Spis treści

Przedmowa

Progresywne aplikacje webowe to ekscytująca, nowatorska forma aplikacji webowych. Wykorzystują najnowsze możliwości webowe celu dostarczania środowiska łączącego cechy natywnych aplikacji mobilnych z zaletami witryn WWW.

Ta książka pomoże zdobyć głęboką wiedzę i praktyczne umiejętności dotyczące programowania nowoczesnych, progresywnych aplikacji webowych na podstawie praktycznych przykładów.

Można się z niej nauczyć, jak budować aplikacje webowe, które korzystają z funkcjonalności stanowiących do tej pory wyłączną domenę aplikacji natywnych. Pozwolą one kierować do użytkowników powiadomienia z serwera, zdobyć miejsce na ekranie startowym, znacząco przyspieszyć działanie witryny i zrealizować w pełni funkcjonalną aplikację, która działa niezależnie od połączenia sieciowego.

W książce zostało przyjęte praktyczne podejście z przykładami ze świata rzeczywistego. Użyjemy istniejącej witryny WWW i rozdział po rozdziale przekształcimy ją w nowoczesną, progresywną aplikację webową.

Dla kogo jest ta książka

Niniejsza książka jest przeznaczona przede wszystkim dla programistów. Docelowa grupa czytelników to osoby, które chcą podnieść swoje istniejące umiejętności programowania witryn WWW i nauczyć się budować nowoczesne, progresywne aplikacje webowe.

Przyjęliśmy założenie znajomości co najmniej podstaw programowania witryn WWW przy użyciu języków HTML i JavaScript, ale znajomość względnie nowszych dodatków do języka JavaScript, takich jak ECMAScript 2015, obiekty promise lub funkcje asynchroniczne ECMAScript 2017 nie jest konieczna. Osoby znające już te nowoczesne konstrukcje językowe mogą pominąć (lub szybko przejrzeć) uwagi, które je wyjaśniają.

Osobom pracującym na stanowiskach niezwiązanych z techniką ta książka może pomóc poznać i ogólnie zrozumieć możliwości nowoczesnych, progresywnych aplikacji webowych. Wiele rozdziałów zawiera studia przypadków zebrane w wywiadach przeprowadzonych z zespołami stojącymi za niektórymi najbardziej wpływowymi witrynami na świecie, takimi jak Twitter, The Washington Post, Housing.com i Lyft. Zrozumienie dzisiejszych możliwości pomoże bardziej efektywnie pracować wszystkim osobom wybierającym między aplikacją natywną a webową, bez względu na to, czy są kierownikami, projektantami, menedżerami produktu, czy innymi osobami decyzyjnymi.

Zawartość książki

W niniejszej książce weźmiemy prostą witrynę WWW fikcyjnego hotelu Gotham Imperial i rozbudujemy ją skryptami service worker, aby ładowała się niemal natychmiast (nawet przy wolniejszych połączeniach), a użytkownicy mieli dostęp do wszystkich funkcji, nawet w trybie offline (w tym przeglądania rezerwacji, a nawet dokonywania nowych). Dowiemy się, jak umożliwić użytkownikom dodanie ikony w celu uruchamiania progresywnej aplikacji webowej z ekranu startowego telefonu. W końcu, aby w pełni uzyskać środowisko przypominające aplikację natywną, dodamy powiadomienia z serwera, pozwalające na informowanie i ponowne przyciąganie użytkowników nawet po opuszczeniu przez nich witryny.

Ponadto zagłębimy się w niektóre ważne zagadnienia związane z programowaniem progresywnych aplikacji webowych. Skupimy się na praktycznych aspektach tych koncepcji w sposób prowadzący do bardziej skutecznego programowania. Między innymi przyjrzymy się przydatnym narzędziom programistycznym, zagadnieniom bezpieczeństwa i zrozumieniu cyklu życia mechanizmu service worker.

