowania	xxxi

Wprowadzenie do mobilnych technologii webowych

1

Wrażliwe projekty, czyli Responsive Web Design

Witajcie! Jesteście gotowi na mobilne technologie webowe? Tworzenie witryn na urządzenia mobilne jest naprawdę ekscytujące. Wiele w tym uroku, emocji i momentów, w których chciałoby się wykrzyknąć: *Eureka!* Ale z drugiej strony pełno tu tajemnic i trudności. Technologie mobilne rozwijają się w tak niewiarygodnym tempie, że cały czas jesteśmy trochę w tyle. Trzymaj się więc mocno! Naszą przygodę rozpoczynamy od ciekawego podejścia do tworzenia witryn internetowych, znanego jako **Responsive Web Design (RWD)**. Dzięki niemu będziesz mógł sprawić, by strony wyglądały równie dobrze na wielu różnych urządzeniach mobilnych i, co ważne, przydadzą Ci się umiejętności, które już masz.

index.html



styles.css



Wszyscy jedziemy na tym samym wózku. Wskakujesz?	2
Coś niedobrego stało się w drodze do pubu	4
Skoro przeglądarki w telefonach komórkowych są takie świetne...	5
...to czy nie powinno to po prostu działać?	6
Wrażliwe projekty — Responsive Web Design	10
Różne arkusze stylów w różnych sytuacjach	12
Zapytania o media w CSS	13
Dotychczasowa struktura witryny pubu Pod Paradnym Morsem	15
Analiza dotychczasowego arkusza CSS	16
Co trzeba zmienić?	17
Szukamy stylów wymagających zmiany	18
Droga do stylów dostosowanych do urządzeń mobilnych	19
Co jest nie tak z układami o stałej szerokości?	26
Dlaczego płynne jest lepsze?	27
Wzór płynności	28
Ciąg dalszy przekształceń	29
Przełączanie kontekstu	31
Co się stało z tymi obrazami?	32
Płynne obrazy	33
Pamiętaj, by być wrażliwym	36
Oto strona w stylu RWD!	40
Podejście Responsive Web Design to również stan umysłu	41

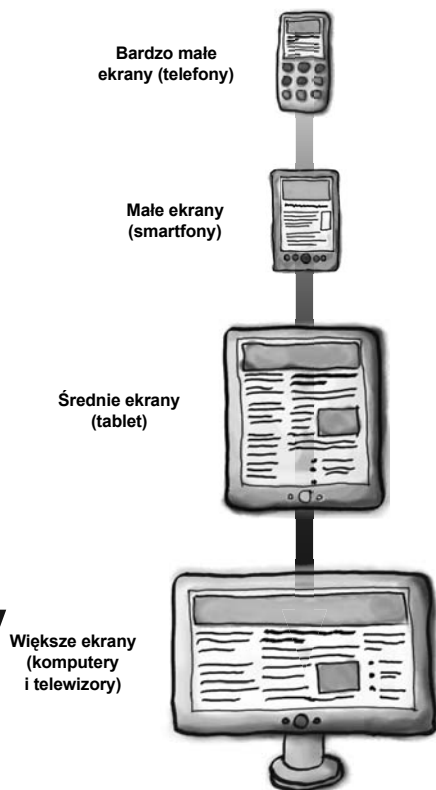
RWD na poważnie

2

Koncepcja Mobile First w podejściu Responsive Web Design

Oto śliczna mobilna witryna. Ale nie oceniaj jej tylko po pozorach. Pod tą piękną powłoką znajdziesz bowiem coś zupełnie innego. Być może wygląda jak mobilna witryna, ale to wciąż zwykła, desktopowa witryna, z tym że przebrana w mobilne ciuszki. Jeśli chcesz, żeby na urządzeniach mobilnych chodziła jak dobrze naoliwiona maszynka, musisz zastosować zasadę **Mobile First**. Jednak najpierw musimy przeprowadzić sekcję obecnej witryny, by odnaleźć ukrywający się w jej wnętrzu desktopowy szkielet. Następnie gruntownie posprzątamy i zaczniemy pracować na świeżo, zgodnie ze strategią **stopniowego ulepszenia**, zaczynając od budowania podstawowych elementów, a kończąc na bogatej wersji desktopowej. Gdy skończymy, nasza strona będzie zoptymalizowana pod każdą możliwą rozdzielczość ekranu.

Stopniowe ulepszenie bazujące na rozmiarze ekranu oraz możliwościach klienta



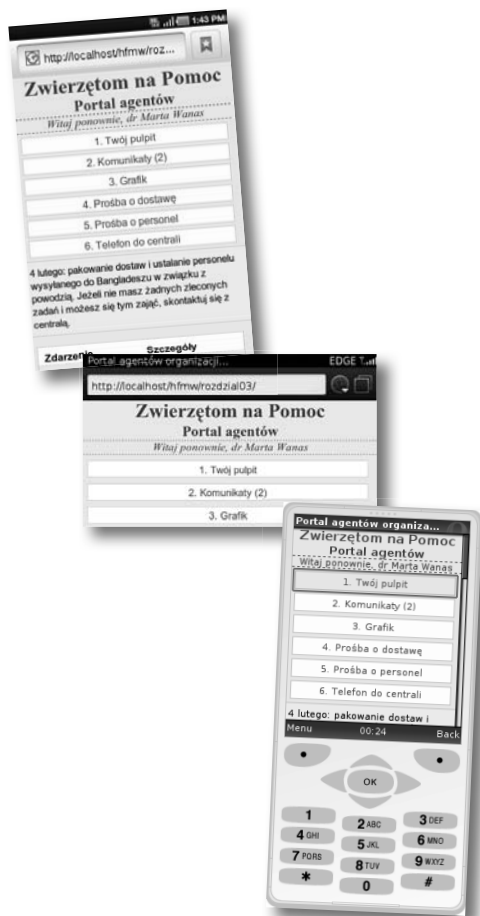
Gdy właśnie zamierzałeś zacząć świętować swój sukces...	44
Czy to naprawdę jest problem? Skąd to wiadomo?	45
Co zrobić, gdy nie śmiga?	47
Nie ma co się oszukiwać, to jest WIELKA strona	48
Dobrodziejstwa pliku HAR	49
Wyteż wzrok i znajdź zawalidrogę	51
Skąd pochodzi skrypt map Google'a	53
Wygląda przyjaźnie, ale takie nie jest	55
Koncepcja Mobile First w podejściu Responsive Web Design	56
Na czym polega stopniowe ulepszenie?	57
Poprawiamy pływanie elementów	60
Zapytania o media w technice Mobile First	61
Niespodzianka! W Internet Explorerze strona się rozsypała	62
Problemy z jednym atrybutem src	68
Powiększanie w znaczniku <meta> viewport	72
Czy powinno się umożliwiać skalowanie?	73
Z pomocą JavaScriptu przywracamy mapę	74
Budujemy pseudozapytanie o media w JavaScriptcie	76
Wstawiamy skrypt na stronę	77
Ten widżet nie jest zgodny z RWD	79
Przenosimy atrybuty do CSS	80
Usuwanie atrybuty z JavaScriptu	81
Mapa znów zasłania treść strony	83
Niech prowadzi Cię zawartość strony	84
Wartości graniczne przybywają na ratunek	87

3

Oddzielna witryna mobilna

Stawiamy czoła niezupełnie sprzyjającym okolicznościom

Wizja jednego, wrażliwego projektu witryny jest cudowna... Mamy tylko jeden układ strony opracowany w zgodzie z koncepcją Mobile First, który dopasowuje się do specyfiki różnych przeglądarek i urządzeń. Brzmi pięknie. Co się jednak stanie, gdy tę wizję pripravimy choćby szczyptą realizmu? Niezaktualizowane systemy, stare urządzenia lub ograniczenia budżetu klienta to tylko kilka przykładów. A co, jeśli zamiast łączyć wersję desktopową z mobilną, chciałbyś je rozdzielić? W tym rozdziale przyjrzymy się szczegółowo **wykrywaniu użytkowników korzystających z urządzeń mobilnych, wspieraniu starszych telefonów i tworzeniu odrębnych witryn dla urządzeń mobilnych.**



Agenci organizacji Zwierzętom na Pomoc patrolują pola	92
Jak agenci mogą otrzymywać i przekazywać informacje?	93
Wysyłamy mobilnych użytkowników na zoptymalizowaną witrynę	96
Jak wywęszyć mobilnych użytkowników?	97
Rozpoznajemy agenta użytkownika	98
Łańcuch user-agent — dzieło szatana?	101
Mówiąc wprost — większość dużych witryn ma swoją wersję mobilną	104
Kiedy jedynym rozwiązaniem jest przekierowanie...	105
Rzuć okiem na skrypt	106
Jak działa skrypt?	107
Przygotowujemy makietę wersji mobilnej	108
Specjalna dostawa... spraw komplikujących życie	110
Nie wszystkie telefony to smartfony...	113
Prostota przede wszystkim — poznaj XHTML-MP	114
Dlaczego chcemy użyć tak starego rozwiązania?	115
XHTML-MP pomaga unikać problemów	116
Przy okazji — przewijanie jest do bani	119
Ostatni problem	119
Klawisze dostępu w akcji	123
Co poszło nie tak?	124
Naprawiamy błędy	125
CSS dostosowany do przeglądarek mobilnych	127
Hm... czegoś tu brakuje	132
Bardzo nam brakuje tych przycisków!	133
Wielki sukces!	134

Komu wsparcie, komu?

Które urzędnicy powinny być obsługiwane?

4

Przetestowanie witryny na wszystkich urządzeniach trwałoby wiecznie.

Musisz ustalić granicę wyodrębniającą urzędnicy, które zamierzasz wspierać. **Ale jak podjąć tę decyzję?** Co z użytkownikami korzystającymi z telefonów, które już dawno powinny trafić na złomowisko? W jaki sposób stworzyć witrynę, by działała na urządzeniach, o których w ogóle nie słyszałeś? W tym rozdziale przygotowujemy magiczną miksturę złożoną z **wymagań projektowych** i **informacji o odbiorcach**. Pomoże nam ona zdecydować, **które urzędnicy mamy wspierać** i co zrobić z tymi, których nie wspieramy.

Skąd wiedzieć, gdzie ustalić granicę?	138
Odejdź na chwilę od komputera	139
Urzędnicy, których <i>nie wspierasz</i> , kontra te, których <i>nie możesz wspierać</i>	140
Zadawaj dużo pytań o swój projekt	142
Składniki magicznej mobilnej mikstury	144
Zajrzyj do szafki z narzędziami i danymi	145
Skąd mam wiedzieć, czy moi klienci używają odpowiednich urządzeń?	150



Bazy i klasy urządzeń

Zapoznaj się z grupą

5

Podczas wybierania wspieranych urządzeń nie wzięliśmy pod uwagę kilku dokuczliwych problemów. Jak mamy się dowiedzieć wystarczająco dużo o mobilnych przeglądarkach użytkowników, by przed dostarczeniem treści spełnić ich oczekiwania? Jak uniknąć tworzenia tylko podstawowej zawartości odpowiadającej najmnijemu wspólnemu mianownikowi urządzeń? No i jak, pozostając przy zdrowych zmysłach, zapanować nad tym wszystkim? W tym rozdziale wkroczymy w świat **możliwości urządzeń** i nauczymy się korzystać z **baz danych urządzeń**, by wreszcie odkryć, jak te wszystkie urządzenia grupować w **klasy**.



Przycisk awaryjny dla spanikowanych studentów	152
Źródła danych o urządzeniach mobilnych spieszą na ratunek	154
Poznaj WURFL	155
WURFL i jego możliwości	156
WURFL — sprytny interfejs API	159
My też możemy zbudować eksplorator	160
Eksplorator — przygotowanie środowiska	161
Szast-prast i eksplorator ulepszony	168
Czas zaprząć możliwości do pracy	170
Korzystamy z WURFL do zróżnicowania zawartości stron	170
Inicjalizacja obiektu urządzenia i przygotowanie danych	172
Czy to jest urządzenie mobilne?	172
Dzięki WURFL strona staje się sprytniejsza	176
Przycisk awaryjny — tylko w telefonach	177
Klasy urządzeń	181
Kolejny lukratywny kontrakt z firmą DaRadę!	182
Jak strona główna wygląda przez mobilne okulary?	183
Zdefiniowanie odmian witryny w zależności od wymagań	184
Przybliżamy klasy urządzeń	185
Zapoznajemy się z funkcją dopasowującą	191
O co chodzi w tej instrukcji switch?	192
Używamy funkcji dopasowującej do testowania możliwości	193
Wypełniamy luki w testach klas urządzeń	200
Musimy bardziej zadbać o bezpieczeństwo	211
Lepiej zapobiegać, niż leczyć	212

Framework dla mobilnych aplikacji internetowych

Tartanator

6

„**My chcemy aplikację!**”. Jeszcze rok czy dwa lata temu tego typu hasło wiązało się nieodłącznie z jednym — tworzeniem natywnych aplikacji dla każdego urządzenia, które zamierzaliśmy wesprzeć. Na szczęście teraz nie jest to jedyne możliwe rozwiązanie. Aplikacje internetowe dla mobilnych przeglądarek są coraz doskonalsze, zwłaszcza ostatnio, kiedy wkroczył **HTML5** wraz z nieodłącznymi kompanami — **CSS3** i **JavaScriptem**. W świat mobilnych aplikacji internetowych wejdzmy wraz z **mobilnym frameworkiem**, czyli zbiorem gotowych rozwiązań programistycznych upraszczających i przyspieszających tworzenie aplikacji.

Hm... całkiem ładne, ale czy moglibyście nad tym jeszcze popracować, by zachowywało się jak prawdziwa natywna aplikacja?



HTML5, aplikacja internetowa... Co te słowa znaczą?	219
Jak się zachowują klasyczne witryny internetowe?	220
Jak się zachowują witryny przypominające aplikacje?	221
Plan pierwszej fazy projektu Tartanator	224
Po co używać frameworków?	225
Dla projektu Tartanator wybraliśmy framework jQuery Mobile	226
Tworzenie prostej strony z jQuery Mobile	228
Kod pozostałych elementów strony	229
Atrybuty data-*	231
Odsyłacze do wielu stron w jQuery Mobile	234
Sedno Tartanatora — tartany jako takie	240
Wrzucamy pozostałe tartany	243
Filtrowanie i porządkowanie listy	244
Dodajemy pasek narzędzi w stopce	249
Upiększamy pasek narzędzi	250
Finalizowanie pracy nad strukturą	251
Czas na przygotowanie formularza do tworzenia tartanów	253
Tłumaczymy wzory tartanów na formularz	255
Tworzymy formularz w HTML5	256
Dodajemy podstawowe pola	257
Dodawanie kolorów umożliwiając użytkownikom listy w listach	258
Pary kolor – rozmiar: pole wyboru koloru	259
Pary kolor – rozmiar: pole rozmiaru	260
Odsyłacz do formularza	262



Mobilne aplikacje w prawdziwym świecie

7

Wyjątkowe mobilne aplikacje internetowe

Mobilne aplikacje są jak dzieci w klasie. Wiesz, chodzi o tę mieszankę fascynacji, przekonania o tym, że można zrobić niesamowite rzeczy, ale też tajemniczości i niepoahamowanego rozrabiactwa. Staramy się trzymać pod kontrolą tych nadpobudliwych geniuszy, ustalając granice, a także dbając, by ich nie przekraczali. Przychodzi jednak czas zbierania korzyści z naturalnych talentów mobilnych aplikacji internetowych. Możemy zastosować metodę **stopniowego ulepszania**, by dopieścić interfejs na potrzeby lepiej rozwiniętych przeglądarek, a problemy z ciągłością dostępu do internetu rozwiązać za pomocą **trybu offline**. Możemy też wydobyć esencję mobilności, korzystając z **geolokalizacji**. Nie ma czasu do stracenia, zabierzmy się za tworzenie wyjątkowych mobilnych aplikacji internetowych!



Wygląda niezłe...	268
Aplikacje mobilne w prawdziwym świecie	270
Na miejsca, gotowi, ulepszać!	274
Ulepszamy formularz	275
Widżet zarządzający listą kolorów i rozmiarów	276
Zaglądamy pod przykrywkę	277
No tak, to było ulepszanie frontendu	278
...a teraz pora na backend	280
Dwie twarze skryptu generate.php	281
Jeszcze tylko jedno...	282
Tryb offline to ważna sprawa	284
Prosty przepis na plik manifestu	285
Na ratunek przybywają narzędzia	286
Udostępniaj pliki manifestu z prawidłowym nagłówkiem content-type	287
Zwyciężyliśmy (w końcu)	297
Jak działa geolokalizacja?	298
Jak poprosić przeglądarkę o dane geolokalizacyjne?	299
Początek pracy nad stroną Znajdź wydarzenie — podstawy	301
Dołączamy geolokalizację	303
Nic nie znalazł	309

Tworzenie hybrydowych aplikacji mobilnych z PhoneGap

Ustrzel tartan! — w stronę natywności

8

Czasem musisz się zdecydować na aplikację natywną. Być może potrzebujesz dostępu do czegoś, co nie jest (jeszcze) osiągalne z poziomu przeglądarki. A może klient chce, by aplikacja *koniecznie* znalazła się w sklepie. Nie możemy się już doczekać chwili, gdy przeglądarka będzie udostępniała wszystko, co jest potrzebne, by tworzyć pełnoprawne aplikacje mobilne. Jednak zanim do tego dojdzie, możemy skorzystać z możliwości **hybrydowego podejścia** — nadal będziemy tworzyć aplikacje z wykorzystaniem **technologii internetowych**, ale użyjemy **biblioteki, która pełni rolę mostu** między naszym kodem a natywnymi możliwościami urządzeń.

Międzyplatformowe natywne aplikacje zbudowane w oparciu o technologie internetowe? Brzmi całkiem niezłe!

Nowe możliwości	314
Jak działają aplikacje hybrydowe?	317
Budowanie mostu za pomocą PhoneGap	318
Dołącz do PhoneGap Build	321
Jak ma działać aplikacja?	322
Śledzenie ustrzelonych tartanów	323
Anatomia projektu Ustrzel tartan!	324
Pobieranie utworzonej aplikacji	328
Wybierz drogę	329
Kto co widział? Zapisujemy znalezione tartany	334
W czym nam może pomóc localStorage?	335
Sprawdzamy, co obsługuje przeglądarka	339
Używamy funkcji wyświetlającej znalezione tartany	340
Metody toggle i toggleClass	341
Znalazłeś tartan? Udowodnij to!	344
Zapręgamy PhoneGap do robienia zdjęć	345
Integracja z PhoneGap jest już prawie gotowa	347
Teraz jesteśmy gotowi na zgłębienie API mediaCapture	348
W jaki sposób obsłużymy akcję zakończoną powodzeniem?	349
W praktyce zawsze wygląda to trochę inaczej	350
Odrobina anonimowości	351
Już ostatnia sprawa!	353
Daliśmy radę!	354



Most
hybrydowych
aplikacji

VIRGIN 3G

4:20 PM

**Ustrzel
tartan** 

ODSZUKAJ TARTANY I WYGRAJ
WYCIECZKĘ DO EDYNBURGA!

Podejście „future friendly”

Odnajdywanie (jakiegoś) sensu w chaosie

Responsive Web Design. Wykrywanie urządzeń. Mobilne aplikacje internetowe. PhoneGap. Chwila... czego właściwie powinienem użyć?

Obecnie istnieje wiele metod tworzenia aplikacji za pomocą technologii mobilnych. Bardzo często w projektach łączy się wiele technik. Nie ma jednego, najlepszego rozwiązania, ale nie przejmuj się, ponieważ kluczem do sukcesu jest nadążanie za rozwojem technologii. **Nie bój się wyzwania.** Wystarczy przyswoić **podjęcie „future friendly”** i dać się ponieść fali, będąc przeświadczonym o swojej elastyczności i gotowości na wszystko, co przyniesie przyszłość.