Chociaż większość treści skupia się na praktycznej nauce, w dwóch rozdziałach (5 i 11) skupimy się szczególnie nowych możliwościach oferowanych przez progresywne aplikacje webowe, myśląc o nich jako o czymś więcej niż tylko nowym zbiorze trików do zastosowania w aplikacjach.

W rozdziale 5 zbadamy filozofię aplikacji webowych działających przede wszystkim w trybie offline, podejściu do budowania nowoczesnych aplikacji webowych, które traktują utratę łączności nie jako błąd, ale jako ewentualność, którą możemy zaplanować i obsłużyć z gracją.

W rozdziale 11 zbadamy nowe wyzwania interfejsu użytkownika i możliwości prezentowane przez progresywne aplikacje webowe. Przełomowe, progresywne aplikacje webowe wykraczają poza zwykłe oczekiwania użytkowników wobec sieci WWW. Niektóre z tych wyzwań obejmują wzmocnienie zaufania użytkowników, że ich dane nie zostaną utracone, gdy są w trybie offline, informowanie ich, że chociaż w tym trybie widoczna zawartość może być parę godzin starsza, mogą zaufać aplikacji, że wyśle powiadomienie, gdy zmieni się coś ważnego. Przy odpowiedniej obsłudze może to oferować wspaniałe możliwości, takie jak zwiększenie zaufania użytkowników do aplikacji, zwiększenie liczby powrotów i zyskanie trwałego miejsca w ich telefonach.

Książka kończy się spojrzeniem na parę nadchodzących technologii i interfejsów API przeglądarki, które pozwolą na dalszy rozwój progresywnych aplikacji webowych.

Konwencje użyte w tej książce

W książce tej wykorzystywane są następujące konwencje typograficzne:

Kursywa
: Wskazuje nowe terminy, adresy URL, adresy e-mail, nazwy plików oraz rozszerzenia plików.

Pogrubienie
: Wskazuje nazwy pakietów R.

`Stała szerokość`
: Wykorzystywana w listingach programów, a także w obrębie akapitów przy odwoływaniu się do elementów programu, takich jak nazwy zmiennych lub funkcji, bazy danych, typy danych, zmienne środowiskowe, instrukcje i słowa kluczowe.

`Pogrubiona stała szerokość`
: Wskazuje polecenia lub inny tekst, który powinien zostać wprowadzony przez użytkownika.

`Stała szerokość z kursywą`
: Wskazuje tekst, który powinien zostać zastąpiony przez wartość podaną przez użytkownika lub nakreśloną przez kontekst. Również tekst komentarza w przykładach kodu.

Ten element symbolizuje wskazówkę lub sugestię.

Ten element symbolizuje uwagę ogólną.

Ten element wyróżnia uwagę pozwalającą spojrzeć na ten sam problem z innej perspektywy.

Ten element wskazuje ostrzeżenie.

Używanie przykładów kodu

Materiał uzupełniający (przykłady kodu, ćwiczenia itp.) jest dostępny do pobrania ze strony *https://github.com/TalAter/gotham_imperial_hotel.*

Ta książka powstała, aby pomóc Czytelnikom wykonywać swoją pracę. W ogólności, jeśli jest podany przykład kodu, można używać go w swoich programach i dokumentacji. Nie trzeba kontaktować się z nami w celu uzyskania pozwolenia, o ile nie reprodukuje się znacznej części kodu. Na przykład napisanie programu, w którym wykorzystano kilka fragmentów kodu z tej książki, nie wymaga pozwolenia. Sprzedawanie lub rozpowszechnianie dysku CD-ROM z przykładami z książek wydawnictwa O'Reilly wymaga pozwolenia. Udzielenie odpowiedzi z cytatem i przykładem kodu z tej książki nie wymaga pozwolenia. Wprowadzenie znacznej ilości kodu przykładowego z tej książki do dokumentacji swojego produktu wymaga pozwolenia.

Doceniamy podanie informacji o naszym wkładzie, ale nie jest to wymagane. Informacje takie zwykle obejmują tytuł, autora, wydawcę i numer ISBN. Na przykład: „Progresywne aplikacje webowe, Tal Ater (O'Reilly). Copyright 2017 Tal Ater, 978-149-196165-0".

Osoby, które chcą użyć przykładów kodu w sposób przekraczający ich zdaniem uczciwe zastosowanie podanych powyżej pozwoleń, mogą skontaktować się z autorem pod adresem *tal@talater.com*.

O'Reilly Safari

Safari (wcześniej Safari Books Online) to platforma szkoleniowa i instruktażowa oparta na członkostwie dla przedsiębiorstw, urzędów, instytucji edukacyjnych i osób indywidualnych.

Członkowie mają dostęp do tysięcy książek, filmów szkoleniowych, ścieżek nauki, interaktywnych samouczków i nadzorowanych list odtwarzania od ponad 250 wydawców, w tym O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett oraz Course Technology i innych.

Więcej informacji można znaleźć na stronie *http://oreilly.com/safari.*

Kontakt

Prosimy nadsyłać komentarze i pytania dotyczące tej książki do wydawcy:

O'Reilly Media, Inc.
1005 Gravenstein Highway North Sebastopol, CA 95472
800-998-9938 (USA i Kanada) 707-829-0515 (międzynarodowo lub lokalnie)
707-829-0104 (faks)

Istnieje strona WWW dotycząca tej książki, zawierająca erratę, przykłady i inne dodatkowe informacje. Dostęp do tej strony jest możliwy pod adresem *http://bit.ly/building-progressive-web-apps*.

Komentarze lub pytania techniczne dotyczące tej książki prosimy nadsyłać na adres e-mail *bookquestions@oreilly.com*.

Więcej informacji o naszych książkach, kursach, konferencjach i wiadomościach znajduje się w naszej witrynie WWW *http://www.oreilly.com*.

Znajdź nas na Facebooku: *http://facebook.com/oreilly*

Śledź nas na Twitterze: *http://twitter.com/oreillymedia*

Oglądaj nas na YouTube: *http://www.youtube.com/oreillymedia*

Podziękowania

Po pierwsze chcę podziękować Aleksowi Russellowi i Jakowi Archibaldowi. Kiedy wpadłem na ich wykład na Google I/O 2014, nie miałem nawet pojęcia, co mogą oznaczać zagadnienia opisane w tej książce. Ich wykład i pisanie tej książki, pomogło mi nauczyć się odrobinę więcej.

Chciałbym także podziękować niesamowitym osobom budującym progresywne aplikacje webowe w świecie rzeczywistym, szczególnie tym, którzy podzielili się swoimi doświadczeniami w tej książce. Są to Joey Marburger, Ritesh Kumar, Rahul Yadav, Nicolas Gallagher, Chris Nguyen i Jeremy Toeman. Dziękuję Wam.

Dziękuję zespołowi wydawnictwa O'Reilly, który umożliwił powstanie tej książki. Szczególnie następującym osobom: Ally MacDonald, Jeff Bleiel, Sonia Saruba, Colleen Cole, David Futato, Rebecca Demarest, Heather Scherer, Ellen Troutman, Amanda Kersey i Karen Montgomery, a także jej dudkowi.

Podziękowania kieruję także do zespołów firm Google, Opera, Mozilla i Microsoft, które pomogły zrealizować tę książkę i opisaną w niej technologię, a także utalentowanym programistom, którzy pomogli mi nadać jej sens. Jeffrey Posnick, Addy Osmani, Matt Gaunt i Paul Kinlan – dziękuję Wam.

Dziękuję Aleksowi Feyerke i zespołowi Hoodie za pionierską pracę i artykuł o używaniu najpierw trybu offline, co stanowiło inspirację dla większości treści tej książki.