I co teraz?	358
Czas rozwiać zbiorowe złudzenie co do sprawowania kontroli	361
Manifest „future friendly”	362
Jeśli nie możesz się uodpornić na przyszłość, zaprzyjaźnij się z nią	364
Dzisiaj aplikacja, jutro witryna	365
Czeka nas długa droga. Przyda się kilka wskazówek	366
Mieszmamy mobilne składniki	369
Spójrz w przyszłość	371

* PRECYZYJNE OGNISKOWANIE *

Nie wystarczy mieć się uśmiech na każdym urządzeniu. Aby poradzić sobie w świecie wciąż rosnącego skumulowania urządzeń, musimy się skupiać na najważniejszych – z punktu widzenia klientów – grawcach, i nie chodzi tu o rozwiązań opartych na najpopularnym wspólnym mianowniku, ale o tworzenie treści i usług mających znaczenie. Użytkownicy są w coraz większym stopniu zniechęceni rosnącym strumieniem informacyjnym i koniecznością szukania sposobów na uproszczenie rzeczywistości. Musisz precyzyjnie zdefiniować swoje usługi, zanim klient i rozwiązań różnorodność zrobią to za Ciebie.

* ORBITOWANIE WOKÓŁ DANYCH *

Ekosystem urządzeń wymaga możliwości przesyłu międzyplatformowych i wydajnych mechanizmów wymiany danych. Dostosuj się do istniejących, ale i dopilnuj co powstających możliwości, definiując dane w taki sposób, by:

- Były możliwy wielokrotny (i elastyczny) dostęp do danych.
- Były stosowane międzyplatformowe standardy.
- Dane były nakierowane na długoterminową integralność.
- Dane zawierały znaczące i katalizowały do całej zawartości.
- Były włączone w operacje odczytu i zapisu.

* UNIWERSALNA ZAWARTOŚĆ *

Nadrzędnym nakazem jest tworzenie dobrze ustrukturyzowanej zawartości. Żądań się, jak opracowana zawartość będzie się zachowywała w różnych kontekstach, uwzględniając ich ograniczenia i możliwości. Bądź odważny i korzystaj z nowych możliwości, ale pamiętaj, że przyszłość może podjąć w różnych kierunkach.

Na naszą przyszłość składają się żywno zawiązane urządzenia o ogromnych możliwościach, jak i proste urządzenia z minimalną funkcjonalnością. Tworząc strukturę zawartości, bierz to pod uwagę.

* IDENTYFIKACJA OKRĘTÓW WIDMO *

Uwzględnienie w projekcie wszystkich możliwych konfiguracji urządzeń jest nie lada wyzwaniem. Proces adaptacji można uprościć, stosując wyskokopozycyjne i dobrze dobrane story standardów dla różnych typów urządzeń. Standardy te można uzupełnić szczegółowymi informacjami o profilu urządzenia.

Tego typu taksonomia może pomóc uszeregować obecnie dostępne urządzenia różnych producentów, pozwalając na dołączenie nowych typów urządzeń, które pojawią się w przyszłości.

* DOWODZENIE FLOTA *

Korzystanie z wielu różnych urządzeń w codziennym życiu pozwala nam rozdzielać zadania i informacje pomiędzy nimi. Gdy jakeś zadanie jest zarządzane przez kolekcję urządzeń, każde z nich może zastosować interakcje, z którymi radzi sobie najlepiej. Dzięki temu nie musimy dostarczać wszystkich aspektów danego zadania do każdego urządzenia, ponieważ nawiązujemy się raczej do pracy w ekosystemie możliwości urządzeń.

Ścinki

A

Sześć najważniejszych spraw (o których nie mówiliśmy)

Czujesz się, jakby coś Ci umknęło? Wiemy, co masz na myśli... Zawsze jest tak samo — myślisz, że to już koniec, a okazuje się, że jest tego więcej. Nie mogliśmy się pohamować, by nie przekazać Ci kilku dodatkowych szczegółów, o których nie wspomnieliśmy w treści książki. Gdybyśmy chcieli napisać o wszystkich ciekawych sprawach, książkę musiałbyś transportować w pancерnej walizce na kołach. Rzuć okiem, co dla Ciebie wybraliśmy.

1. Testowanie na urządzeniach mobilnych	374
2. Zdalne debugowanie	376
3. Sprawdź, co obsługują przeglądarki	382
4. Interfejsy API urzędzeń	384
5. Sklepy z aplikacjami oraz dystrybucja	385
6. RESS: REsponsive design + komponenty Server-Side	386

Postaw swój serwer

B

Gdzieś trzeba zacząć

„Mobilny internet” nie istnieje bez słowa „internet”. Bez dwóch zdań. Jeśli chcesz się zająć tworzeniem witryn i aplikacji mobilnych, będziesz potrzebował serwera WWW. Prędzej czy później dojdzie do sytuacji, w której będziesz potrzebował części serwerowej swojej aplikacji. Możesz wtedy skorzystać z bezpłatnego lub komercyjnego hostingu albo uruchomić serwer na swoim komputerze. W tym dodatku opiszemy proces **stawiania lokalnego serwera WWW z obsługą PHP** z wykorzystaniem bezpłatnego oprogramowania.

Czego będziesz potrzebował?	388
Dostępny tylko lokalnie	389
Windows i Linux — zainstaluj i skonfiguruj XAMPP-a	390
Ciąg dalszy XAMPP-a	391
Na koniec Mac — MAMP	392
Sprawdź, czy zadokowałeś we właściwym porcie	393
Dostań się do swojego serwera	394
Informacje od phpinfo	396

Instalowanie WURFL

Jak wywęszyć urządzenia?



Pierwszy krok do rozwiązania tajemnicy wykrywania urządzeń wymaga trochę zachodu. Każdy przyzwoity gliniarz wie, że trzeba zbierać poszlaki i przesłuchiwać świadków. Musimy zacząć od mózgu całej operacji, którym jest **WURLF API dla PHP**. Później przyjdzie kolej na mięśniaka — jeden **plik XML** zawierający informacje o możliwościach tysięcy urządzeń. Nie będzie jednak łatwo zmusić ich do współpracy, więc będziemy musieli poświęcić im trochę czasu i pogrzebać w **konfiguracji**.

Skąd wziąć mózg?	398
A co z mięśniakiem?	399
Jak zmusić tę dwójkę do współpracy?	400
Czas na porządki w systemie plików	401
Zwróć na to uwagę!	402

Instalowanie SDK i narzędzi dla Androida

Zadbaj o środowisko



Aby gruntownie testować natywne aplikacje pod Androida, musisz przygotować sobie odpowiednie środowisko. Musisz dołączyć swój komputer do małego ekosystemu, do którego zagonisz wszystkie aplikacje Androida uruchamiane zarówno na wirtualnych (emulowanych), jak i rzeczywistych urządzeniach. Abyś mógł stać się pasterzem tego stada, pokażemy Ci, jak pobrać **SDK (ang. *Software Development Kit*) Androida**, jak zainstalować niektóre **narzędzia dla tej platformy**, jak **utworzyć wirtualne urządzenia** i jak **instalować oraz odinstalowywać aplikacje**.

Pobieramy SDK Androida	404
Wybierz odpowiednie narzędzia	405
Tworzenie nowego urządzenia wirtualnego	408
Znajdź właściwą ścieżkę	413



Skorowidz

417

7. Mobilne aplikacje w prawdziwym świecie

Wyjątkowe mobilne aplikacje internetowe

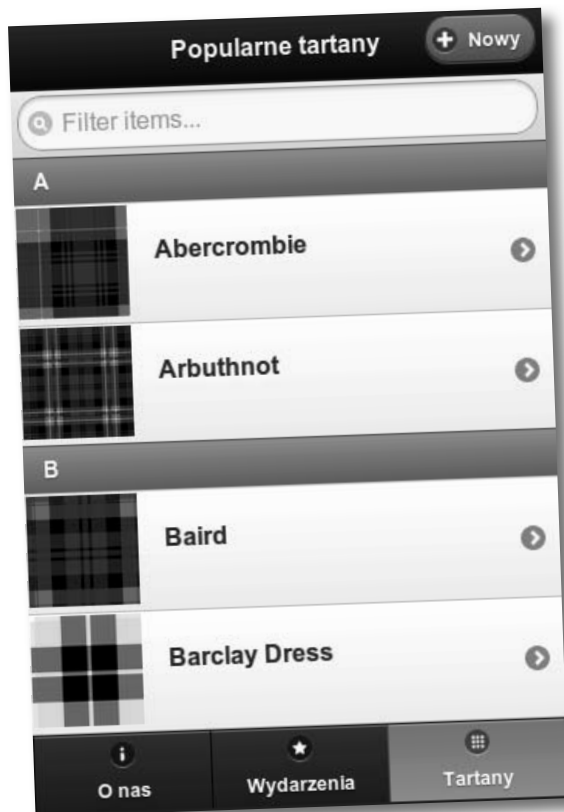
Jeszcze raz mnie pociągniesz za warkoczyki, to popamiętasz! Nie masz się gdzie schować. Znajdę cię wszędzie...



Mobilne aplikacje są jak dzieci w klasie. Wiesz, chodzi o tę mieszankę fascynacji, przekonania o tym, że można zrobić niesamowite rzeczy, ale też tajemniczości i niepojętego rozrabiactwa. Staramy się trzymać pod kontrolą tych nadpobudliwych geniuszy, ustalając granice, a także dbając, by ich nie przekraczali. Przychodzi jednak czas zbierania korzyści z naturalnych talentów mobilnych aplikacji internetowych. Możemy zastosować metodę **stopniowego ulepszania**, by dopieścić interfejs na potrzeby lepiej rozwiniętych przeglądarek, a problemy z ciągłością dostępu do internetu rozwiązać za pomocą **trybu offline**. Możemy też wydobyć esencję mobilności, korzystając z **geolokalizacji**. Nie ma czasu do stracenia, zabierzmy się za tworzenie wyjątkowych mobilnych aplikacji internetowych!

Wygląda nieźle...

Po zakończeniu pierwszej fazy projektu Tartanator wygląda już jak prawdziwa mobilna aplikacja internetowa.



Pusta gadanina...

Są przyciski. I pasek nawigacji. Nie zabrakło też fajnych efektów. Ładujemy fragmenty zawartości za pomocą AJAX-a. Redukujemy obciążenie łącza i liczbę żądań oraz ograniczamy przetwarzanie struktury DOM do minimum. Framework jQuery Mobile pomaga nam dostosować elementy formularza przygotowanego w HTML5 do możliwości urządzeń mobilnych.



Stylowa, to prawda.
Mobilna – jak najbardziej.
Szkopuł w tym, że nic nie
robi! Czy aplikacje nie
powinny czegoś robić?

Spokojnie. Może się wydawać, że pierwsza faza prac nad projektem była pełna spektakularnych akcji, a tu nic. Jednak trzeba uczciwie przyznać, że położyliśmy całkiem solidne fundamenty pod aplikację.

W tej chwili mamy prostą strukturę opartą na HTML5. Czas przejść na kolejny poziom!

...a tu potrzeba działania!

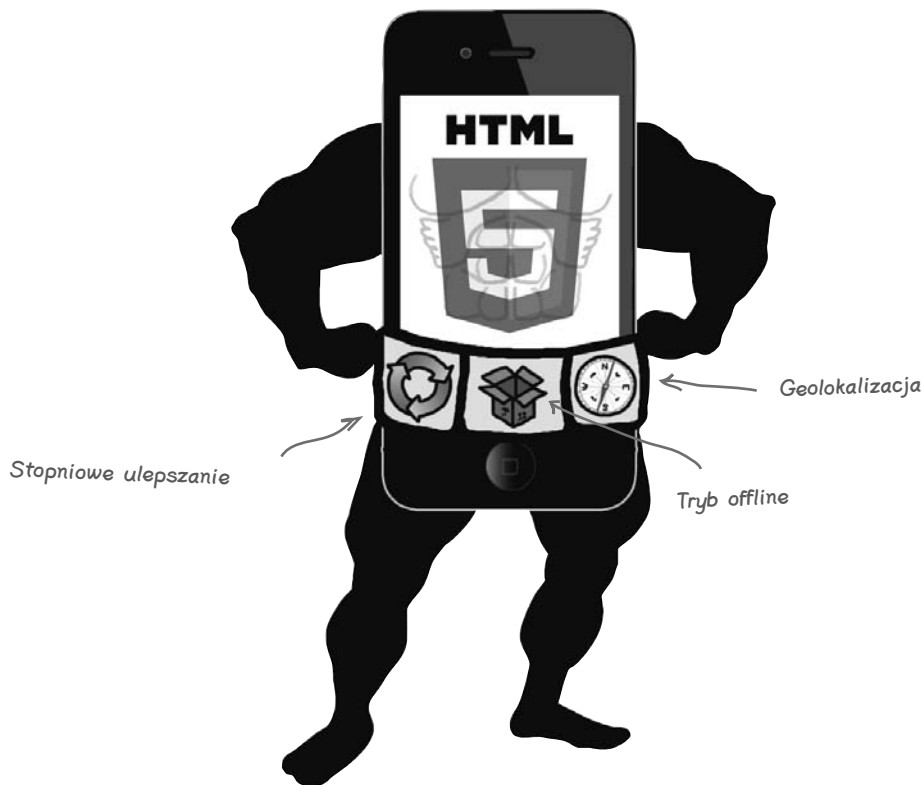
W drugiej fazie prac będziemy bazować na tym, co udało nam się stworzyć w pierwszej, i zamienimy solidną (choć nie do końca funkcjonalną) aplikację internetową w to, co można by nazwać wyjątkową mobilną aplikacją!

Aplikacje mobilne w prawdziwym świecie

Mobilne aplikacje internetowe, które korzystają z dobrych wzorców mobilnego świata, mają kilka cech wspólnych.

To, co można nazwać *wyjątkowymi mobilnymi aplikacjami internetowymi*, to aplikacje mobilne dostosowane do wymogów prawdziwego świata. Dopasowują się do możliwości konkretnych urządzeń użytkowników, korzystając z techniki **stopniowego ulepszania**. Jeśli chwilowo użytkownik nie ma dostępu do internetu, działają dalej w **trybie offline**. W wielu sytuacjach używają **geolokalizacji** oferowanej przez przeglądarki mobilne, by dostarczać treści powiązane z lokalizacją użytkownika.

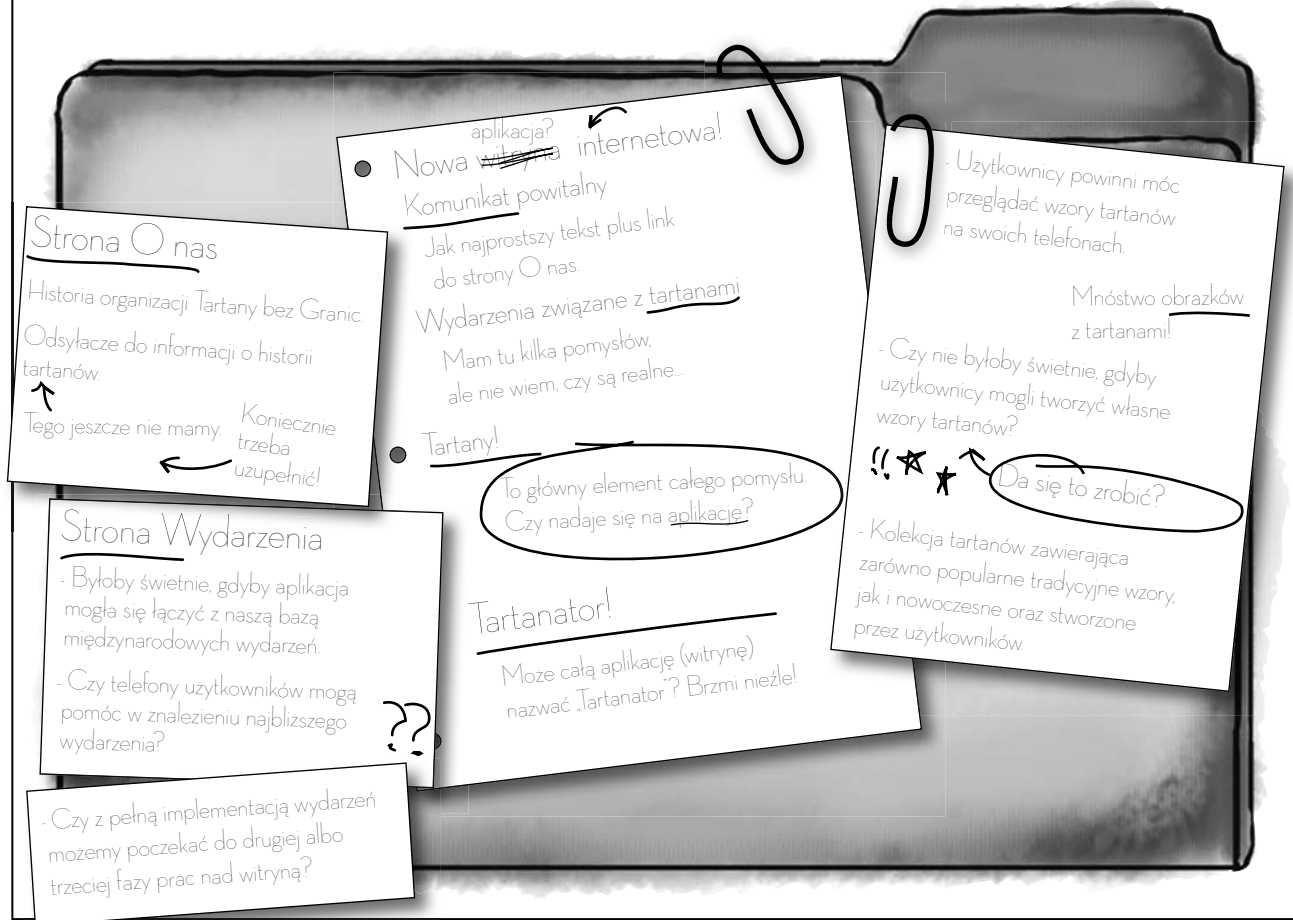
Wyjątkowe mobilne aplikacje internetowe wiedzą wszystko o trzech wymienionych cechach. W tym rozdziale przyjrzymy się im bliżej, dzięki czemu będziemy mogli tworzyć jeszcze lepsze aplikacje, biorąc to, co najlepsze, ze świata mobilności.



Zaostrz ołówki

Co zrobimy, żeby zakończyć drugą fazę prac i sprawić, by Tartanator stał się wyjątkową mobilną aplikacją internetową?

Przede wszystkim musimy się wziąć za to, co zostało do zaimplementowania zgodnie ze specyfikacją, którą podał Ewan w rozdziale 6. Zaznacz te wymagania, którymi się jeszcze nie zajęliśmy. Za chwilę sformułujemy na ich podstawie kluczowe zadania dla drugiej fazy projektu.