Najtrudniejszą kwestią podczas pisania takiej książki jest niepewność pracy przez prawie rok bez zewnętrznych informacji zwrotnych. Chciałbym podziękować wszystkim osobom, które poświęciły czas, aby wysłać mi swoje opinie na temat tej książki w trakcie jej tworzenia. Są to James Stanley, Patrick Conant, Fabio Rotondo, Neville Franks i Florian Semrau. Dziękuję Wam.

W końcu chcę podziękować osobom, które pomogły mi upewnić się, że to, co napisałem, rzeczywiście ma sens, dzięki wykonaniu pełnej technicznej redakcji tej książki. Są to Andreas Bovens, Kenneth Rohde Christiansen, Patrick Kettner i Thomas Steiner. Dziękuję Wam.

Rozdział 1

Wprowadzenie do progresywnych aplikacji webowych

> Słowa są bladymi cieniami zapomnianych imion. Jako że imiona mają moc, słowa też ją mają. Słowa mogą rozpalić ogień w ludzkich umysłach. Mogą sprawić, że najtwardsze serca zapłaczą. Jest siedem słów, które sprawią, że ktoś cię pokocha. Istnieje dziesięć słów, które złamią wolę najsilniejszego mężczyzny. Ale słowo to nic innego jak wizerunek ognia. Imię jest ogniem samym w sobie.
>
> – Patrick Rothfuss, *Imię wiatru*, w przekładzie Jana Karłowskiego

Co kilka lat w sieci WWW następuje nagły zwrot. Moment, kiedy wiele oddzielnych technologii łączy się ze sobą i wychodzi na światło dzienne. Istniejące technologie mogą być obecne przez lata lub nowsze mogą właśnie zyskać obsługę w przeglądarkach. Jednak dla zewnętrznego obserwatora zdaje się to jednym, nagłym błyskiem, kiedy sieć WWW robi krok do przodu.

Widzieliśmy to w przypadku technologii *Ajax*, która jednego dnia eksplodowała jakby z niczego (chociaż wiele technologii leżących u jej podstaw, takich jak *XMLHttpRequest*, było obecnych przez lata) i zmieniła postrzeganie sieci WWW jako serii połączonych, głównie statycznych stron.

Sama technologia Ajax była częścią rewolucji *Web 2.0*, innej potężnej nazwy, która zdaje się wyskoczyła znikąd w roku 2004 i eksplodowała z dnia na dzień.

Parę lat później hasło *po pierwsze mobilność* wyjrzało na światło dziennie i zasygnalizowało zmianę w postrzeganiu programowania webowego. Dzięki nadaniu nazwy zbiorowi filozofii projektowych te słowa zyskały niezwykłą moc. To krótkie hasło sprawiło, że mogliśmy ogłosić, że dni użytkowników siedzących przed domowym komputerem z 20-calowym monitorem i kablem podłączonym do gniazdka ściennego minęły. Hasło to pozwoliło zrozumieć, że nadszedł czas na zmianę podejścia do projektowania webowego.

Takie momenty często pojawiają się nie wtedy, kiedy rodzi się technologia, ale kiedy zostaje nazwana. Nazwy mają taką moc. Pozwalają nam pojąć nowe idee. Pozwalają nam dyskutować o nowych koncepcjach. Przyciągają naszą uwagę do kipieli bulgocącej pod powierzchnią.

Podobna wielka zmiana odbywa się właśnie teraz. Na szczęście, ma nazwę[1].

Powrót sieci WWW

Progresywne aplikacje webowe to nowa odmiana aplikacji webowych, które łączą korzyści aplikacji natywnych z prostotą sieci WWW.

Progresywne aplikacje webowe zaczynają jako proste witryny WWW, ale w miarę angażowania się użytkownika stopniowo nabierają nowej mocy. Przekształcają się z witryny WWW w coś, co w znacznie większym stopniu przypomina tradycyjną aplikację natywną.

Wyobraź sobie, że budzisz się rano, bierzesz telefon i wchodzisz na witrynę lokalnej firmy kolejowej. Szybko sprawdzasz rozkład jazdy pociągu, którym jeździsz do pracy, zamykasz przeglądarkę i wkładasz telefon z powrotem do kieszeni. Wieczorem odwiedzasz witrynę ponownie i sprawdzasz następne odjazdy kolei (nie zauważając nawet, że w windzie, którą jedziesz, nie ma zasięgu sieci komórkowej, ponieważ witryna firmy kolejowej działa teraz nawet, gdy jesteś w trybie offline). Następnego dnia odwiedzasz znowu witrynę, a przeglądarka wyświetla pytanie, czy chcesz dodać skrót do niej na ekranie startowym, a Ty z radością się zgadzasz. Następnego dnia, gdy uruchamiasz witrynę z ikony na ekranie startowym, witryna powiadamia, że z powodu prac remontowych mogą być opóźnienia i wyświetla pytanie, czy chcesz otrzymywać powiadomieniach o przyszłych zmianach w rozkładzie jazdy na tej trasie. Następnego ranka, gdy budzisz się, otrzymujesz powiadomienie na telefon, że pociąg ma 15-minutowe opóźnienie. Naciskasz przycisk drzemki na budziku i zyskujesz parę dodatkowych, cennych chwil snu.

To, co zaczęło działać jako prosta witryna, powoli zyskało nową siłę, aż uzyskało możliwości podobne do aplikacji natywnych w telefonie. Zamiast próbować odsyłać Cię do sklepu z aplikacjami, z nadzieją, że zainstalujesz ich aplikację, firma transportowa *zdobyła* trwałe miejsce w Twoim telefonie – krok po kroku.

Ten nowy model progresywny zastępuje binarną naturę *zainstalowano/nie zainstalowano* aplikacji natywnych. Progresywne aplikacje webowe budują zaufanie użytkowników i zdobywają nowe możliwości, gdy są one potrzebne.

Możemy zastanawiać się, na czym polega przewaga tego rozwiązania nad aplikacjami natywnymi? Dlaczego nie wystarczy poprzestać na tym, co działa? Dlatego, że jak już wiemy, obecne rozwiązanie sprawdza się jedynie w przypadku nielicznych szczęściarzy. Każdego roku szanse zainstalowania nowej aplikacji są coraz mniejsze. Każdego roku koszt zdobycia nowych użytkowników znacznie rośnie. Każdego roku utrzymanie zaangażowania użytkowników jest coraz trudniejsze.

1 A raczej, na szczęście, Alex Russel i Frances Berriman zjedli pewnego wieczoru razem kolację i wymyślili nazwę. Zobacz wpis na blogu Aleksa Russela pod *„Progressive Web Apps: Escaping Tabs Without Losing Our Soul"* (Progresywne aplikacje webowe: ucieczka przed kartami bez utraty ducha) – *https://pwabook.com/pwasmoment*.

Obecny krajobraz aplikacji mobilnych

Kiedy w roku 2007 powstał pierwszy iPhone, jego zabójcze funkcje pozwalały przeglądać witryny WWW w telefonie. Kiedy w następnym roku zostały wprowadzone aplikacje mobilne, programiści byli w końcu w stanie przekroczyć ograniczoną funkcjonalność strony WWW (przyjmując wiele nowych ograniczeń wynikających z wprowadzenia sklepu App Store).

Dzięki takim funkcjom, jak zaawansowana grafika, geolokacja, powiadomienia z serwera, dostępność w trybie offline, ikony na ekranie startowym itp. sieć WWW wydawała się blednąć w porównaniu z aplikacjami natywnymi w oczach wielu programistów. Aplikacje natywne szturmem zdobyły świat (i nasze telefony).