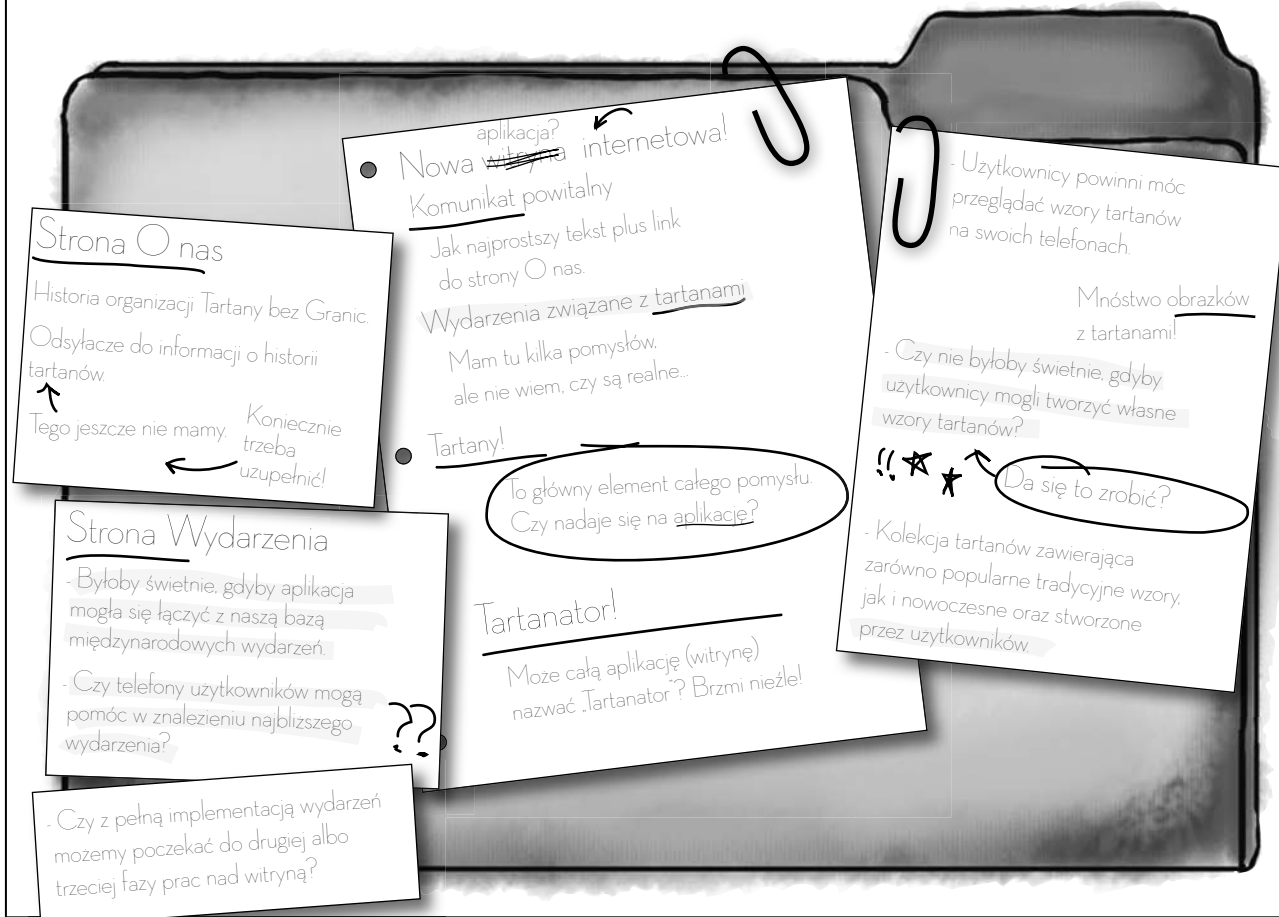
Cele drugiej fazy projektu Tartanator



Zaostrz ołówki. Rozwiązanie

Kluczowe zadania drugiej fazy projektu tak naprawdę służą osiągnięciu dwóch głównych celów:

- 1 Dokończenie prac nad edytorem własnych tartanów.**
Mamy już prototyp formularza. Teraz musimy przejść do konkretów i sprawić, by formularz zaczął działać. Musimy też ulepszyć formularz, by był bardziej użyteczny w lepszych smartfonach (przy okazji dodamy kilka wodotrysków).
- 2 Stworzenie dynamicznej strony Wydarzenia, w której da się wyszukiwać i która „zna” lokalizację użytkownika.**
Musimy umożliwić wyszukiwanie wydarzeń powiązanych z tartanami, które odbywają się najbliżej aktualnej lokalizacji użytkownika. W tym celu skorzystamy ze źródła danych z informacjami o wydarzeniach oraz geolokalizacji.





Łukasz: To pierwsze duże wymaganie wygląda groźnie, nie sądzicie? Może rozbijemy je na mniejsze zadania?

Kuba: Ale chwila, zanim się tym zajmiemy, co z obiecanyimi tekstami na stronę O nas i opisem historii tartanów?

Łukasz: A, właśnie! Sytuacja wygląda tak, że żona człowieka, który miał napisać te teksty dla Tartanów bez Granic, właśnie urodziła. Świeżo upieczony tatuś obudził się, nomen omen, z ręką w nocniku — nic wcześniej nie napisał, a teraz odkrył, że dziecko jest bardziej zajmujące, niż sądził, więc w najbliższym czasie nie będzie mógł brać udziału w projekcie. Niespodzianka, co? Jednym słowem, tę sprawę zostawiamy na później.

Kuba: No cóż, przynajmniej jedno zadanie mniej. Pomówmy o pracach nad edytorem tartanów. Nie chcę się chwalić, ale już to trochę przemyślałem. Pomajstrowałem nawet przy formularzu i dodałem trochę JavaScriptu.

Przemek: Świetnie. Formularz będzie śliczny i w ogóle, ale musimy się przecież jeszcze zająć częścią serwerową, która wykona całą ciężką pracę...

Kuba: Oj tam, część serwerowa... Uwierzcie, ulepszenie interfejsu formularza to wielka rzecz.

Przemek: Jasne, jasne. No dobra — zgłaszam się na ochotnika do pisania skryptów PHP. Kto się zajmie sprawą wydarzeń?

Kuba: A czy moglibyśmy się skupić tylko na tworzeniu nowych tartanów? Kiedy tylko z tym skończymy, wrócimy do tematu wyszukiwania wydarzeń. Mój mózg pozwala mi na skupienie się tylko na jednym zadaniu naraz.

Przemek: Może tak być, ale pod warunkiem że uporamy się z tym szybko. Mamy dosyć napięty harmonogram.

Do zrobienia — implementacja edytora tartanów

← To nasz plan działania związany z edytorem tartanów.

Zacniemy od tego!

- Ulepszyć istniejący formularz, by mógł wykorzystać możliwości oferowane przez nowsze przeglądarki mobilne.
- Napisać kod strony serwerowej, który jest odpowiedzialny za przetwarzanie danych z formularza oraz generowanie zasobów (obrazów, kodu HTML itp.) na potrzeby tartanów utworzonych przez użytkownika.
- Zapewnić możliwość pracy w trybie offline w tej części aplikacji.

Na miejsca, gotowi, ulepszać!

Zapnijcie pasy! Zamierzamy w rekordowo krótkim czasie wprowadzić sporo ulepszeń do formularza Tartanatora.

Zrobiliśmy, co było trzeba

Zaprojektowaliśmy formularz, który spełnia minimalne wymagania. Żaden użytkownik nie pozostanie na lodzie (no dobra, może jakiś by się znalazł). Zobacz, jak wygląda formularz w przeglądarce z wyłączoną obsługą JavaScriptu.

Teraz możemy ulepszać

Co prawda formularz działa na każdym sprzęcie, ale ma kilka drobnych wad. Na przykład sześć pól z rozwijaną listą do wyboru koloru sprawdzi się w przypadku lepszych urządzeń, ale już niekoniecznie na tych nie tak dobrych.

Skoro podstawowa wersja formularza działa, możemy się zająć dodawaniem ulepszeń, dzięki którym korzystanie z niego na smartfonach będzie prawdziwą przyjemnością.

Pierwszy krok do wyjątkowych mobilnych aplikacji internetowych: dopracuj interfejs użytkownika na potrzeby przeglądarek, które go dobrze wspierają.



Spokojnie

Nie wpadaj w panikę na myśl o własnym widzenie. JavaScript już się nie może doczekać, by Ci w tym pomóc.

Kilku genialnych programistów frontendowych opracowało już te ulepszenia. To ciekawe, że kiedy trzeba pracować nad mobilnymi aplikacjami internetowymi, wszyscy są tacy podekscytowani i produktywni.

Pokażemy Ci najistotniejsze elementy (oczywiście będziesz mógł poświęcić trochę czasu na dogłębnější analizę kodu), ale sam nie będziesz musiał napisać ani linijki!

Formularz działa prawidłowo nawet bez żadnego wsparcia ze strony JavaScriptu!

W przeglądarkach na nowszych smartfonach wygląda jednak zdecydowanie lepiej.

Możemy się pozbyć tych powtarzających się zestawów pól (które niepotrzebnie zajmują dużo przestrzeni) i w zamian utworzyć pojedynczy widżet.

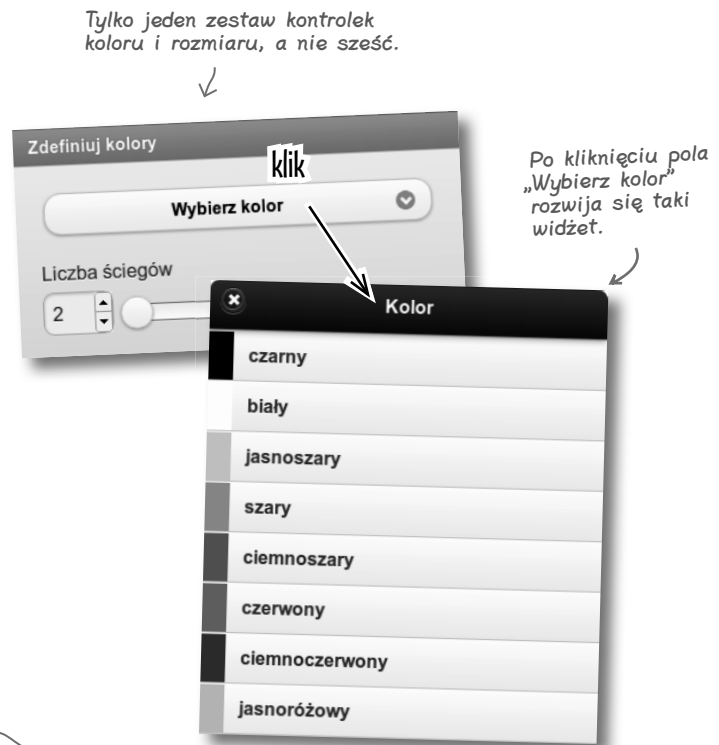
Ulepszamy formularz

Zamiast sześciu pól — zajmujących dużo miejsca i średnio funkcjonalnych — zastosujemy jeden widżet. Dzięki niemu użytkownicy będą mogli dodać tyle kolorów, ile chcą. Zastosujemy w tym celu skrypt JavaScript, który **usunie wszystkie pola koloru i rozmiaru z wyjątkiem pierwszego zestawu**.

Własny widżet dla pola wyboru koloru

Domyślny interfejs pola wyboru możemy zastąpić widżetem pochodzącym z frameworku jQuery Mobile. Dzięki temu w każdej pozycji listy rozwijanej będziemy mogli wyświetlić próbkę koloru.

Widżet, który zamierzamy zastosować, jest wyświetlany podobnie jak okno dialogowe, tyle że zawiera dostępne opcje koloru.



Dla maniaków

Dla zainteresowanych jQuery i JavaScriptem.

Jeśli nie jest Ci obca biblioteka jQuery, z pewnością wielokrotnie się spotkałeś z konstrukcją `$(document).ready()` (lub nawet z niej korzystałeś). Metoda `ready()` przypomina (w pewnym stopniu) zdarzenie przeglądarki `body.onload` — jest wywoływana po załadowaniu i utworzeniu struktury DOM. Programiści korzystający z jQuery zazwyczaj otaczają swój kod właśnie tą konstrukcją, by został wykonany dopiero wtedy, gdy struktura DOM jest gotowa.

Podczas korzystania z jQuery Mobile (jQM) zwykle nie będziesz używał konstrukcji `$(document).ready()`. jQM wprowadza bowiem kilka nowych zdarzeń związanych z ładowaniem strony, a najistotniejsze z nich to `pageinit` oraz `pagecreate`. Zdarzenie `pagecreate` jest zgłaszane po zakończeniu inicjalizacji struktury DOM danej strony przez jQuery i jQM, ale zanim zostaną wyświetlone widżety. Z kolei zdarzenie `pageinit` jest zgłaszane po wyrenderowaniu całej strony, łącznie z widżetami. Pamiętaj, że później wyświetlane strony są często wstawiane do istniejącej struktury DOM, co oznacza, że zdarzenia `pagecreate` i `pageinit` mogą być wielokrotnie zgłaszane podczas pojedynczego ładowania strony.

Framework jQuery Mobile wprowadza także wiele „mobilnych” zdarzeń związanych na przykład z dotykaniem, zmianą orientacji urządzenia czy z przejściami.

Nie wszystko wydaje Ci się jasne? Nie przejmuj się, nam też zajęło chwilę ogarnięcie tego. Więcej na ten temat znajdziesz w dokumentacji jQM (<http://jquerymobile.com/demos>).

Widżet zarządzający listą kolorów i rozmiarów

Usunęliśmy wszystkie zestawy pól kolor – rozmiar z wyjątkiem jednego. Musimy sprawić, by za pomocą pozostawionej pary pól można było zdefiniować dowolną liczbę kolorów.

Aby to zrobić, musimy dodać nowy przycisk — *Dodaj wybrany kolor*. Po jego kliknięciu wybrany kolor i liczba szwów muszą zostać wstawione do listy () jako element . W tym elemencie umieścimy też ukryte pole formularza, w którym zapiszemy kolor i rozmiar.

Tym zajmie się JavaScript.

Kliknięcie elementu znajdującego się na liście ma spowodować jego usunięcie. Z kolei przycisk *Utwórz!* ma służyć do wygenerowania wzoru tartanu na podstawie zdefiniowanych kolorów.

Zdefiniuj kolory

Pozostawiona para pól kolor – rozmiar.

Po kliknięciu przycisku „Dodaj wybrany kolor” do listy jest dodawany ustawiony kolor.

Lista par kolor – rozmiar. `ul#colorlist`

Tło pola select odpowiada aktualnie wybranemu kolorowi.

Podczas zmiany położenia suwaka funkcja sprawdza, czy liczba szwów jest parzysta.

To jest obejście problemu ze wsparciem dla atrybutu step w polach typu range.

Kliknięcie elementu listy powoduje jego usunięcie.

Przycisk zatwierdzający formularz.

Ulepszona wersja formularza z pliku build.php (fragment)

Utwórz!

Weź to w obroty

Mamy dobrą wiadomość. Nie musisz nic sam robić, by udoskonalić formularz. Zrobiliśmy to za Ciebie.

Punktem wyjścia jest katalog *rozdzial7*, w którym umieściliśmy skrypt JavaScript z zaimplementowanymi usprawnieniami. Sam sprawdź! Zobacz też, jak to działa w mobilnej przeglądarce.

Pamiętaj, że formularz znajduje się w pliku `build.php`.

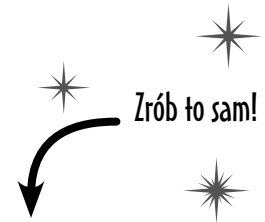
Nie zajęliśmy się jeszcze stroną serwerową. Kliknięcie przycisku „Utwórz!” na razie nie działa.

Zaglądamy pod przykrywkę

Obiecaliśmy, że nie będziesz musiał napisać ani kawałka kodu JavaScript, ale może uda nam się skłonić Cię do zainteresowania się, co właściwie ten kod robi.

W skrócie: jest kilka istotnych działań, które mają miejsce po załadowaniu strony z formularzem i utworzeniu struktury DOM (zdarzenie `pagecreate`) oraz po zakończeniu operacji przeprowadzanych przez framework jQuery Mobile (zdarzenie `pageinit`).

W tych funkcjach inicjalizujących przypisaliśmy funkcje do różnych zdarzeń (zmiany pola formularza, kliknięcie, zatwierdzenia formularza itp.). Dzięki temu udało się wzbogacić interakcję i poprawić funkcjonalność widżetu.



Zaprzyjajnij się z kodem. Przeznacz trochę czasu na zapoznanie się z aktualizacjami w plikach aplikacji Tartanator, a zwłaszcza ze skryptem `tartanator.js`.

KTO ZA CO ODPOWIADA?

Dopasuj każdą z funkcji JavaScript umieszczonych w pliku `tartanator.js` do realizowanych zadań oraz zdarzeń, które je wywołują. W główkowaniu mogą Ci pomóc komentarze umieszczone w skrypcie.

Zadanie dodatkowe!

Nazwa funkcji	Realizowane zadanie	Zdarzenie, które ją wywołuje
<code>onStitchSizeChange()</code>	Odświeża elementy <code></code> i <code></code> bieżącej listy kolorów.	<code>pageinit</code> oraz <code>change</code> (pola wyboru koloru)
<code>styleColorListItem()</code>	Tworzy przycisk dodający parę kolor – rozmiar i wstawia go do struktury DOM.	<code>click</code> (przycisk <i>Dodaj wybrany kolor</i>)
<code>buildAddButton()</code>	Ustawia kolor tła pola wyboru koloru na aktualnie wybrany.	<code>pageinit</code>
<code>onColorListChange()</code>	Tworzy ukryte pole formularza, umieszcza je w nowym elemencie <code></code> , a następnie dołącza ten element do istniejącej listy kolorów.	<code>click</code> (element bieżącej listy kolorów) oraz po dodaniu nowego koloru do listy
<code>addColor()</code>	Za pomocą CSS umieszcza kolorową ramkę (próbkę) z lewej strony każdego elementu opcji rozwijanej listy kolorów.	<code>change</code> (po zaktualizowaniu wartości pola rozmiaru)
<code>setColorSelectStyle()</code>	Sprawdza, czy ustawiony rozmiar jest liczbą parzystą.	<code>pagecreate</code>

No tak, to było ulepszanie frontendu

- Ulepszyć istniejący formularz, by mógł wykorzystać możliwości ← Zrobione!
oferowane przez nowsze przeglądarki mobilne.
- Napisać kod strony serwerowej, który jest odpowiedzialny za
przetwarzanie danych z formularza oraz generowanie zasobów (obrazów,
kodu HTML itp.) na potrzeby tartanów utworzonych przez użytkownika. ←
Następne na warsztata!
- Zapewnić możliwość pracy w trybie offline w tej części aplikacji.

KTO ZA CO ODPOWIADA?

ROZWIĄZANIE

Dopasuj każdą z funkcji JavaScript umieszczonych w pliku *tartanator.js* do realizowanych zadań oraz zdarzeń, które je wywołują. W główkowaniu mogą Ci pomóc komentarze umieszczone w skrypcie.

Nazwa funkcji	Realizowane zadanie	Zdarzenie, które ją wywołuje
<i>onStitchSizeChange()</i>	Odświeża elementy <code></code> i <code></code> bieżącej listy kolorów.	pageinit oraz change (pola wyboru koloru)
<i>styleColorListItem()</i>	Tworzy przycisk dodający parę kolor – rozmiar i wstawia go do struktury DOM.	click (przycisk <i>Dodaj wybrany kolor</i>)
<i>buildAddButton()</i>	Ustawia kolor tła pola wyboru koloru na aktualnie wybrany.	pageinit
<i>onColorListChange()</i>	Tworzy ukryte pole formularza, umieszcza je w nowym elemencie <code></code> , a następnie dołącza ten element do istniejącej listy kolorów.	click (element bieżącej listy kolorów) oraz po dodaniu nowego koloru do listy
<i>addColor()</i>	Za pomocą CSS umieszcza kolorową ramkę (próbkę) z lewej strony każdego elementu opcji rozwijanej listy kolorów.	change (po zaktualizowaniu wartości pola rozmiaru)
<i>setColorSelectStyle()</i>	Sprawdza, czy ustawiony rozmiar jest liczbą parzystą.	pagecreate



Wszystko o ulepszeniach interfejsu

Wywiad tygodnia:

Ulepszenia: czy to jedynie kosmetyczne poprawki?

Redaktor: Cześć, muszę o to zapytać. Zrobiliśmy właśnie szybką rundkę po ulepszeniach interfejsu za pomocą JavaScriptu w projekcie, który wykorzystuje framework jQuery Mobile. Czy naprawdę było warto?

Ulepszenia interfejsu: To, na co pozwala jQuery Mobile i jak ułatwia tworzenie interfejsów wprost stworzonych dla mobilnych przeglądarek, jest naprawdę niesamowite, ale rozumiem, o co ci chodzi. Chciałbym zarysować szerszy sens tego typu działań.

Redaktor: Słuchamy...

Ulepszenia: Bez względu na to, czy stosujesz jakiś framework, czy piszesz od podstaw, najlepszym rozwiązaniem jest projektowanie z myślą o minimalnych wymaganiach, a następnie wprowadzanie ulepszeń.

Redaktor: Za pomocą JavaScriptu?

Ulepszenia: Otwórz swój umysł... to wykracza daleko poza JavaScript. Mówię tu o sytuacji, w której zaczynasz od zdefiniowania podstawowej funkcjonalności i zawartości, a następnie stopniowo je ulepszasz. W ten sposób wychodzisz naprzeciw możliwościom oferowanym przez różne urządzenia mobilne.

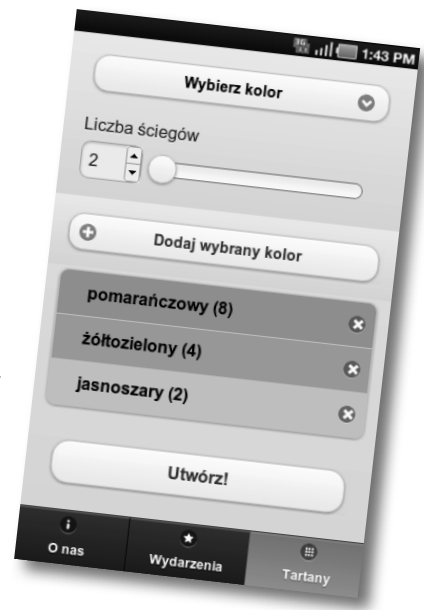
Redaktor: Tak, znam to. Dodaje się zwykle jakieś wodotryski...

Ulepszenia: Chciałbym wierzyć, że mój wkład w sprawę to coś więcej niż tylko zaokrąglone wierzchołki i gradienty. Przed chwilą zaczęliśmy pracę od dość ograniczonego i nieszczęśliwego przypadku, a skończyliśmy na nie dość, że ładnym, to jeszcze funkcjonalnym formularzu.

Redaktor: Zmieniłeś sposób renderowania natywnych elementów formularza. Pole wyboru koloru używa teraz widżetu jQuery Mobile. Czy nie przesadzasz?

Ulepszenia: Zgodzę się, że mieszanie w natywnych kontrolkach formularza jest trochę kontrowersyjne. Nie kłóć się z tym jako ogólną zasadą. Jednak w przypadku pola wyboru koloru nie byłoby możliwe osiągnięcie takiego efektu, czyli kolorowych próbek przy nazwach kolorów, za pomocą tradycyjnych pól wyboru.

Zwróć jednak uwagę, że kod HTML odpowiadający za to pole (`<select>`) nadal jest semantycznie poprawny. Ja je tylko trochę poprawiłem, by stało się bardziej użyteczne. Podsumowując, masz rację, że do zmiany natywnych kontrolki formularza należy podchodzić z dużą ostrożnością.

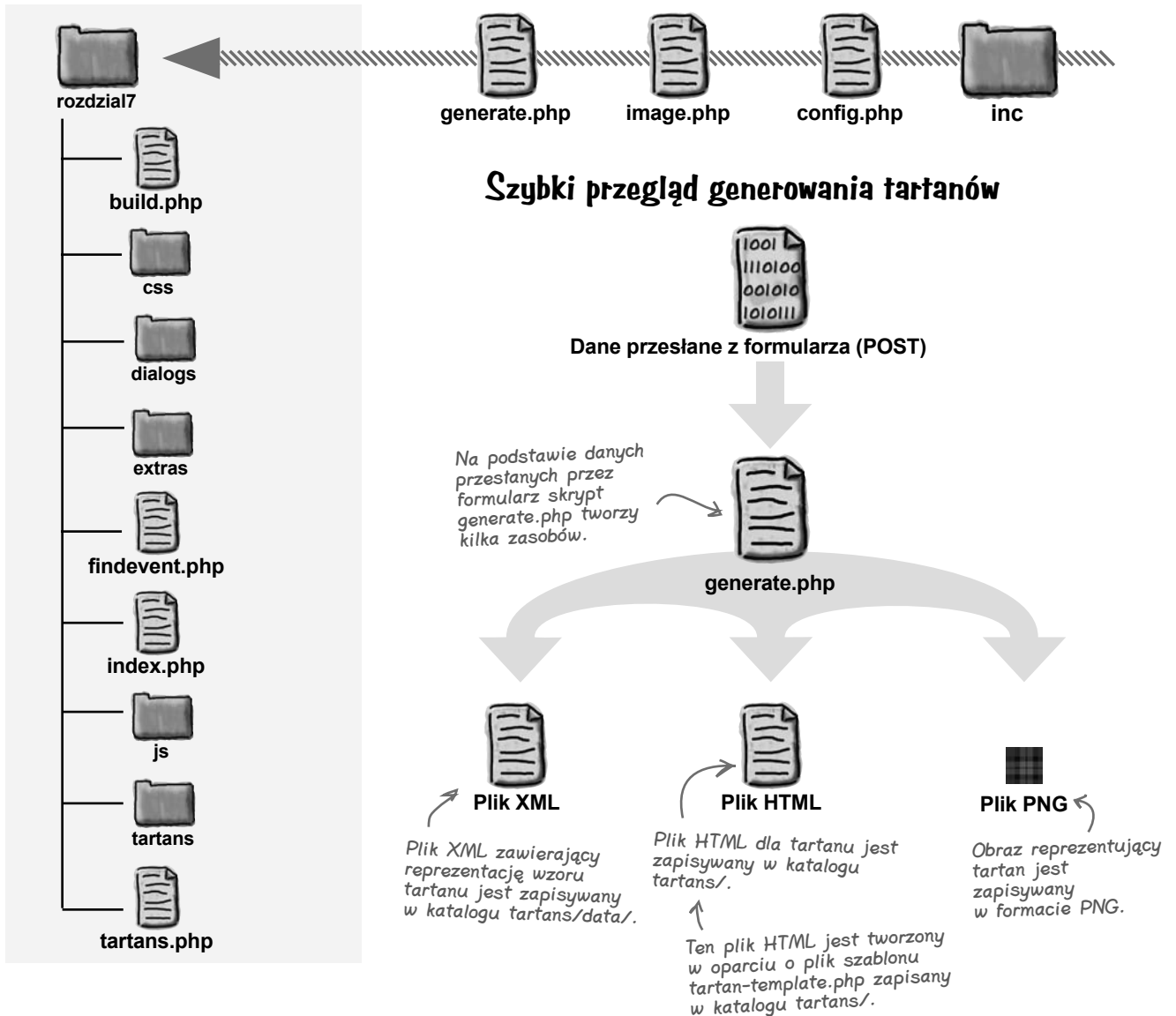


Formularz do tworzenia tartanów w przeglądarce Androida

...a teraz pora na backend

Punktem wyjścia w rozdziale 7. jest struktura plików przedstawiona poniżej. Dołączenie elementów działających po stronie serwera wymaga przeprowadzenia kilku prostych operacji.

Skopiuj całą zawartość katalogu *extras/scripts* do katalogu *rozdziel7*. Po zakończeniu kopiowania w katalogu *rozdziel7* powinny się pojawić trzy nowe pliki: *config.php*, *generate.php* i *image.php* oraz katalog *inc* (zawierający dwa pliki).



Dwie twarze skryptu generate.php

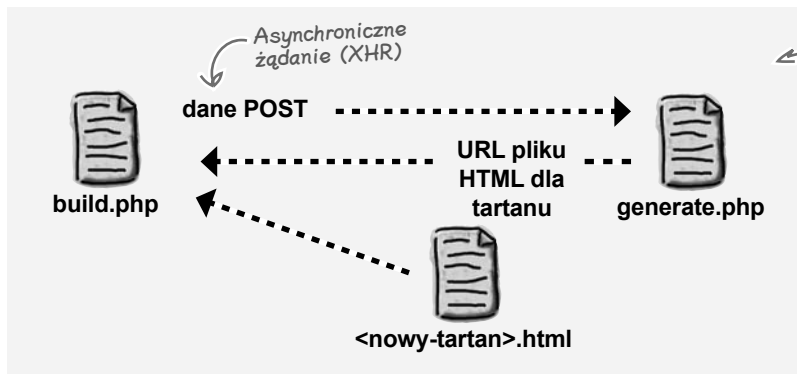
Sposób, w jaki skrypt *generate.php* zachowuje się po utworzeniu zasobów związanych z tartanem, zależy od sposobu jego wywołania.

Żądanie za pomocą AJAX-a

W przeglądarkach, które prawidłowo obsługują AJAX-a, skrypt JavaScript znajdujący się w pliku *build.php* przesyła dane z formularza do skryptu *generate.php* za pomocą obiektu XMLHttpRequest (w skrócie XHR). Jeśli operacja się powiedzie, skrypt *generate.php* w odpowiedzi przesyła adres URL nowo utworzonego pliku HTML dla zaprojektowanego tartanu. Zawartość tej strony jest następnie umieszczana w strukturze DOM bieżącej strony.

Postaraliśmy się zapewnić wsparcie zarówno dla przeglądarek obsługujących AJAX-a, jak i tych, które go nie obsługują.

Asynchroniczne żądanie zrealizowane za pomocą XHR



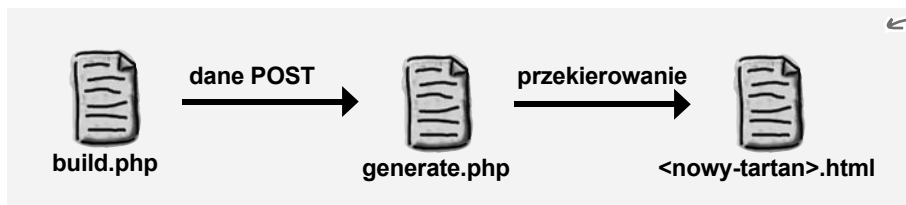
W tej metodzie nigdy nie dochodzi do pełnego przeładowania strony. Zawartość strony nowego tartanu jest wstawiana do struktury DOM strony edytora tartanów.

Formularz przesłany bezpośrednio

W przypadku przeglądarek, które nie obsługują JavaScriptu ani obiektu XMLHttpRequest, formularz jest przesyłany do skryptu *generate.php* tradycyjną metodą. Po utworzeniu w skrypcie zasobów związanych z tartanem przeglądarka jest przekierowywana do nowo utworzonej strony tartanu.

Bez względu na metodę podstawowe zadanie skryptu *generate.php* jest jedno: utworzyć zasoby dla nowego tartanu.

Tradycyjna metoda — przekierowanie do nowo utworzonej strony tartanu



W tej metodzie klient jest przekierowywany do nowo utworzonej strony tartanu (dochodzi do pełnego przeładowania strony).

Jeszcze tylko jedno...

Teraz, kiedy użytkownicy mogą już tworzyć własne wzory, lista tartanów powinna się zmieniać. Umieść poniższy kod w pliku *tartans.php* — dołącza on skrypt generujący elementy `` dla każdego istniejącego tartanu.



Ćwiczenie

Czas utworzyć trochę tartanów! Za pomocą formularza ze strony *build.php* zaprojektuj kilka lub kilkanaście wzorów tartanów. Twoje projekty powinny się pojawić na liście tartanów na głównej stronie (*tartans.php*).

Nie istnieją grupie pytania

P: Pomocy! Coś tu nie działa prawidłowo!

U: Jeśli masz problem z uruchomieniem formularza lub nie udaje Ci się utworzyć nowego tartanu, powinieneś sprawdzić kilka spraw. Przede wszystkim sprawdź, czy istnieje katalog *tartans* i czy masz możliwość zapisywania w jego podkatalogach. Sprawdź też, czy w katalogu *tartans* znajduje się plik *tartan-template.php*. Na koniec powinieneś trzy razy sprawdzić znacznik `<form>` w pliku *build.php* pod kątem atrybutów `act` i `on` i `method`.

P: Jak działa strona z listą tartanów?

U: Plik *list.php* dołączony do strony szuka w katalogu *tartans/* plików HTML i tworzy z nich listę. Dla każdego z nich pobiera powiązany z nim plik XML (z katalogu *tartans/data/*) i odczytuje z niego dodatkowe informacje, jak choćby nazwę. Następnie tworzy kod elementu `` dla każdego tartanu, zawierający odsyłacz do strony HTML tartanu.

P: O co dokładnie chodzi z tym plikiem XML?

U: Dane opisujące wzory tartanów postanowiliśmy umieścić w pliku XML z kilku powodów. Po pierwsze, to fajny, prosty i przenośny format danych. Po drugie, dzięki temu unikamy konieczności używania bazy danych (co oznacza mniej pracy dla Ciebie!).

P: A tak przy okazji — czym są te wspaniałe mobilne aplikacje internetowe, o których mówiliście wcześniej? Czy to jakiś standard, inicjatywa, czy coś jeszcze innego?

U: Nic z tych rzeczy. Po prostu tak nazwaliśmy mobilne aplikacje internetowe, które wykorzystują dobrodziejstwa oferowane przez urządzenia mobilne i ich przeglądarki.

Dobra robota — dwa zadania z trzech załatwione

Coraz bardziej zbliżamy się do wyjątkowej mobilnej aplikacji internetowej. Udoskonaliliśmy już interfejs, z czego skorzystają użytkownicy lepszych przeglądarek. Zadbaliśmy też w końcu o to, by aplikacja coś robiła!

- Ulepszyć istniejący formularz, by mógł wykorzystać możliwości oferowane przez nowsze przeglądarki mobilne.
- Napisać kod strony serwerowej, który jest odpowiedzialny za przetwarzanie danych z formularza oraz generowanie zasobów (obrazów, kodu HTML itp.) na potrzeby tartanów utworzonych przez użytkownika.
- Zapewnić możliwość pracy w trybie offline w tej części aplikacji.

← *Musimy sobie poradzić jeszcze z tym.*

Ale to jeszcze nie wszystko

Teraz musimy się zająć trzecim elementem układanki — Tartanator musi działać offline.



To frustrujące. Chciałem się pochwalić koledze tartanem, który zaprojektowałem, ale był tak słaby zasięg, że nie udało mi się połączyć z siecią. Koniec końców, nie udało mi się nic pokazać.

W przypadku urządzeń mobilnych nie możemy zagwarantować dobrego ani nawet jakiegokolwiek zasięgu sieci.

Musimy zrobić coś, co zagwarantuje działanie aplikacji Tartanator nawet bez aktywnego połączenia z internetem.

Tryb offline to ważna sprawa

Chcąc zapewnić wysoki poziom funkcjonalności i użyteczności aplikacji, musimy się zająć zachowaniem witryny lub aplikacji, w przypadku gdy nie jest dostępne połączenie z internetem.

Musimy się teraz poważnie zastanowić nad tym, co zrobić, by Tartanator działał lepiej w przypadku braku połączenia. Z całą pewnością kilka elementów musi być dostępnych offline.



Jak możemy
decydować, co jest
dostępne offline? Czy to
jest w ogóle możliwe?

Możemy użyć tzw. pliku manifestu, w którym definiuje się składniki aplikacji dostępne offline.

Zamanifestuj to!

Podręczna pamięć aplikacji (ang. *Application Cache*) jest częścią specyfikacji HTML5. Pozwala na ustalenie, które zasoby są przechowywane na potrzeby trybu offline. Służy do tego *plik manifestu* (ang. *cache manifest*).

Plik manifestu jest umieszczany na serwerze, a jego zadaniem jest przekazywanie szczegółowych informacji na temat przechowywania określonych zasobów w pamięci podręcznej przeglądarki użytkownika. W ten sposób będziemy mogli określić, które pliki aplikacji Tartanator mają być dostępne w trybie offline.

Przeglądarki, które obsługują pamięć podręczną aplikacji (często określa się ją w skrócie jako *appCache*), udostępniają obiekt `window.applicationCache` i powiązane z nim zdarzenia, z których możemy korzystać z poziomu JavaScriptu.

Większość nowoczesnych przeglądarek obsługuje ten mechanizm. Niechlubnymi wyjątkami są Internet Explorer 9 i Opera Mini. Zastosowanie pliku manifestu w żaden sposób nie wpłynie jednak na ich pracę — po prostu go zignorują.



WYSIL SZARE KOMÓRKI

Czy umiałbyś wyjaśnić, dlaczego przeglądarka Opera Mini może mieć problem ze wsparciem dla tego mechanizmu?

Prosty przepis na plik manifestu

Aby utworzyć plik manifestu i zastosować go w witrynie lub aplikacji, trzeba wykonać trzy kroki:

- 1 Napisać plik manifestu.
- 2 Do znacznika `<html>` wszystkich pożądaných stron dodać atrybut `manifest` i przypisać mu adres URL pliku manifestu (może być względny).
- 3 Sprawić, by plik manifestu był udostępniany z prawidłowym nagłówkiem typu. ←

Plik musi mieć nagłówek `content-type` równy `text/cache-manifest`. W przeciwnym razie nie będzie działać.

Podejrzenie prosta składnia pliku manifestu

Zawartość pliku manifestu wygląda na mało skomplikowaną. W sekcji `CACHE:` (nie jest to wymagane, ale powinno się jednoznacznie określać tę sekcję) wypisuje się zasoby, które mają być dostępne offline.

Ten wiersz jest wymagany (dokładnie w takiej postaci).

Adresy URL mogą być względne (jak w tym przypadku) lub bezwzględne (np. `http://...`).

```

CACHE MANIFEST
# tak się umieszczają komentarze

CACHE:

index.html
foo/bar.html
baz.html
css/styles.css
icons/plus.png
    
```

Plik nie musi się nazywać „manifest”, ale powinien mieć rozszerzenie „.appcache”.

manifest.appcache

Następnie trzeba zaktualizować stronę (lub strony), dodając atrybut `manifest` odwołujący się do pliku manifestu. Trzeba też zadbać o to, by plik był udostępniany z prawidłowym nagłówkiem `content-type` (lub inaczej `MIME-type`). W przypadku serwera Apache wystarczy dodać odpowiedni wpis do pliku konfiguracyjnego lub, co jest lepszym rozwiązaniem, do pliku `.htaccess` umieszczonego w głównym katalogu witryny.

```

<!DOCTYPE html>
<html manifest="manifest.appcache">
<head>
    
```

```

AddType text/cache-manifest .appcache
    
```

Ten wpis informuje serwer WWW, by udostępniał pliki o rozszerzeniu `.appcache` z nagłówkiem `content-type` równym `text/cache-manifest`.

Plik musi być tego typu, bo w przeciwnym przypadku przeglądarka go nie rozpozna.

.htaccess

Jak zwykle diabeł tkwi w szczegółach

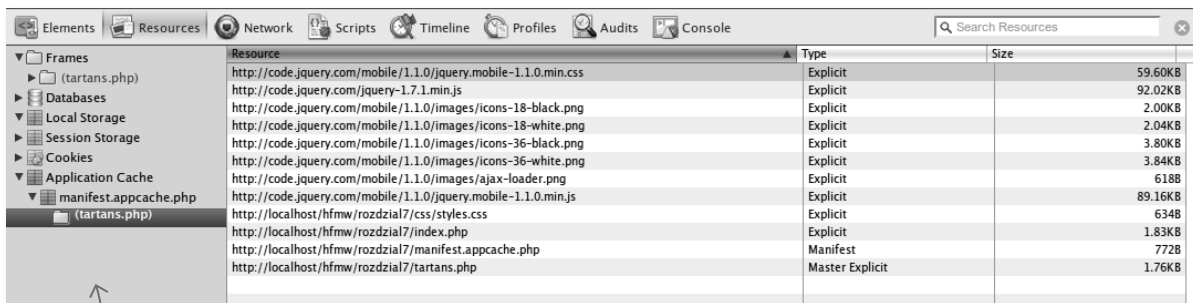
W teorii zarządzanie zasobami w trybie offline za pomocą pliku manifestu jest proste. Wystarczy wypisać pliki, które mają być dostępne offline, dodać atrybut `manifest` do znacznika `<html>` na stronach i już!

Sprawy nie wyglądają jednak tak różowo. Zmuszenie całego mechanizmu, by działał tak, jak chcemy, może być trudne, a czasem wręcz frustrujące. Po drodze czai się wiele niebezpieczeństw, które mogą Cię sprowadzić na manowce. Koniecznie musisz się zaopatrzyć w dobre narzędzie, które pomoże Ci zapanować nad tym, co się dzieje.

Na ratunek przybywają narzędzia

Silnik WebKit dostarcza narzędzie Web Inspector, które jest dostępne zarówno w przeglądarce Chrome, jak i Safari (która też jest oparta na tym silniku). Szczegóły dotyczące mechanizmu Application Cache znajdziesz w zakładce *Resources*.