Ale ten trend się zmienia. Chociaż spędzamy więcej czasu niż kiedykolwiek wcześniej, korzystając z telefonów i aplikacji mobilnych, korzystamy z coraz mniejszej liczby aplikacji. Użytkownicy instalują mniej aplikacji i używają jedynie garści z tych, które zainstalowali. Jeśli nasza aplikacja jest jedną z 10 najpopularniejszych w sklepie z aplikacjami, prawdopodobnie możemy być zadowoleni. Ale próba podbicia rynku z nową aplikacją jest prawie niemożliwa, nie wspominając o kosztach.

Jak używamy aplikacji mobilnych

Zgodnie z raportem *2016 comScore*[2] przeciętna osoba przez 84% czasu korzystania z urządzeń przenośnych używa zaledwie 5 najpopularniejszych aplikacji. Przykro mi, ale to nie są Twoje aplikacje. W przypadku tabletów ta liczba jest jeszcze wyższa – użytkownicy spędzają 95% czasu, używając 5 najpopularniejszych aplikacji.

W tym samym raporcie zaprezentowano także liczby pokazujące, że znacznie łatwiej dotrzeć do dużej publiczności za pomocą witryny mobilnej niż aplikacji natywnej. Istnieje blisko 600 mobilnych witryn WWW, które docierają do ponad 5 milionów odwiedzających – prawie 4,5 razy więcej niż aplikacji natywnych o podobnych liczbach odbiorców. Tysiąc najpopularniejszych mobilnych witryn WWW ma publiczność prawie 3-krotnie większą niż tysiąc najpopularniejszych aplikacji, a *ich publiczność rośnie dwukrotnie szybciej niż publiczność aplikacji natywnych*.

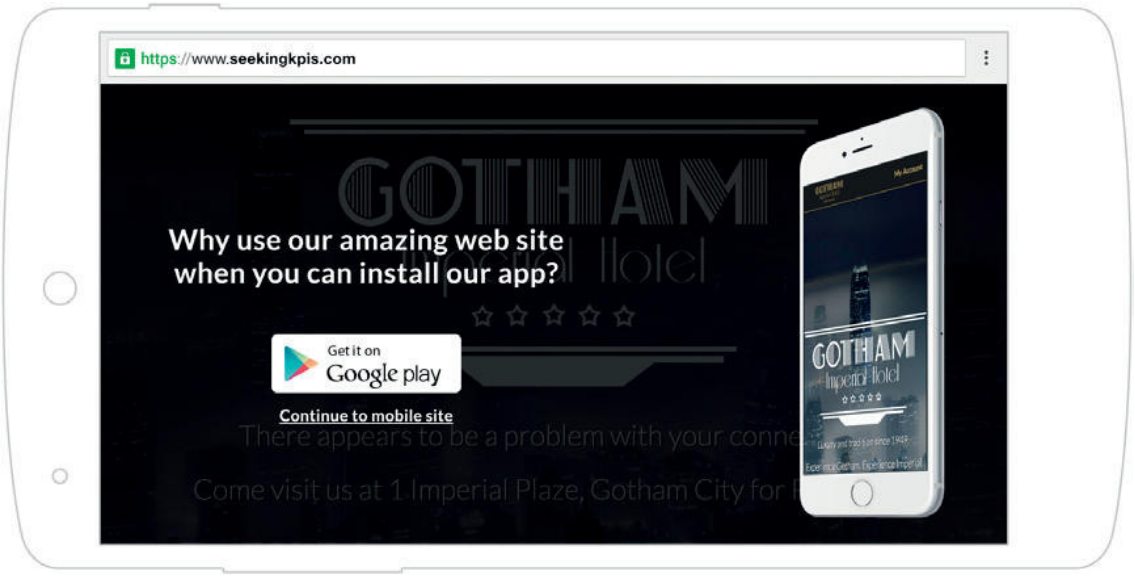
Zachęcanie użytkowników do instalacji i używania aplikacji przypomina przesiewanie przez sito. Użytkownicy muszą dowiedzieć się o niej (dzięki tradycyjnym reklamom online lub w witrynie WWW), a następnie odwiedzić naszą stronę w sklepie z aplikacjami. Później mają kliknąć w celu rozpoczęcia instalacji i zgodzić się na udzielenie aplikacji różnych uprawnień. Następnie powinni poczekać na pobranie i zainstalowanie aplikacji. W końcu muszą chociaż raz uruchomić aplikację, a nawet jej użyć.