Szczegółowe informacje o podręcznej pamięci aplikacji znajdują się na zakładce *Resources* w przeglądarce Chrome.



W przeglądarce Safari wygląda to niemal identycznie.



Podczas pracy z plikami manifestu łatwo jest się zaciąć.

Obejrzyj to!

Zachowanie mechanizmu podręcznej pamięci aplikacji może być momentami niezrozumiałe i trudne w debugowaniu. Niepoprawne pliki manifestu mogą sporo namieszać, a znalezienie przyczyny nieprawidłowego zachowania bez odpowiednich narzędzi może być trudne. Proponujemy, byś podczas tworzenia i testowania plików manifestu dla aplikacji (również dla Tartanatora) korzystał z przeglądarek Chrome lub Safari. Musisz też wiedzieć, że mechanizm podręcznej pamięci aplikacji nie jest obsługiwany w Internet Explorerze do wersji 10. (która w chwili pisania książki jest nadal w fazie testów).

Dobrym pomysłem jest też tworzenie i testowanie aplikacji w przeglądarce desktopowej, a nie mobilnej.

Usuwanie źle zdefiniowanych ustawień z plików manifestu w przeglądarce mobilnej może być bardzo przykrym doświadczeniem. →

Udostępniaj pliki manifestu z prawidłowym nagłówkiem content-type



Nigdy się nie bawiłam plikami .htaccess i nie wiem nawet, czy mój hosting na to pozwala.

Jest bardzo prawdopodobne, że możesz używać plików .htaccess na swoim serwerze — zarówno hostowanym, jak i lokalnym. My jednak obejdziemy się bez nich.

Wiemy, że możesz używać PHP. Skoro tak, wygenerujemy plik manifestu za pomocą PHP i przy okazji ustawimy nagłówki content-type na text/cache-manifest.



Zaimplementuj podstawowy plik manifestu dla Tartanatora.

Ćwiczenie

- 1 Utwórz nowy plik i zapisz go w katalogu *rozdziel7* pod nazwą *manifest.appcache.php*.
- 2 Na samym początku pliku umieść poniższy wiersz kodu:

```
<?php header('Content-type: text/cache-manifest'); ?>
```

Rozszerzenie .php jest potrzebne, by mógł być wykonany kod PHP umieszczony w tym pliku.

W ten sposób ustawiamy nagłówek content-type na text/cache-manifest.

- 3 Dodaj sekcję CACHE: i wypisz główne strony witryny, a także pliki JavaScript i CSS.
- 4 Do znacznika <html> w plikach *index.php*, *build.php*, *tartans.php* i *findevent.php* dodaj atrybut manifest.

Zastosuj składnię przedstawioną na stronie 285.



Rozwiązanie ćwiczenia

Gotowy plik `manifest.appcache.php` powinien wyglądać tak jak poniżej. Zwróć uwagę, że dołączyliśmy też pliki CSS i JavaScript z serwisu CDN (ang. *Content Delivery Network*) hostowanego przez jQuery.

```
<?php header('Content-type: text/cache-manifest'); ?>
CACHE MANIFEST
CACHE:
index.php
build.php
tartans.php
findevent.php
css/styles.css
js/tartanator.js
http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css
http://code.jquery.com/jquery-1.7.1.min.js
http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js
```

Poza względnymi adresami URL zasobów witryny...

...możesz też używać bezwzględnych adresów zasobów znajdujących się w innych domenach.



`manifest.appcache.php`

Znaczniki `<html>` na stronach składających się na aplikację Tartanator powinny wyglądać w ten sposób.

```
<html manifest="manifest.appcache.php">
```

W związku z tym, że do generowania pliku manifestu użyliśmy PHP, nie możemy zastosować rozszerzenia `.appcache`.

Chyba coś robię nie tak.
Na stronach przestały być wyświetlane obrazki i ikony, nawet kiedy jestem w trybie online!





Wszystko o pliku manifestu

Wywiad tygodnia:

Jak zmusić appCache, by zrobił, co mu każemy

Sfrustrowany programista: Wrr... Przejrzałem poszczególne strony Tartanatora i widzę, że brakuje mnóstwa obrazów i ikon, nawet kiedy pracuję online.

appCache: No cóż, to dlatego, że do listy w sekcji CACHE nie dołączyłeś tych zasobów. Powinieneś je koniecznie dodać do pliku manifestu.

SP: Czyli muszę dołączyć wszystkie zasoby witryny, bo w przeciwnym razie się nie pojawiają?

appCache: Nie, musisz dodać te zasoby, które są potrzebne plikom HTML zapisanym w pamięci podręcznej.

SP: Nic z tego nie rozumiem. Co masz na myśli, mówiąc o plikach HTML zapisanych w pamięci podręcznej?

appCache: Znasz już jeden sposób na uwzględnianie plików HTML w pamięci podręcznej aplikacji — wypisanie ich w sekcji CACHE pliku manifestu. Jednak wszystkie pliki HTML, które w znaczniku <html> mają atrybut manifest, są uwzględniane przez mechanizm pamięci podręcznej, nawet jeśli nie są wypisane w pliku manifestu. Tylko dla tych stron, które są wypisane w pliku manifestu lub mają atrybut manifest, trzeba dołączyć wszystkie zasoby.

SP: Zatem jeśli zapomnę dodać wszystkie zasoby w sekcji CACHE dla którejkolwiek strony uwzględnionej w pamięci podręcznej, nie będą one wyświetlane nawet w trybie online?

appCache: No cóż, nie do końca tak jest. Wiesz już co nieco o sekcji CACHE, ale nie słyszałeś jeszcze o sekcji NETWORK.

SP: A do czego służy?

appCache: Sekcja NETWORK pozwala zdefiniować zasoby, które nie mają być przechowywane w podręcznej pamięci aplikacji. Przeglądarka powinna pobierać zawsze ich świeżą kopię (oczywiście przy dostępnym połączeniu). W tej sekcji można użyć poręcznego symbolu wieloznacznego *, który reprezentuje wszystko z wyjątkiem zasobów wypisanych w sekcji CACHE.

SP: Eureka! Tego właśnie potrzebowałem.

appCache: Musisz jednak uważać. To, co umieścisz w sekcji NETWORK (bezpośrednio lub za pomocą symbolu *), zawsze będzie pobierane z serwera. Zachowaj ostrożność podczas definiowania ścieżek.

Musisz zdecydować, które zasoby koniecznie powinny być dostępne offline, i umieścić je w sekcji CACHE. Powinny się tam znaleźć na przykład pliki ikon i obrazów. Jednak w przypadku dynamicznych zasobów, takich jak ekrany logowania czy wywołania API, powinieneś skorzystać z sekcji NETWORK i symbolu *.



WYTEŻ UMYSŁ

Czy masz pomysł, które elementy aplikacji Tartanator nie powinny być dostępne offline? Dlaczego właśnie one?

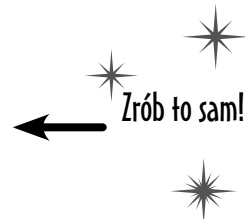


Chcemy, by strony i obrazy tartanów były dostępne offline, więc musimy je dodać do sekcji CACHE. Ale jak to zrobić, skoro nie znamy ich adresów?

Wygląda to na klasyczny problem „co było pierwsze: jajko czy kura”...

Na szczęście plik manifestu generujemy za pomocą PHP, więc wystarczy dodać trochę kodu, który dynamicznie wygeneruje listę wszystkich utworzonych plików HTML i obrazów tartanów.

W pliku `current_file_list.txt` znajdującym się w katalogu `rozdzial7/extras` znajdziesz kod PHP dynamicznie generujący listę obrazów i plików HTML.



Obejrzyj to!

Zmiany w plikach zasobów wymienionych w sekcji CACHE pliku manifestu nie zostaną pobrane przez przeglądarkę do czasu zmiany samego pliku manifestu.

W sekcji CACHE pliku manifestu uwzględniamy plik `css/styles.css`. Możemy wprowadzać w nim zmiany, ale przeglądarka ich nie zobaczy. Po zapisaniu jakiegoś zasobu przez mechanizm podręcznej pamięci aplikacji jedyną możliwością jego odświeżenia jest zmiana samego pliku manifestu.

Popularnym sposobem rozwiązania tego problemu jest umieszczenie w komentarzu numeru wersji. Wystarczy zmienić tę wartość, by przeglądarka uznała, że plik manifestu został zmodyfikowany, co powoduje jego pobranie oraz sprawdzenie, czy pliki zasobów uległy zmianie.

Wspomniany kod PHP automatycznie generuje listę plików HTML i obrazów utworzonych tartanów, czego efektem ubocznym jest ładowanie przez przeglądarkę pliku manifestu za każdym razem, gdy zostanie dodany nowy tartan. Jednak jeśli wprowadziliśmy zmiany w innych plikach, musimy samodzielnie powiększyć numer wersji zapisany w pliku manifestu.

Pamiętaj, że komentarze oznacza się symbolem #.



Spokojnie

Bardzo łatwo jest utworzyć plik manifestu, który — jakby to delikatnie ująć — nie do końca odpowiada naszym zamierzeniom.

Okiełznanie krnąbrnych plików manifestu jest najprostsze w przeglądarce Chrome. Wystarczy w pasku adresu wpisać URL `chrome://appcache-internals`, a zostanie wyświetlona strona z informacjami o bieżących danych mechanizmu appCache. Przy zestawieniu dla każdej witryny znajduje się odsyłacz *Remove* służący do usuwania.

W przeglądarce Safari z menu ustawień trzeba wybrać pozycję *Preferencje* i przejść na zakładkę *Prywatność*. Można tu albo usunąć wszystko za pomocą przycisku *Usuń wszystkie dane witryn*, albo wcisnąć przycisk *Szczegóły* i usunąć dane wybranej witryny.

W przeglądarce Firefox z menu *Edycja* trzeba wybrać pozycję *Preferencje* i w oknie przejść na zakładkę *Zaawansowane/Sieć*. W sekcji *Pamięć trybu offline* znajduje się lista witryn, dla których są przechowywane dane. Z tego poziomu można wybrać odpowiednią witrynę i usunąć powiązane z nią dane.



Ćwiczenie

Najwyższy czas, by nasz plik manifestu zachowywał się trochę lepiej. Musimy uwzględnić wszystko to, o czym do tej pory mówiliśmy, i zastosować w pliku *manifest.appcache.php*.

- 1 Wyśledź brakujące ikony i obrazy.** Jest kilka ikon i obrazów używanych przez jQuery Mobile, które musimy dodać do sekcji CACHE.

← Czapki z głów, jeśli udało Ci się już to zrobić wcześniej.

- 2 Dodaj sekcję NETWORK z symbolem wieloznacznym *.** Umieść to przed sekcją CACHE:

```
NETWORK:
*
```

- 3 Dołącz fragment kodu PHP.** Na końcu pliku manifestu umieść kod z pliku *extras/current_file_list.txt*.

- 4 Z pliku manifestu usuń pliki build.php oraz findevent.php.** Po przemyśleniu sprawy uznaliśmy, że te dwie strony powinny być dostępne tylko w trybie online. Co prawda na stronie Wydarzenia jeszcze nic ciekawego nie ma, ale już niedługo.

← Nie zapomnij usunąć atrybutu `manifest` ze znacznika `<html>` w tych dwóch plikach.



Rozwiązanie ćwiczenia

Plik `manifest.appcache.php` powinien wyglądać jak poniżej.

```
<?php header('Content-type: text/cache-manifest'); ?>
CACHE MANIFEST
```

```
NETWORK:
*
```

```
CACHE:
```

```
http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.css
http://code.jquery.com/jquery-1.7.1.min.js
http://code.jquery.com/mobile/1.1.0/jquery.mobile-1.1.0.min.js
http://code.jquery.com/mobile/1.1.0/images/ajax-loader.png
http://code.jquery.com/mobile/1.1.0/images/icons-18-white.png
http://code.jquery.com/mobile/1.1.0/images/icons-18-black.png
http://code.jquery.com/mobile/1.1.0/images/icons-36-white.png
http://code.jquery.com/mobile/1.1.0/images/icons-36-black.png
index.php
tartans.php
css/styles.css
```

Uważaj, żeby między `header()` i `CACHE MANIFEST` nie było żadnego pustego wiersza.

```
<?php // W tym miejscu ma się znaleźć kod PHP z pliku current_file_list.txt.
// Nie zamieszczamy go tu, by zaoszczędzić trochę miejsca i ocalić kilka drzew.??>
```

Teraz zapisz plik i sprawdź, jak działa aplikacja!



Obejrzyj to!

Jeden błąd w pliku manifestu może spowodować problemy z całą aplikacją.

Pewnie się tego spodziewałeś — jest jeszcze więcej pułapek. Wystarczy, że żądanie jednego zasobu z pliku manifestu zakończy się kodem 404 (brak pliku), a cały plik manifestu jest ignorowany. Taki sam efekt może spowodować zwykła niewinna literówka.

Powinieneś się odpowiednio zabezpieczyć. Możesz skorzystać z walidatora <http://manifest-validator.com/> oraz na bieżąco śledzić informacje w przeglądarkowych narzędziach dla programistów, by kontrolować, czy żaden zasób nie został pominięty.



Nic z tego nie rozumiem.
Już myślałem, że to działa, ale kiedy
dodałem nowy tartan, nie pojawił
się na stronie z listą tartanów.
Musiąłem przeładować stronę.
Coś jest chyba nie tak?

**Mimo że plik manifestu został
zaktualizowany, aby zobaczyć nowo
dodany wzór, musisz przeładować
stronę z listą tartanów (*tartans.php*).
Dlaczego?**



Łukasz: Poczytałem o tym trochę. Przyznaję, bardzo mnie wciągnęło. Chcesz się dowiedzieć, jak to działa? Powiedzmy, że przeglądarka ma już plik manifestu dla twojej witryny. Wchodzisz na stronę z listą tartanów po utworzeniu kilku wzorów. Przeglądarka zauważa, że plik manifestu się zmienił — różni się w porównaniu z tym, który ma listę plików powiązanych z tartanami. W związku z tym natychmiast pobiera nowy plik manifestu i sprawdza, co się zmieniło.

Kuba: To czemu, do jasnej ciasnej, nie pojawiają się zaktualizowane lub nowe elementy?

Łukasz: Przeglądarka nie czeka z renderowaniem strony na załadowanie wszystkich zaktualizowanych i nowych elementów. Po pobraniu zasoby siedzą zwarte i gotowe w pamięci podręcznej, ale nie zostaną wyświetlone aż do ponownego przeładowania strony.

Kuba: W takim razie jedyną możliwością zaktualizowania strony jest jej przeładowanie?

Łukasz: No cóż, mam chyba pomysł na obejście tego problemu w tej sytuacji. Napisałem taki mały skrypcik JavaScript...

Kuba: No jasne, jak zwykle JavaScript!

Łukasz: Dynamiczna część strony jest generowana przez skrypt PHP zawarty w pliku *list.php*. Dla każdego tartanu jest tworzony jeden element ``. Pozostała część strony jest statyczna. Tak naprawdę powinniśmy się zatroszczyć tylko o zawartość listy `` z tartanami.

Kuba: Prawda.

Łukasz: W moim kodzie korzystam z obiektu `window.applicationCache` i jego metod, by sprawdzić, czy coś się zmieniło w pamięci podręcznej, a tak jest w przypadku modyfikacji pliku manifestu. Zgadza się? Jeśli doszło do zmiany, trzeba pobrać zaktualizowaną zawartość ze skryptu *list.php*.

Kuba: Wszystko jasne. Jeśli plik manifestu został zmieniony, powinniśmy też zaktualizować listę tartanów. A najlepiej, żeby nie wymagało to przeładowania całej strony.

Łukasz: Zgadza się. Zatem jeśli doszło do zmiany pliku manifestu, mój kod wysyła żądanie AJAX do skryptu *list.php*. Ten skrypt nie znajduje się na liście manifestu i w przypadku bezpośredniego wywołania zwraca kod HTML zawierający listę aktualnych wzorów tartanów. Otrzymany kod wstawiam w miejsce dotychczasowego, czyli w miejsce listy tartanów. I to wszystko!



Jazda próbna

Skorzystaj z kodu JavaScript przygotowanego przez Łukasza i sprawdź, czy pomoże w poradzeniu sobie z problemami mechanizmu podręcznej pamięci aplikacji.

1 Skopiuj plik z kodem.

Plik *cache-manager.js* z katalogu *extras/js* skopiuj do katalogu *rozdzial7/js*.

2 Dołącz skrypt do stron Tartanatora.

W plikach *index.php*, *build.php*, *findevent.php* i *tartans.php* dołącz skrypt *cache-manager.js* (bezpośrednio pod znacznikiem dodającym framework jQuery Mobile).

```
<script src="js/cache-manager.js" type="text/javascript"></script>
```

3 W pliku tartans.php umieść dodatkowy kod JavaScript.

Ten kod zapewnia aktualną zawartość elementu *#tartans-list* znajdującego się w bloku *<div>* o identyfikatorze *#tartans_page* na podstawie danych zwracanych przez skrypt *inc/list.php*.

```
<div data-role="page" id="tartans_page">
  <script type="text/javascript" charset="utf-8">
    var tartanPage = $('#tartans_page');
    tartanPage.live('pageinit', function () {
      if (!tartanPage.data.cacheManager) {
        tartanPage.data.cacheManager = new CacheManager('#tartans_page');
        tartanPage.data.cacheManager.ensureFreshContent('#tartans-list', 'inc/list.php');
      }
    });
  </script>
<div data-role="header" data-position="fixed">
```



tartans.php

Nie istnieją,
głupie pytania

P: Skoro mogę użyć symbolu wieloznacznego w sekcji NETWORK pliku manifestu, to czy nie mógłbym skorzystać z niego w sekcji CACHE, by do trybu offline dołączyć wszystkie zasoby witryny?

U: Na szczęście (albo na nieszczęście) nie możesz użyć symboli wieloznacznym w sekcji CACHE. Symbol * możesz zastosować tylko w sekcji NETWORK i tak naprawdę to nie jest symbol wieloznacznym. Nie możesz go umieścić w dowolnej ścieżce — występuje zawsze samotnie i ma szczególne znaczenie (które można wyjaśnić tak: jeśli dany zasób nie jest uwzględniony w innej sekcji, pobierz go bezpośrednio z serwera).

P: Dlaczego strony z atrybutem manifest w znaczniku `<html>` są przechowywane w pamięci podręcznej nawet wtedy, gdy nie są uwzględnione w pliku manifestu?

U: Faktycznie może to być nie do końca zrozumiałe i powodować problemy. Chodzi tu o to, by zamiast pobierania i zapisywania w podręcznej pamięci całego mnóstwa stron i zasobów za jednym razem umożliwić ich „leniwe” dodawanie, czyli takie, w którym dana strona jest umieszczana w pamięci offline wtedy, gdy użytkownik odwiedza ją po raz pierwszy.

P: Co macie na myśli, mówiąc „za jednym razem”?

U: Kiedy przeglądarka odczyta nowy lub zaktualizowany plik manifestu, natychmiast zabiera się za pobieranie wszystkich nowych zasobów i sprawdzanie zmian w tych już istniejących. Koniecznie musisz to uwzględnić podczas definiowania plików manifestu.

P: W naszym pliku manifestu są uwzględnione pliki HTML i obrazy wszystkich tartanów. Czy to nie jest mnóstwo danych do pobrania „za jednym razem”?

U: Trochę się tym martwiłymi. Wygeneruje to całkiem sporo żądań HTTP. Jednak pliki są stosunkowo małe: większość obrazów ma rozmiar poniżej 1 kB, a pliki HTML jeszcze mniej. Obciążenie łączy nie będzie więc zbyt duże, zwłaszcza jeśli się weźmie pod uwagę to, że pliki są pobierane asynchronicznie. Użytkownik nie powinien tego raczej odczuć.

P: Czy przeglądarka ponownie pobiera wszystkie zasoby z pliku manifestu za każdym razem, gdy jest on aktualizowany?

U: Na szczęście nie. Przeglądarka sprawdza, czy zasoby, które znajdują się już w pamięci trybu offline, uległy zmianie. Jeśli nie, nie są ponownie pobierane.