2 *https://pwabook.com/comscore*

To sito może nie wydawać się tak złe, kiedy instalujemy aplikację, którą znamy i lubimy, taką jak Twitter lub Facebook. Ale wiele badań pokazało, że przeciętnie *20% użytkowników odpada na każdym etapie tego przesiewania*. Nie jest niczym niezwykłym dla programistów aplikacji, aby płacić za kliknięcia banera, tylko po to, aby odkryć, że mniej niż 20% tych użytkowników rzeczywiście uruchomiło aplikację.

Powstają witryny, które tak desperacko próbują zachęcić użytkowników do instalacji ich aplikacji, że uciekają się do nowej metody reklamy. Znamy to: odwiedzamy witrynę próbując przeczytać krótki artykuł lub poznać prognozę pogody na jutro. Ta informacja jest tuż, tuż, w zasięgu ręki, ale wtedy wyskakuje na ekranie baner blikujący dostęp do żądanej zawartości. Zawiera pytanie, czy chcemy zainstalować aplikację, zamiast przeczytać zawartość, która jest już przed naszymi oczami.

Niektóre osoby nazywają to *pełnostronicową reklamą wewnętrzną*. Preferuję krótszą nazwę: *trzaśnięcie drzwiami* (rysunek 1-1). Więcej szczegółów dotyczących skutków i nieskuteczności pełnostronicowych reklam wewnętrznych znajduje się w dodatku B.



Rysunek 1-1 *Typowe trzaśnięcie drzwiami*

To jest brutalna, kosztowna walka, aby zachęcić użytkownika do instalacji aplikacji w telefonie. Ale przewaga aplikacji natywnych nad witrynami WWW sprawia, że programiści aplikacji chcą maksymalnie wykorzystać (i wymusić) instalację aplikacji.

Czas życia aplikacji natywnej, w przeciwieństwie do tradycyjnej witryny WWW, wykracza poza krótki okres, który użytkownik spędza, korzystając z aplikacji od jej odkrycia do opuszczenia (czasem zaledwie kilka sekund później). Po zainstalowaniu aplikacje zyskują trwałe miejsce na ekranie startowym. Mogą docierać do użytkownika z powiadomieniami w dowolnym czasie i przypominać o swoim istnieniu. Dają ich programistom szanse na podjęcie próby uzyskania zwrotu z inwestycji w dłuższym czasie życia ich aplikacji.

Jednak wraz z wprowadzeniem progresywnych aplikacji webowych ta sytuacja wreszcie się zmienia. Przewaga wynikająca z niezwykłych możliwości, które do tej pory były wyłączną domeną aplikacji natywnych, jest teraz dostępna dla aplikacji webowych. Po połączeniu tego z bezproblemowym, jednoetapowym sitem sieci WWW (klikanie łącza w przeciwieństwie do instalowania aplikacji) zobaczymy, dlaczego użytkownicy, programiści webowi i firmy mogą wiele zyskać dzięki stosowaniu progresywnych aplikacji webowych.

Zalety progresywnych aplikacji webowych

Wraz z wprowadzeniem dodatkowych, niezwykłych możliwości progresywne aplikacje webowe spełniają wiele oczekiwań, jakie mamy względem aplikacji natywnych.

Oto kilka zalet, które zostały opisane w tej książce:

Dostępność bez względu na połączenie
: Progresywne aplikacje webowe nie zależą od połączenia użytkownika w taki sposób, w jaki zależą tradycyjne witryny WWW. Kiedy użytkownik odwiedza progresywną aplikację webową, rejestrowany jest skrypt service worker (zobacz „Karta, WWW i service worker" na stronie 7), który może wykrywać zmiany połączenia użytkownika i reagować na nie. Może zapewniać w pełni funkcjonalne środowisko użytkownika dla użytkowników w trybach offline, online, a także przy niestabilnym połączeniu.