P: Wszystko pięknie i ładnie, ale co się stanie, gdy przeglądarka użytkownika nie obsługuje mechanizmu podręcznej pamięci aplikacji?

U: Witryna będzie się zachowywała dokładnie tak samo jak wcześniej, zanim dodaliśmy plik manifestu. Nie dzieje się nic złego, ale brakuje oczywiście funkcjonalności trybu offline.

P: Nic nie wspomnieliście o sekcji FALLBACK.

U: Poza sekcjami CACHE i NETWORK jest jeszcze opcjonalna sekcja FALLBACK, która daje możliwość zdefiniowania wersji offline plików, kiedy są niedostępne ich wersje online (na przykład plik `login.html` może zostać podmieniony na `offline.html`).

P: Mechanizm appCache jest fajny, ale co z localStorage? W tej chwili to wygląda tylko na połowę opowieści.

U: Ej, mądralo, wyprzedziłeś materiał! Do tematu lokalnego składowania danych, czyli localStorage, wrócimy w rozdziale 8. (nie przejmuj się, jeżeli jeszcze o tym nie słyszałeś).



Obejrzyj to!

Uważaj, by nie nadużywać atrybutu manifest na stronach.

Już o tym wspomnieliśmy, ale musimy to powtórzyć: każda strona, która w znaczniku `<html>` ma atrybut manifest, trafia do pamięci trybu offline bez względu na to, czy znajduje się w pliku manifestu. Nie ma na to wpływu również sekcja NETWORK ani inne sztuczki.

Zwyciężyliśmy (w końcu)

Uff. Było z tym trochę problemów, ale w końcu Tartanator działa w trybie offline. To znaczy, że zakończyliśmy prace nad edytorem tartanów z drugiej fazy projektu.

- ☑ Ulepszyć istniejący formularz, by mógł wykorzystać możliwości oferowane przez nowsze przeglądarki mobilne.
- ☑ Napisać kod strony serwerowej, który jest odpowiedzialny za przetwarzanie danych z formularza oraz generowanie zasobów (obrazów, kodu HTML itp.) na potrzeby tartanów utworzonych przez użytkownika.
- ☑ Zapewnić możliwość pracy w trybie offline w tej części aplikacji.



Gratulacje! Udało Ci się zmusić tryb offline do prawidłowego działania.

Zdefiniowanie pliku manifestu w taki sposób, by tryb offline działał tak, jak tego chcesz, jest trudne. Tobie się udało!

w Twojej okolicy

Teraz kolej na szukanie wydarzeń

Kolejnym elementem drugiej fazy projektu jest stworzenie strony, która pozwala wyszukiwać wydarzenia związane z tartanami w najbliższej okolicy użytkowników.

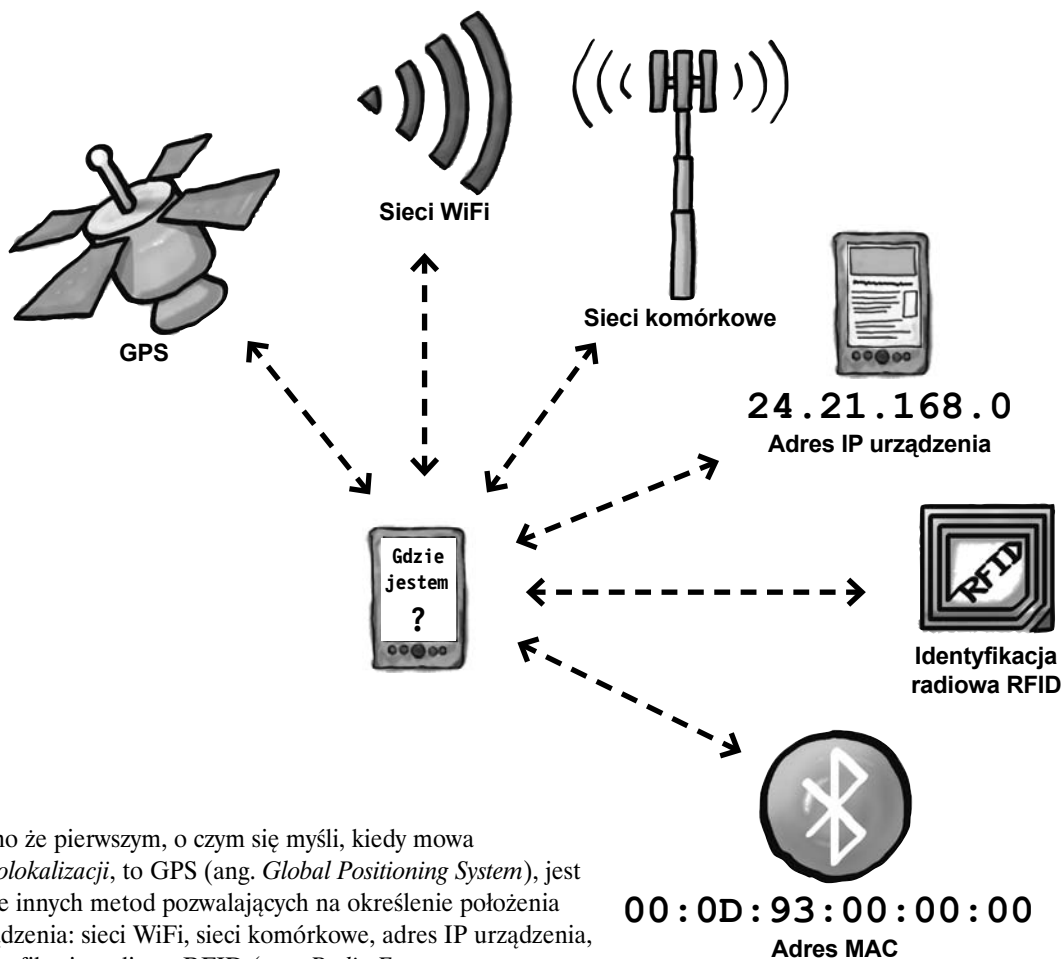


Formularz „Znajdź wydarzenie”, który zamierzamy stworzyć.

Chcemy wyszukiwać wydarzenia, które odbywają się blisko bieżącej lokalizacji użytkownika. To oznacza, że musimy porozmawiać o geolokalizacji.

Jak działa geolokalizacja?

Skorzystanie z *geolokalizacji* w przeglądarce internetowej wymaga użycia JavaScriptu do pobrania bieżącej lokalizacji urządzenia. Wiele nowoczesnych przeglądarek mobilnych ma zaimplementowane wsparcie dla interfejsu API geolokalizacji zgodnego ze specyfikacją W3C (ang. *World Wide Web Consortium*), umożliwiające stosunkowo proste odczytywanie danych geolokalizacyjnych. Jest też kilka innych rozwiązań pozwalających na geolokalizację, jak choćby Google Gears API czy interfejsy API konkretnych przeglądarek. Przeglądarki w starszych telefonach, a nawet w niektórych nowszych smartfonach w ogóle nie obsługują geolokalizacji.



Mimo że pierwszym, o czym się myśli, kiedy mowa o *geolokalizacji*, to GPS (ang. *Global Positioning System*), jest wiele innych metod pozwalających na określenie położenia urządzenia: sieci WiFi, sieci komórkowe, adres IP urządzenia, identyfikacja radiowa RFID (ang. *Radio Frequency Identification*), a także adresy MAC (ang. *Media Access Control*) urządzeń WiFi i Bluetooth.

Jak poprosić przeglądarkę o dane geolokalizacyjne?

Przeglądarki, które mają zaimplementowany interfejs API geolokalizacji zgodny z W3C, udostępniają obiekt `navigator.geolocation`. Najważniejszą metodą tego obiektu jest `getCurrentPosition`, która powoduje odczytanie bieżącej lokalizacji.

Spróbuj pobrać bieżące położenie urządzenia.

```
navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
```

Nazwa funkcji JavaScript, która ma zostać wywołana po określeniu położenia.

Nazwa funkcji JavaScript, która ma zostać wywołana, gdy wystąpi błąd.

Korzystanie z danych dostarczanych przez `getCurrentPosition`

Po pomyślnym określeniu położenia jest wywoływana funkcja zwrotna, której nazwę przekazaliśmy metodzie `getCurrentPosition`. Funkcja ta otrzymuje jako parametr obiekt `position`. Najbardziej przydatnymi danymi z tego obiektu są szerokość i długość geograficzna zapisane we właściwościach `latitude` i `longitude`, które znajdują się z kolei we właściwości `coords` obiektu `position`.

Funkcja zwrotna operacji zakończonej pomyślnie.

Obiekt position zawiera informacje, których potrzebujemy.

Najbardziej przydatną właściwością obiektu position jest coords...

```
function successCallback(position) {
    var coords = position.coords;
    alert(coords.latitude + ', ' + coords.longitude);
}
```

...a głównymi właściwościami obiektu coords są latitude i longitude.

Funkcja zwrotna, która jest wywoływana w przypadku wystąpienia błędu.

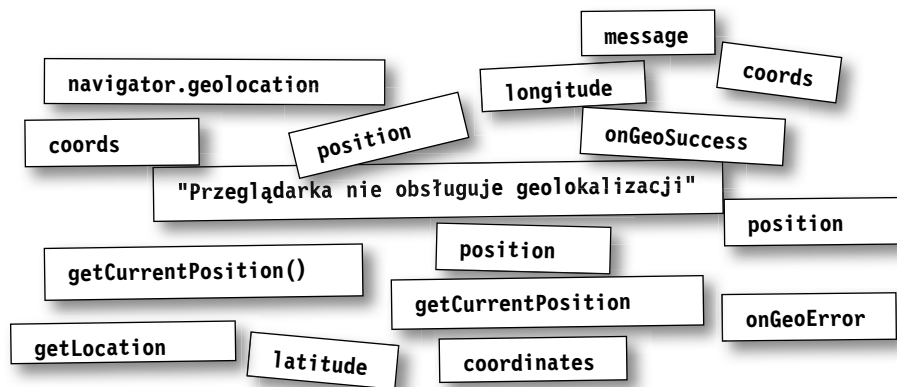
```
function errorCallback(error) {
    alert(error.message);
}
```

W obu tych funkcjach wyświetlamy po prostu okienko alert (co może być dosyć irytujące).



Magnesiki z kodem geolokalizacji w JavaScriptcie

Magnesiki z kodem umieść we właściwych miejscach na listingu przedstawionym poniżej. Każdego magnesu możesz użyć tylko raz, ale uważaj, bo niektóre z nich są zbędne.



```
function getLocation () {
    // Sprawdzamy, czy przeglądarka obsługuje geolokalizację
    if (.....) {
        navigator.geolocation.....(....., onGeoError);
    } else { // Nie obsługuje
        onGeoError(new Error(.....));
    }
}

function onGeoSuccess (.....) {
    var ..... = ..... coords;
    alert(coordinates..... + ',' + coordinates.....);
}

function ..... (error) {
    alert(error.....);
}

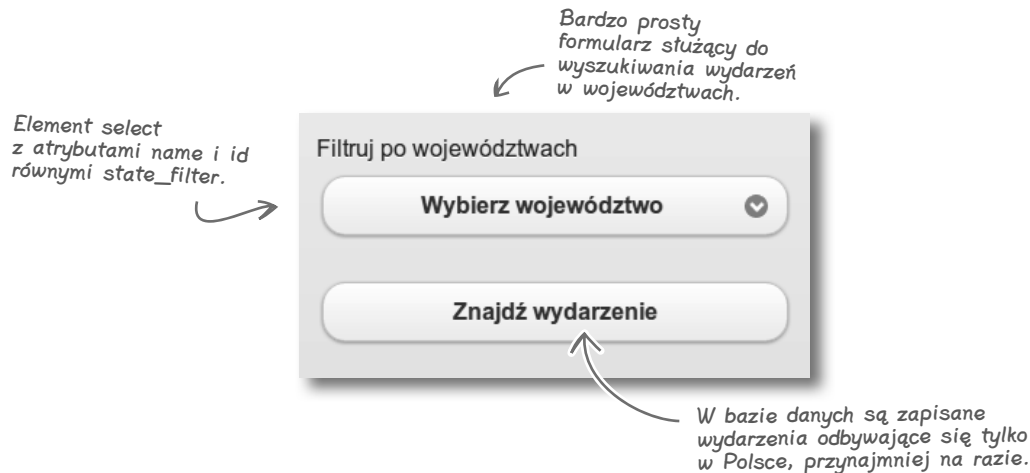
.....();
```

→ Odpowiedzi na stronie 302

Początek pracy nad stroną Znajdź wydarzenie — podstawy

Głównym zadaniem strony Znajdź wydarzenie jest... znajdowanie wydarzeń. Mamy już ogólny obraz tego, jak za pomocą JavaScriptu można określić aktualną lokalizację urządzenia. Musimy jednak zacząć od stworzenia podstawowej wersji strony. Co będzie, jeśli przeglądarka użytkownika nie obsługuje JavaScriptu lub geolokalizacji?

Zacniemy od utworzenia formularza służącego do wyszukiwania wydarzeń, który nie wymaga ani jednego, ani drugiego. Punktem wyjścia będzie pole wyboru z listą województw.



Ćwiczenie

Zbuduj prosty formularz do wyszukiwania wydarzeń w dokumencie *findevent.php*, który posłuży jako baza. Kod formularza umieść w bloku `<div>` z zawartością strony.

- 1** W znaczniku `<form>` ustaw atrybuty `method` na `GET`, `id` na `search_form`, a `action` na `events.php`.
- 2** Do formularza dodaj element `select`. Opcjami tego elementu mają być nazwy wszystkich 16 województw, a wartościami poszczególnych opcji powinny być trzyliterowe skróty nazw.
- 3** Przycisk zatwierdzający formularz powinien mieć identyfikator `search_submit`.

Zaraz Ci powiemy, gdzie możesz znaleźć plik `events.php`.

Gotową listę znajdziesz w pliku `województwa.txt` umieszczonym w katalogu `extras/events`.

Nie zapomnij umieścić pól w bloku `<div>` z atrybutem `data-role` ustawionym na `fieldcontain` (ze względu na `jQuery`).



Magnesiki z kodem geolokalizacji w JavaScriptcie. Rozwiązanie

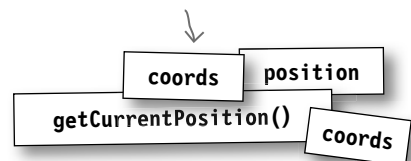
Udało Ci się umieścić magnesyki we właściwych miejscach?

```
function getLocation () {  
    // Sprawdzamy, czy przeglądarka obsługuje geolokalizację  
    if ( navigator.geolocation ) {  
        navigator.geolocation.getCurrentPosition ( onGeoSuccess , onGeoError );  
    } else { // Nie obsługuje  
        onGeoError(new Error( "Przeglądarka nie obsługuje geolokalizacji" ));  
    }  
}  
  
function onGeoSuccess ( position ) {  
    var coordinates = position.coords;  
    alert(coordinates.latitude + ',' + coordinates.longitude );  
}  
  
function onGeoError (error) {  
    alert(error.message );  
}  
  
getLocation ();
```

Kiedy już skompletujesz ten kod, **umieść go w pliku o nazwie `geolocation.js`** i zapisz w katalogu `rozdzial7/js`. Następnie dołącz go do pliku `findevent.php` za pomocą znacznika `<script>` umieszczonego w bloku `<div>` z identyfikatorem `data-role="page"`.

Powinien się znaleźć tuż nad formularzem.

Te magnesyki nie były potrzebne.



Dołączamy geolokalizację

Mamy już podstawową wersję formularza. A, i jeszcze jedno — bardzo denerwujące okienka wyświetlane przez JavaScript.

Teraz zajmiemy się ciekawszą częścią — zastosujemy geolokalizację w przeglądarkach, które ją obsługują. Po wprowadzeniu kilku zmian formularz będzie w nich wyglądać tak:

Kiedy już dopeścimy formularz, użytkownicy przeglądarek obsługujących geolokalizację będą mogli użyć swojego aktualnego położenia podczas wyszukiwania wydarzeń.



Kod podstawowej wersji formularza wygląda jak poniżej.

Ćwiczenie.

Rozwiązanie

```
<div data-role="content">
  <script type="text/javascript" src="js/geolocation.js"></script>
  <form method="get" action="events.php" id="search_form">
    <div data-role="fieldcontain">
      <label for="state_filter">Filtruj po województwach</label>
      <select id="state_filter" name="state_filter">
        <option value="">Wybierz województwo</option>
        <option value="DOL">dolnośląskie</option>
        <!-- itd. -->
      </select>
    </div>
    <div data-role="fieldcontain">
      <input type="submit" value="Znajdź wydarzenie" id="search_submit" />
    </div>
  </form>
</div><!-- /content -->
```

GEOLOKALIZACYJNE PRACE KONSTRUKCYJNE

Dopracujemy teraz kod skryptu *geolocation.js*. Zwróć szczególną uwagę na wyróżnione miejsca.



```
(function () {  
    var $page, $searchForm, $submitButton, $stateFilter;  
  
    $page = $('#event_page');  
    if (!$page.data.initialized) {  
        $page.live('pagecreate', initGeo); ← Po zainicjalizowaniu strony  
        $page.data.initialized = true;      wywołujemy funkcję initGeo.  
    }  
  
    function initGeo() {  
        $searchForm = $('#search_form');  
        $submitButton = $('#search_submit');  
        $stateFilter = $('#state_filter');  
        if (navigator.geolocation) {  
            initGeoOptions(); ← Jeśli przeglądarka obsługuje  
        }                                geolokalizację, inicjalizujemy  
    }                                elementy formularza związane  
    }                                z lokalizowaniem.  
  
    function initGeoOptions() {  
        var $latField, $longField, $flipSwitch; ← Dodaliśmy suwak stylizowany  
        $flipSwitch = $('<select name="usegeo" id="usegeo"      na przełącznik, dzięki  
data-role="slider"><option value="off">WYŁ.</option><option      któremu użytkownicy mogą  
value="on">WŁ.</option></select>').change(toggleLocation);  skorzystać z informacji  
        $flipSwitch.prependTo($searchForm).wrap('<div      o aktualnym położeniu.  
data-role="fieldcontain"></div>');  
        $flipSwitch.before('<label for="usegeo">Użyj mojej  
lokalizacji</label>');  
        $latField = $('<input type="hidden" />').attr({ name : ← Dodaliśmy dwa ukryte pola,  
'latitude', id : 'latitude'});                                które przechowują wartości  
        $longField = $('<input type="hidden" />').attr({name:    szerokości i długości  
'longitude', id : 'longitude' });                                geograficznej pobrane  
        $latField.appendTo($searchForm);                                z API geolokalizacji.  
        $longField.appendTo($searchForm);  
    }  
}
```

Funkcja `toggleLocation` jest podpięta do zdarzenia `change` suwaka.

```

function toggleLocation(event) {
  var geoActivated = ($(event.target).val() == 'on') ? true : false;
  if (geoActivated) {
    $submitButton.button('disable');
    $stateFilter.selectmenu('disable');
    $.mobile.showPageLoadingMsg();
    navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);
  } else {
    $stateFilter.selectmenu('enable');
    $submitButton.button('enable');
  }
}

function onGeoSuccess(position) {
  var coordinates = position.coords;
  $('#latitude').val(coordinates.latitude);
  $('#longitude').val(coordinates.longitude);
  $.mobile.hidePageLoadingMsg();
  $submitButton.button('enable');
}

function onGeoError(error) {
  $('#usegeo').val('off').trigger('change');
  alert(error.message);
}
})();

```

Jeśli użytkownik włączy geolokalizację, dezaktywowane są pole wyboru województwa i przycisk zatwierdzający (przycisk zostanie ponownie aktywowany po określeniu lokalizacji).

W tym miejscu uruchamiamy geolokalizację.

Po wyłączeniu przetącznika ponownie aktywujemy pole wyboru województwa.

To jest funkcja zwrotna wywoływana po pomyślnym określeniu położenia. Zmieniliśmy ją tak, by aktualizowała ukryte pola `#latitude` i `#longitude` otrzymanymi wartościami.

W przypadku błędu przetącznik jest wyłączony, a następnie jest wyzwalane zdarzenie `change` (do którego jest podpięta funkcja `toggleLocation`).

Sprawdź sam, czy to działa!
Pobaw się chwilę formularzem,
by poczuć na własnej skórze to, czego
dokonałiśmy dzięki JavaScriptowi.

Przedstawiony tu kod możesz znaleźć
w pliku `extras/js/enhanced_geo_form.js`.

Zaktualizuj plik `geolocation.js`, zapisz go, a następnie przejdź na zaktualizowaną stronę Znajdź wydarzenie.

Odwiedź bibliotekę

Kuba: Przed chwilą sprawdzałem, jak na różnych urządzeniach działa nasz nowy formularz. Wiem, że przeglądarka w BlackBerry OS 5 obsługuje geolokalizację, ale w formularzu nie pojawił się przełącznik *Użyj mojej lokalizacji*.

Łukasz: Tak, przeglądarka w BlackBerry OS 5 obsługuje geolokalizację, ale nie jest ona zgodna ze specyfikacją W3C. Poszperałem trochę na ten temat. Starsze przeglądarki w Androidzie też nie mają obsługi zgodnej z W3C, bo bazują na Google Gears.

Kuba: To wszystko wygląda na strasznie skomplikowane...

Łukasz: Bo takie jest. A nam się zbliża termin. Nie sądzę, żebyśmy mieli dość czasu na przyjrzenie się wszystkim niuansom, przetestowanie na wszystkich możliwych urządzeniach i poprawienie najdrobniejszych błędów. Znalazłem coś, co może nam pomóc. To mała javascriptowa biblioteka typu open source, która pośredniczy w dostępie do geolokalizacji na wielu różnych platformach — zarówno tych zgodnych z W3C, jak i niezgodnych. Nazywa się dosyć prosto: *geo-location-javascript*.

Kuba: Ale czy to przypadkiem nie znaczy, że musimy zrefaktoryzować cały do tej pory napisany kod JavaScript?

Łukasz: Nie. Interfejs API tej biblioteki jest zgodny ze specyfikacją W3C. Nazwy wszystkich ważniejszych metod i właściwości są identyczne. Sądzę, że wystarczy zmienić kilka linijek kodu, by przestać się martwić o międzyplatformową kompatybilność.

Kuba: Skoro już będziemy grzebać przy tym kodzie, może zrobimy coś z tymi okropnymi okienkami z komunikatami o błędach?

Łukasz: Słusznie. Myślę, że powinniśmy użyć okienek dialogowych z frameworku jQuery Mobile, z których korzystałem już w formularzu edytora tartanów.

Kuba: Nic mi o tym nie wspomniłeś! Jakie okienka?

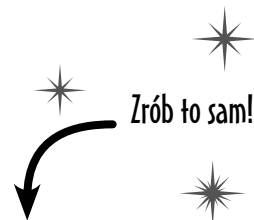
Łukasz: Jeśli użytkownik chce zatwierdzić formularz z tartanem, a nie poda jego nazwy, pojawia się okienko dialogowe informujące o błędzie. Wygląda całkiem ładnie — podobnie jak pozostałe elementy z jQM.

Kuba: A jak ci się udało wyświetlić te okna?

Łukasz: To są oddzielne strony jQM. Zresztą zajrzyj do katalogu *dialogs*, a sam zobaczysz. Wyświetlam je z poziomu JavaScriptu.

Kuba: Świetnie! Koniecznie muszę się temu przyjrzeć.

Łukasz: W porządku. Ja w tym czasie szybko dorzucę tę bibliotekę geolokalizacji.



Plik *extras/hs/geo.js* skopiuj do katalogu *rozdzial7/js*. To jest właśnie wspomniana biblioteka *geo-location-javascript*. Możesz też odwiedzić stronę projektu, jeśli chcesz się dowiedzieć więcej: <http://bit.ly/sx7JrH>.



Zaktualizuj stronę Znajdź wydarzenie oraz kod JavaScript geolokalizacji, tak by była wykorzystywana międzyplatformowa biblioteka, a błędy pojawiały się w okienkach jQM.

Ćwiczenie

- 1 Do pliku `findevent.php` dołącz skrypt JavaScript nowej biblioteki geolokalizacji.

```
<div data-role="content">
<script src="http://code.google.com/apis/gears/gears_init.js" type="text/
javascript" charset="utf-8"></script>
<script src="js/geo.js" type="text/javascript" charset="utf-8"></script>
<script type="text/javascript" src="js/geolocation.js"></script>
```

- 2 Zmodyfikuj plik `geolocation.js`, by wykorzystywał nową bibliotekę.

Znajdź poniższy kod:

```
if (navigator.geolocation) {
  initGeoOptions();
}
```

i zmień go na:

```
if (geo_position_js.init()) {
  initGeoOptions();
}
```

Zmień również to:

```
navigator.geolocation.getCurrentPosition(onGeoSuccess, onGeoError);
```

na to:

```
geo_position_js.getCurrentPosition(onGeoSuccess, onGeoError);
```

- 3 Utwórz okno dla błędów geolokalizacji i dodaj kod wywołujący.

Posłuż się plikami z katalogu `dialogs` jako wzorem i utwórz ładne okno informujące o błędach geolokalizacji. Zapisz je w pliku `geolocation_error.html`.

Następnie wywołanie `alert` w funkcji `onGeoError` zamień na wyświetlenie utworzonego okna. Przykład odpowiedniego kodu znajdziesz w pliku `js/tartanator.js`. Wystarczy, że zmienisz adres URL.



Rozwiązanie ćwiczenia

Funkcja `onGeoError` po zamianie zwykłego okienka `alert` na okno jQuery Mobile powinna wyglądać jak poniżej. Nie zapomnij utworzyć pliku HTML dla tego okna!

```
function onGeoError(error) {  
    $('#usegeo').val('off').trigger('change');  
    $.mobile.changePage( "dialogs/geolocation_error.html", {  
        transition: "pop",  
        reverse: false,  
        role: 'dialog'  
    });  
}
```



Przedarliśmy się biegiem przez gąszcz JavaScriptu.

Spokojnie Przyznajemy, że ostatnie przeżycia były dosyć intensywne. Jest też kod, którego dokładnie nie omówiliśmy. Ale to nie jest przecież książka *Head First JavaScript. Edycja polska*, a my nie chcemy odchodzić znowu od głównego zadania: wygrania walki z mechanizmem podręcznej pamięci aplikacji. Chcemy też zająć się interfejsem `mediaCapture` dostępnym w technologii PhoneGap (nawet jeśli jeszcze nie wszystkie przeglądarki go wspierają).



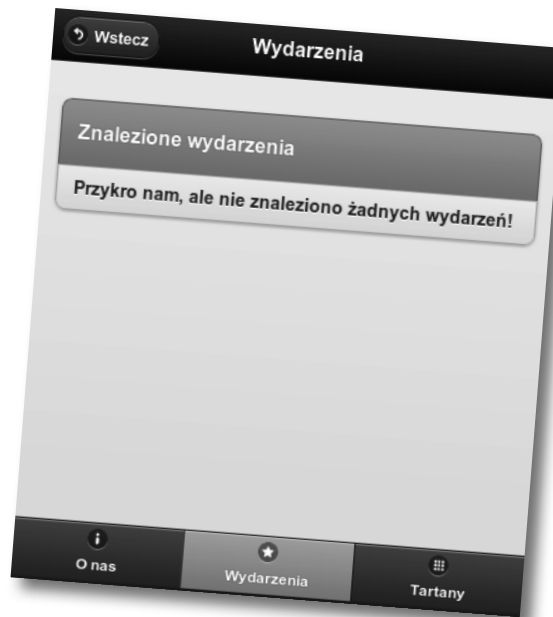
Jazda próbna

Na razie wystarczy! Zerknijmy, jak to się sprawdza w praktyce. Dołożenie kodu wyszukującego wydarzenia wraz z przykładowymi danymi nie zajmie Ci wiele czasu.

- 1 Skopiuj kilka niezbędnych plików z katalogu `extras/events`.**
Do katalogu `rozdzial7/inc` skopiuj pliki `event_search.inc` i `event_list.inc`. Plik `events.php` umieść w katalogu `rozdzial7`.
- 2 Sprawdź, jak to działa!**
W przeglądarce mobilnej przejdź na stronę `Znajdź wydarzenie` i włącz przełącznik *Użyj mojej lokalizacji*.

Nic nie znalazł

Ha, ha! Przepraszamy za to. Najprawdopodobniej otrzymasz taki wynik, chyba że jesteś akurat w okolicy miejsc, które zdefiniowaliśmy w pliku `event_list.inc`.



Najprawdopodobniej otrzymasz właśnie taki wynik.

Jest to spowodowane doбором przykładowych danych oraz ustalonym promieniem poszukiwań (50 km). Co Ty na to, żeby dodać własne definicje wydarzeń, by otrzymać jakieś wyniki?



Ćwiczenie

Dodaj kilka przykładowych definicji wydarzeń odbywających się w Twojej okolicy. Otwórz plik `event_list.inc` umieszczony w katalogu `rozdzial7/inc/`. Znajduje się w nim tablica zawierająca definicje wydarzeń. Do tablicy dodaj kilka elementów z adresami i współrzędnymi odpowiadającymi Twojej najbliższej okolicy.

Jeżeli nie najlepiej czujesz się z PHP, możesz jedynie zmodyfikować istniejące definicje. Najważniejsze dane używane do wyszukiwania to szerokość i długość geograficzna (`latitude` i `longitude`) oraz województwo (`state`).

W szybkim określeniu współrzędnych geograficznych może pomóc narzędzie dostępne pod adresem <http://itouchmap.com/latlong.html>.



Chtopaki, to wygląda wyśmienicie!
Właśnie tak to sobie wyobrażałem,
kiedy pierwszy raz wpadł mi do głowy
pomysł na mobilną aplikację.

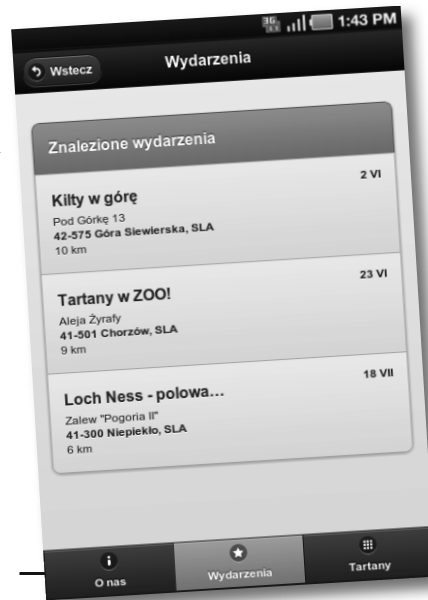
Super! Udało nam się zintegrować kilka elementów, dzięki którym możemy o naszej mobilnej aplikacji powiedzieć „wyjątkowa”:
stopniowe ulepszanie, tryb offline i wykorzystanie geolokalizacji.



WYSIŁ SZARE KOMÓRKI

Zadanie dodatkowe: w jaki sposób można by dodać do formularza pole typu range, by użytkownik mógł ustalać promień poszukiwań? Kiedy pole powinno się pojawiać? Kod działający po stronie serwera jest już przygotowany, więc wystarczy, że dodasz pole o nazwie radius.

Wygląda dobrze w wielu różnych przeglądarkach mobilnych.



Tak wygląda w Androidzie.



CELNE SPOSTRZEŻENIA

- Mobilne aplikacje internetowe, które są wygodne i zachowują się podobnie jak aplikacje natywne, często charakteryzują się **stopniowym ulepszeniem, dobrze działającym trybem offline oraz wykorzystaniem geolokalizacji**.
- Rozpoczynanie od **podstawowej wersji** pozwala na dotarcie do tak wielu użytkowników, jak to tylko możliwe. Dzięki zastosowaniu **stopniowego ulepszenia** możemy przygotować olśniewającą aplikację dla bardziej wydajnych urządzeń.
- Dostarczenie możliwości **pracy offline** jest bardzo ważne, ponieważ w przypadku urządzeń mobilnych trudno zapewnić stały dostęp do internetu.
- Aby skorzystać z mechanizmu **podręcznej pamięci aplikacji**, musimy utworzyć **plik manifestu** informujący przeglądarkę o zasobach, które mają być dostępne offline.
- Budowa pliku manifestu jest prosta, ale trzeba uważać na **wiele pułapek** tego mechanizmu.
- W pliku manifestu możemy wskazać zasoby, które mają zostać pobrane do podręcznej pamięci i być dostępne offline (**sekcja CACHE**), jak również te, które za każdym razem — przy aktywnym połączeniu — mają być pobierane z serwera (**sekcja NETWORK**).
- **Geolokalizacja** jest obsługiwana przez większość nowoczesnych smartfonów. W wielu z nich zaimplementowano **interfejs API zgodny ze specyfikacją W3C**, który udostępnia z poziomu JavaScript obiekt `navigator.geolocation`.
- Formularz Znajdź wydarzenie zintegrowaliśmy z geolokalizacją w taki sposób, by była dostępna na tych urządzeniach, które ją obsługują. Zrobiliśmy to w podobny sposób jak w przypadku formularza edytora tartanów.
- Niektóre przeglądarki mobilne, zwłaszcza dostępne na starszych smartfonach, **mają zaimplementowaną geolokalizację niezgodną z W3C**. Aby zapewnić ich obsługę, skorzystaliśmy z zewnętrznej biblioteki `geo-location-javascript`.
- **Biblioteka geo-location-javascript** emuluje API W3C, więc nie wymaga wprowadzania większych zmian w istniejącym kodzie JavaScript nastawionym na API zgodne z W3C.

Skorowidz

@import, 37
@media, 12, 13
<a>, znacznik, 119
, znacznik, 68, 69
<meta>, znacznik, 22, 72, 73, 89
<tbody>, znacznik, 124
<thead>, znacznik, 124
4G, telefon, 65

A

a, znacznik, 119
accesskey, atrybut, 117, 119, 122
adb, 411
addColor(), 279
adres IP, 394
Affero General Public License, *Patrz* AGPL
AGPL, 158
AJAX, 236
algorytmy dopasowujące, 159
Android, 3, 319
 plik pakietu, 415
 SDK, 404, 406, 411
 android, plik, 405
 instalowanie aplikacji, 411
 instalowanie platform i narzędzi, 405, 406
 odinstalowywanie aplikacji, 411
 pobieranie, 404
 ścieżka środowiskowa PATH, 412, 413
 tworzenie wirtualnych urządzeń, 407, 408
 uruchamianie urządzeń, 409
Android debug bridge, *Patrz* adb
Apache, 390
Apache Cordova, 320
apachefriends.org, 390, 391
API, 95

API mediaCapture, 344, 348, 355
API urządzenia, 384
APK, 328, 415
aplikacja internetowa, 219, 264
 cechy dobrej aplikacji mobilnej, 270
 sklepy, 385
 sprzedaż, 385
 testowanie na urządzeniach mobilnych, 374, 375
 testowanie na własnym serwerze www, 394
aplikacje hybrydowe, 316, 317, 320, 355
aplikacje mobilne, 270
appCache, *Patrz* plik manifestu
Apple, 2
Application Cache, *Patrz* podręczna pamięć aplikacji
arkusze stylów
 czcionki, 35
 display, właściwość, 54
 em, jednostki, 37
 flexbox, 65
 jQuery Mobile, 239
 kolejność, 61
 media, 12, 13, 14
 cechy, 12, 41
 deklaracja w arkuszu stylów, 12, 13
 deklaracja w znaczniku link, 12
 overflow, właściwość, 131
 płynna siatka, 25, 27, 28, 29, 41
 wzór płynności, 28
 płynne obrazy, 32, 33, 41
 procenty, jednostki, 37
 src, atrybut, 68
 sztywna siatka, 25, 26
atrybuty
 accesskey, 117, 119, 122
 class, 57

data-*, 231, 236
data-filter, 244
data-icon, 250
data-inset, 233
data-position, 239
data-rel, 262
data-role, 229, 258
 collapsible, 322
 content, 229
 fieldcontain, 258
 list-divider, 244
 listview, 232
 navbar, 249
 page, 235
data-theme, 239
font-size, 41
height, 33
id, 57
placeholder, 257
src, 68
step, 265
width, 33
audio, 71
AVD, 329

B

bazy danych, urządzenia mobilne, 154
BlackBerry
 geolokalizacja, 306
 PhoneGap Build, 331
Blaze, 47
bookmarklet, 85
browserscope.org, 383
buildAddButton(), 279

C

cache manifest, *Patrz* plik manifestu
CACHE, sekcja, 289, 290, 296, 311
caniuse.com, 382

Skorowidz

captureImage, metoda, 348, 355
CDN, 229

cechy mediów, 12, 41
change, zdarzenie, 279
Charles Proxy, 65

Chrome

chrome-resizer, 85
plik manifestu, 291
Web Developer Toolkit, 85

C-HTML, 117

ciasteczka, 334

class, atrybut, 57

click, zdarzenie, 279

CMS, 95

Compact HTML, *Patrz* C-HTML

Content Delivery Network, *Patrz* CDN

content-type, nagłówek, 287

coords, właściwość, 299

CSS

czcionki, 35

display, właściwość, 54

em, jednostki, 37

flexbox, 65

kolejność, 61

media, 12, 13, 14

cechy, 12, 41

deklaracja w arkuszu stylów,
12, 13

deklaracja w znaczniku link, 12

overflow, właściwość, 131

płynna siatka, 25, 27, 28, 29, 41

wzór płynności, 28

płynne obrazy, 32, 33, 41

procenty, jednostki, 37

src, atrybut, 68

szywna siatka, 25, 26

wersje, 37

CSS Mobile Profile 2.0, 127, 131, 135

list-style-position, właściwość, 132

zaokrąglenie wierzchołków, 129

CSS3, 37

wsparcie w przeglądarkach, 37

CSS-MP, *Patrz* CSS Mobile Profile 2.0

CustomDevice, obiekt, 189

czcionki, 35

rozmiar, 35

D

DAP, 384

DaRadę! Przygotowanie do testów,
strona, 152

rozpoznawanie telefonu, 153

data-*, atrybuty, 231, 236

data-filter, atrybut, 244

data-icon, atrybut, 250

data-inset, atrybut, 233

data-position, atrybut, 239

fixed, 239

data-rel, atrybut, 262

back, 262

data-role, atrybut, 229

collapsible, 322

content, 229

fieldcontain, 258

list-divider, 244

listview, 232

navbar, 249

page, 229, 235

data-theme, atrybut, 239

debug.phonegap.com, 377, 381

debugowanie, 376

desktop, klasa urządzeń, 185

Device Anywhere, 375

Device APIs Working Group, *Patrz*

DAP

device.php, 163, 165

deviceready, zdarzenie, 345, 355

display, właściwość, 54

DOCTYPE, 116

dopasowujące, algorytmy, 159

DTD, 116

duże obrazy, 54

E

ECMAScript Mobile Profile, 135

EDGE, 65

efekty przejścia, 245

em, jednostki, 37

emulator, 329

aplikacja kamery, 353

F

facebookexternalhit, 210, 211, 213

FALLBACK, sekcja, 296

Firefox

plik manifestu, 291

Web Developer Toolkit, 85

Firtman, Maximiliano, 374, 381

Flash, 35

flexbox, *Patrz* Flexible Box Layout

Flexible Box Layout, 65

font, rozmiar, 35, 37

font-size, atrybut, 41

formularze, 256

input, 257

placeholder, atrybut, 257

range, pole, 260

struktura w aplikacji

Tartanator, 256

textarea, 257

frameworki, 225, 227

jQuery Mobile, 226, 227, 230, 235,
236, 264

wybór, 225

future friendly, manifest, 362, 363, 366

G

generic, identyfikator, 210

generic_web_browser, identyfikator, 211

geolocation-javascript, biblioteka, 311

geolokalizacja, 298, 299, 311

BlackBerry, 306

coords, właściwość, 299

dane geolokalizacyjne z

przeglądarki, 299

getCurrentPosition, metoda, 299

latitude, właściwość, 299

longitude, właściwość, 299

navigator.geolocation, obiekt, 299

getCapability, metoda, 171, 177

getCurrentPosition, metoda, 299

getDeviceForHttpRequest,

metoda, 172

getDeviceForUserAgent, metoda, 164

Global Positioning System, *Patrz* GPS

Google
 Gears, 298
 mapy, 53, 75, 79, 80
 GPS, 298
 Guda, Krishna, 158

H

HAR, plik, 49
 has_cellular_radio, 176
 haz.io, 383
 hCard, 369
 height, atrybut, 33
 higher_mobile, klasa urządzeń, 184, 185, 186
 HTML5, 219, 264
 formularze, 256
 placeholder, atrybut, 257
 range, pole, 260
 placeholder, atrybut, 257
 podręczna pamięć aplikacji, 284
 range, pole, 260
 html5test.com, 383
 hybrydowe, aplikacje, 317, 320, 355

I

id, atrybut, 57
 ifconfig, 394
 iframe
 mapy, 53
 wideo, 35
 img, znacznik, 68, 69
 interfejs programowania aplikacji,
Patrz API
 Internet Explorer, 65
 komentarze warunkowe, 62, 63, 89
 zapytania o media, 62
 iOS Developer Program, 319
 IP, adres, 394
 ipconfig, 394
 iPhone, 2
 is_wireless_device, 172

J

JavaScript
 mobilna wersja, 135

pobieranie na urządzeniach
 mobilnych, 78
 pseudozapytanie o media, 76
 umieszczanie mapy na stronie, 74, 75, 76, 77
 umieszczanie na stronie, 77
 jednostki
 em, 37
 procenty, 37
 jQM, *Patrz* jQuery Mobile
 jQuery
 toggle, metoda, 341
 toggleClass, metoda, 341
 jQuery Mobile, 226, 227, 235, 236, 264
 AJAX, 235
 arkusz stylów, 239
 data-*, atrybuty, 231, 236
 data-filter, atrybut, 244
 data-icon, atrybut, 250
 data-inset, atrybut, 233
 data-position, atrybut, 239
 data-rel, atrybut, 262
 data-role, atrybut, 229
 collapsible, 322
 content, 229
 fieldcontain, 258
 list-divider, 244
 listview, 232
 navbar, 249
 page, 229, 235
 data-theme, atrybut, 239
 dodawanie miniaturek, 241
 domyślny wygląd, 230
 efekt przejścia, 245
 fieldcontain, 258
 filtry, 244
 lista, 232, 233
 listview, 233
 odsyłacze, 234
 pagecreate, zdarzenie, 275
 pageinit, zdarzenie, 275
 pasek narzędzi w stopce, 249, 250
 pasek nawigacji, 249
 podział listy na sekcje, 244
 prosta strona, 228, 229, 230

przeźreń wokół elementów
 nagłówkowych, 237
 ready, metoda, 275
 Theme Roller, 239
 wersje, 236
 widżet pola wyboru koloru, 275, 276
 wsparcie przez przeglądarki, 245
 wsparcie przez urządzenia, 245

K

Kamerman, Steve, 158
 klasa urządzeń, 180, 181, 186, 215
 definiowanie, 181
 klawisze dostępu, 119
 Koch, Peter-Paul, 189
 komentarze warunkowe, 62, 63, 89

L

latitude, właściwość, 299
 Linuks, XAMPP, 391
 list-style-position, właściwość, 132
 localStorage, 334, 335, 343, 355
 bezpieczeństwo, 343
 clear, metoda, 338
 getItem, metoda, 341
 gettery, 335
 limit danych, 343
 obsługa przez przeglądarkę, 339, 343
 settery, 335
 zapisywanie obrazów, 343
 lokalne składowanie danych, 334
 longitude, właściwość, 299

M

Mac
 Apache, 393
 MAMP, 392, 393, 395
 ścieżka PATH, 413
 WURFL, 401
 MAMP, 392, 393, 395
 manifest “future friendly”, 362, 363, 366
 manifestu, plik, 284, 286, 311
 CACHE, sekcja, 289, 290, 296, 311
 Chrome, 291
 FALLBACK, sekcja, 296

Skorowidz

- manifestu, plik
 - Firefox, 291
 - NETWORK, sekcja, 289, 311
 - odświeżenie, 290
 - Safari, 291
 - składnia, 285
 - walidator, 292
 - wsparcie przez przeglądarki, 284
 - Maps.gstatic.com, 52
 - mapy
 - iframe, 53
 - odsyłacz, 75
 - płynne, 79, 80
 - ukrywanie, 53
 - Marcotte, Ethan, 10
 - media queries, *Patrz* zapytania o media
 - mediaCapture API, 344, 348, 355
 - meta, znacznik, 22, 72, 73, 89
 - mikroformaty, 57
 - Mobile First, 43, 56, 57, 89
 - zapytania o media, 61
 - Mobile Perf Bookmarklet, 65
 - mobilne, przeglądarki, 6
 - interfejs użytkownika, 6
 - szybkość, 6
 - wsparcie nowych technologii, 6
 - mobilne, urządzenia
 - aplikacje hybrydowe, 316, 317, 320, 355
 - bazy danych, 154
 - bezpieczeństwo, 320
 - EDGE, 65
 - Flash, 35
 - geolokalizacja, 298, 299, 311
 - klawisze dostępu, 119
 - nawiązywanie połączeń
 - telefonicznych, 176, 178
 - przebudowa strony, 14
 - przekierowanie na oddzielną witrynę, 96, 97, 104, 105, 108
 - przewijanie stron, 119
 - skrypt wykrywający, 105, 106, 107
 - smartfony, 113
 - testowanie aplikacji, 374, 375
 - tryb offline, 284, 286, 311
 - ukrywanie mapy, 53
 - wspieranie, 139, 140
 - wydajność, 46, 47, 48, 49, 50, 51, 52, 65
 - Mobitest, 47, 52, 65
 - czas ładowania, 47
 - HAR, plik, 49
 - Show Statistics, 50
 - wykres kaskadowy, 48, 51
 - Modernizr, 383
 - MySQL, 390
- ## N
- navigator.device.capture.
 - captureImage, metoda, 355
 - navigator.geolocation, obiekt, 299, 311
 - getCurrentPosition, metoda, 299
 - NETWORK, sekcja, 289, 311
- ## O
- obrazy
 - duże, 54
 - height, atrybut, 33
 - optymalizacja, 67, 68, 69
 - płynne, 32, 33, 41
 - pobieranie przez przeglądarkę, 71
 - Sencha.io Src, 68, 69, 71
 - src, atrybut, 68
 - width, atrybut, 33
 - odsyłacze, tworzenie, 234
 - offline, tryb, 284, 286, 311
 - onColorListChange(), 279
 - onStitchSizeChange(), 279
 - Opera Dragonfly, 65, 381
 - Opera Mini, 110, 112, 120
 - symulator, 111
 - Opera Mobile, 110
 - optymalizacja obrazów, 67, 68, 69
 - overflow, właściwość, 131
- ## P
- pagecreate, zdarzenie, 275, 277, 279
 - pageinit, zdarzenie, 275, 277, 279
 - Passani, Luca, 158
 - PATH, zmienna, 412
 - Perfecto Mobile, 375
 - PhoneGap, 318, 355
 - cena, 320
 - deviceready, zdarzenie, 345, 355
 - robienie zdjęć, 345
 - wsparcie, 320
 - PhoneGap Build, 318, 320, 321, 324, 355
 - BlackBerry, 331
 - config.xml, 326
 - ekran powitalny aplikacji, 332
 - nowa aplikacja, 327
 - phonegap.js, 344, 355
 - plik konfiguracyjny, 326
 - pobieranie utworzonej aplikacji, 328
 - przebudowa aplikacji, 332
 - wsparcie, 320
 - phonegap.js, 344, 355
 - PHP
 - sprawdzenie wersji, 396
 - włączenie zgłaszania błędów, 401
 - phpinfo, 396
 - placeholder, atrybut, 257
 - plik manifestu, 284, 286, 311
 - CACHE, sekcja, 289, 290, 296, 311
 - Chrome, 291
 - FALLBACK, sekcja, 296
 - Firefox, 291
 - NETWORK, sekcja, 289, 311
 - odświeżenie, 290
 - Safari, 291
 - składnia, 285
 - walidator, 292
 - wsparcie przez przeglądarki, 284
 - płynne układy, 24, 25, 27, 28, 29, 32, 41
 - wzór płynności, 28
 - płynności, wzór, 28
 - Pod Paradnym Morsem, strona, 4
 - arkusz CSS, 16
 - czcionki, 35
 - meta, znacznik, 22
 - Mobile First, 58
 - obrazy, 32
 - płynny układ, 34
 - struktura witryny, 15

- umieszczanie mapy na stronie, 74, 75, 76, 77
 - wideo z Youtube'a, 35
 - wydajność, 48, 49, 50, 52
 - wygląd w przeglądarkach
 - mobilnych, 5, 8
 - zmiany, 17, 18
 - podręczna pamięć aplikacji, 284, 286, 311
 - połączenia telefoniczne, nawiązywanie, 176, 178
 - PPK, 382
 - procenty, jednostki, 37
 - progressive enhancement, *Patrz* stopniowe ulepszanie
 - proxy, serwer, 46, 65
 - przeglądarki
 - bookmarklet, 85
 - dane geolokalizacyjne, 299
 - listy możliwości, 382, 383
 - obsługa local storage, 339, 343
 - Opera Mini, 120
 - pobieranie obrazów, 71
 - proxy, 112
 - przeglądarki mobilne, 6
 - CSS Mobile Profile, 127
 - interfejs użytkownika, 6
 - najpopularniejsze, 110
 - Opera Mini, 110, 112
 - symulator, 111
 - Opera Mobile, 110
 - szybkość, 6
 - wsparcie dla HTML5, 227
 - wsparcie nowych technologii, 6
 - wykrywanie, 105, 106, 107
- Q**
- QuirksBlog, 189
 - quirksmode.org, 382
- R**
- range, pole, 260
 - RDW, 10, 24, 41, 71
 - Mobile First, 56, 89
 - przykład strony, 11
 - stosowane techniki, 10
 - ready, metoda, 275
 - Reduction In String, *Patrz* RIS
 - Responsive Web Design, 10, 24, 41, 71
 - Mobile First, 56, 89
 - przykład strony, 11
 - stosowane techniki, 10
 - RESS, 386
 - Rieger, Bryan, 386
 - Rieger, Stephanie, 386
 - RIS, 159
 - rozmiar czcionki, 35
- S**
- Safari, plik manifestu, 291
 - ScientiaMobile, 158
 - SDK Androida, 406
 - android, plik, 405
 - instalowanie aplikacji, 411
 - instalowanie platform i narzędzi, 405, 406
 - odinstalowywanie aplikacji, 411
 - pobieranie, 404
 - ścieżka środowiskowa PATH, 412, 413
 - tworzenie wirtualnych urządzeń, 407, 408
 - uruchamianie urządzeń, 409
 - semantyczne, znaczniki, 57
 - Sencha.io Src, 68, 69, 71
 - serwer obrazów, 68
 - serwer proxy, 46, 65
 - serwer WWW, 389
 - dostęp z urządzeń mobilnych, 394
 - setColorSelectStyle(), 279
 - SGML, 116
 - simpler_mobile, klasa urządzeń, 184, 185, 186
 - skalowanie, 72, 73
 - blokowanie, 73
 - smartfony, 113
 - sprzedaż aplikacji, 385
 - src, atrybut, 68
 - Standard Generalized Markup Language, *Patrz* SGML
 - step, atrybut, 265
- stopniowe ulepszanie, 55, 57**
- strona internetowa, 264
 - przekierowanie urządzeń
 - mobilnych, 96, 104, 105, 108
 - przewijanie strony, 119
 - przypominająca aplikację, 221
 - skalowanie, 72, 73
 - blokowanie, 73
 - testowanie na urządzeniach
 - mobilnych, 374, 375
 - testowanie na własnym serwerze
 - www, 394
 - walidacja, 123
 - zachowanie, 220
 - styleColorListItem(), 279
 - symulator Opery Mini, 111
 - system zarządzania treścią, *Patrz* CMS sztywna, siatka, 25, 26
- T**
- Tartanator, aplikacja, 222, 223, 227
 - formularze, 253, 254, 255, 256, 258, 259, 260
 - generate.php, 281
 - geolocation.js, 304, 305
 - manifest.appcache.php, 288, 292
 - plan projektu, 224
 - tartans.html, 240
 - tartans-list.txt, 243
 - tbody, znacznik, 124
 - technologie mobilne, 3
 - telefony
 - 4G, 65
 - EDGE, 65
 - nawiązywanie połączeń, 176, 178
 - smartfony, 113
 - zwykle, 113
 - Tera-WURFL, 156
 - thead, znacznik, 124
 - Theme Roller, 239
 - toggle, metoda, 341
 - toggleClass, metoda, 341
 - transkoder, 110
 - Trasatti, Andrea, 158
 - tryb offline, 284, 286, 311

Skorowidz

typy mediów, 12, 13
cechy, 12, 41
deklaracja w arkuszu stylów, 12, 13
deklaracja w znaczniku link, 12

U

UAProf, 103
ui-btn-active, klasa, 250
układy
dopasowujące się do szerokości
okna, 27
o stałej szerokości, 26
oparte na tabelach, 118
płynne, 24, 25, 27, 28, 29, 32, 41
wzór płynności, 28
sztywne, 25, 26
urządzenia mobilne
aplikacje hybrydowe, 316, 317,
320, 355
bazy danych, 154
bezpieczeństwo, 320
EDGE, 65
Flash, 35
geolokalizacja, 298, 299, 311
klawisze dostępu, 119
nawiązywanie połączeń
telefonicznych, 176, 178
przebudowa strony, 14
przekierowanie na oddzielną
witrynę, 96, 97, 104, 105, 108
przewijanie stron, 119
skrypt wykrywający, 105, 106, 107
smartfony, 113
testowanie aplikacji, 374, 375
tryb offline, 284, 286, 311
ukrywanie mapy, 53
wspieranie, 139, 140
wydajność, 46, 47, 48, 49, 50, 51,
52, 65
User-Agent, 97, 98, 99, 100, 101, 103, 104
pl-PL, 103
U, 103
user-agent sniffing, 101, 134
user-agent spoofing, 101

Ustrzel tartan!, aplikacja, 322, 323
anatomia projektu, 324

V

viewport, 22, 72, 73, 89

W

W3C Markup Validation Service, 123
WAC, 384
walidacja
pliku manifestu, 292
witryny, 123
Web Developer Toolkit, 85
Web Inspector, 286
Web INspector REmote, *Patrz* *weinre*
WebGL, 141
WebKit, 189
Web Inspector, 286
weinre, 65, 376, 379
bezpieczeństwo, 381
działanie, 376
uruchomienie, 377
Wholesale Applications Community,
Patrz WAC
video, 71
Youtube, 35
width, atrybut, 33
widzety, 324, 355
podstawowe pliki, 324
pola wyboru koloru, 275, 276
window.applicationCache, obiekt, 284
window.localStorage, obiekt, 339
Windows
serwer Windows IIS, 393
ścieżka PATH, 413
WURFL, 401
XAMPP, 390, 393, 395
WIP, *Patrz* Wireless Industry
Partnership
Wireless CSS, 135
Wireless Industry Partnership, 385
Wireless Markup Language, *Patrz* WML
Wireless Universal Resource File,
Patrz WURFL

witryna internetowa, 264
przekierowanie urządzeń
mobilnych, 96, 104, 105, 108
przewijanie stron, 119
przypominająca aplikację, 221
skalowanie, 72, 73
blokowanie, 73
testowanie na urządzeniach
mobilnych, 374, 375
testowanie na własnym serwerze
www, 394
walidacja, 123
zachowanie, 220
właściwości
coords, 299
display, 54
is_wireless_device, 172
latitude, 299
list-style-position, 132
longitude, 299
overflow, 131
WML, 117
Wroblewski, Luke, 386
WURFL, 155, 156, 157, 158, 175, 210,
215, 398, 401
API, 159, 160, 215, 398
CustomDevice, obiekt, 189
dostęp do zasobów, 401
eksplorator, 160
etapy tworzenia, 160
przygotowanie środowiska, 161
struktura katalogów, 161
ulepszanie, 168
generic, identyfikator, 210
getCapability, metoda, 171, 177
getDeviceForHttpRequest,
metoda, 172
getDeviceForUserAgent,
metoda, 164
has_cellular_radio, 176
is_wireless_device, właściwość, 172
licencja, 158
pobieranie API, 398
pobieranie pliku XML z danymi
WURFL, 399

ścieżka do kodu WURL API dla PHP, 402
 wurfl-config.xml, 400
 xmlhttp_make_phone_call_string, 176, 178
 xmlhttp_make_phone_string, 177
 WWW serwer, 389
 dostęp z urządzeń mobilnych, 394
 wydajność, 45, 46, 47, 48, 49, 50, 51, 52
 przeglądarki mobilne, 6
 wykres kaskadowy, Mobitest, 48, 51
 wzór płynności, 28

X

XAMPP, 390
 Linuks, 391
 Windows, 390, 393, 395
 XHR, 281
 XHTML, 116

XHTML Mobile Profile, 114, 115, 117, 134
 accesskey, atrybut, 117
 tbody, znacznik, 124
 thead, znacznik, 124
 zalety, 116
 XHTML MP, *Patrz* XHTML Mobile Profile
 xmlhttp_make_phone_call_string, 176, 177
 xmlhttp_make_phone_string, 178
 XHTML-Basic, 119, 134
 tbody, znacznik, 124
 thead, znacznik, 124
 XML, 282
 XMLHttpRequest, 281
 XUI, 227

Y

Youtube, 35

Z

zapytania o media, 12, 13, 14, 37, 41
 @import, 37
 @media, 12
 Internet Explorer, 62
 Mobile First, 61
 zdjęcia, robienie, 345
 Zepto.js, 227
 znaczniki
 <a>, 119
 , 68, 69
 <meta>, 22, 72, 73, 89
 <tbody>, 124
 <thead>, 124
 znaczniki semantyczne, 57
 Zwierzętom na pomoc, strona, 92, 93
 skrypt przekierowujący, 108
 wymagania, 94

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**

Zdobądź nowych użytkowników smartfonów!

Mobile Web. Rusz głową!

Liczba użytkowników internetu przeglądających strony internetowe za pomocą smartfonów rośnie lawinowo. Powszechnie sądzi się, że wkrótce będzie ich więcej niż tych, którzy korzystają z tradycyjnych komputerów. Dlatego już dziś należy przygotować się na rewolucję i zmienić podejście do tworzenia stron internetowych. Skorzystaj z nowości HTML5 oraz CSS3 i przekonaj się, że to wcale nie musi być trudne.

Jeżeli dołożysz do tego kolejną książkę z serii *Rusz głową!*, całe zagadnienie może okazać się wręcz banalne. Dzięki nowatorskim technikom nauczania będziesz chłonał wiedzę niczym gąbka. W trakcie lektury nauczysz się korzystać z podejścia *Responsive Web Design* oraz sprawdzisz, jak rozpoznać, że użytkownik korzysta ze smartfona. Ponadto poznasz biblioteki, które wspomogą Cię w trakcie realizacji postawionych zadań — jQuery Mobile to jedna z nich, a jej możliwości są oszałamiające. Twoją ciekawość powinien wzbudzić też projekt PhoneGap. Dzięki niemu będziesz mógł skonwertować swoją aplikację napisaną przy użyciu HTML5 do natywnego formatu, a to da Ci parę nowych możliwości. Książka ta jest obowiązkową pozycją dla wszystkich projektantów stron internetowych, którzy chcą być na czasie! Zaliczasz się do nich?

Wykorzystaj potencjał HTML5 i CSS3!

Poznaj:

▼ przyszłość internetu i stron WWW

▼ podejście *Responsive Web Design*

▼ możliwości biblioteki jQuery i projektu PhoneGap

▼ rozwiązania typowych problemów

helion.pl
księgarnia internetowa



Helion

Nr katalogowy: 11711

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>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-4864-1



Cena 79,00 zł

Informatyka w najlepszym wydaniu

9 788324 648641