  Użytkownicy mogliby używać progresywnych aplikacji webowych podczas lotu nad Atlantykiem, a nawet podejmować czynności (np. wysyłać wiadomości, potwierdzać udział w wydarzeniach lub komentować wpisy), wiedząc, że te czynności zostaną ukończone, kiedy tylko zostanie przywrócony tryb online – nawet po zamknięciu aplikacji i przeglądarki. Zobacz rozdział 7, aby poznać więcej szczegółów.

  Progresywne aplikacje webowe wprowadzają nowy poziom wiarygodności, a następnie zwiększają zaufanie użytkownika do poziomu, na jaki do tej pory zasługiwały tylko aplikacje natywne. Użytkownik wie, że może otworzyć aplikację WhatsApp w dowolnym czasie, napisać szybką wiadomość i zamknąć telefon bez martwienia się o bieżący stan połączenia. Do tej pory sieć WWW nie cieszyła się tym poziomem zaufania, co jest jednym z powodów preferowania aplikacji natywnych przez użytkowników.

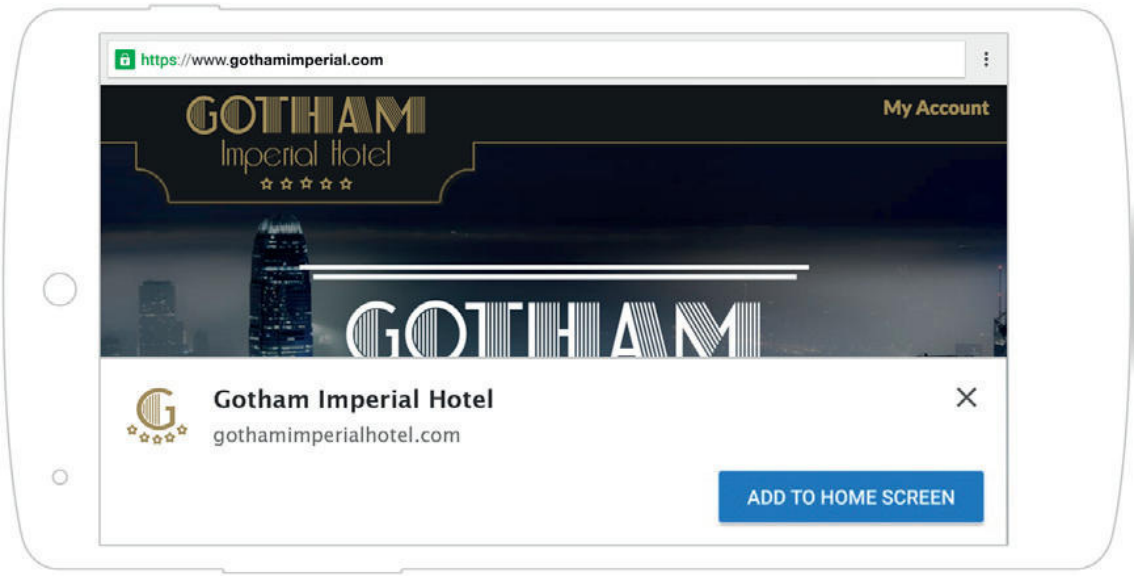
Szybkie czasy ładowania
: Używając skryptów service worker, możemy tworzyć witryny, które uruchamiają się błyskawicznie bez względu na to, czy użytkownik ma niesamowicie szybkie połączenie, zawodne połączenie 2G, a nawet nie ma żadnego połączenia. Witryny mogą ładować się w czasie milisekund, znacznie szybciej niż cokolwiek, czego doświadczaliśmy wcześniej w sieci WWW, a często nawet szybciej niż aplikacje natywne. W rozdziale 5 nauczymy się, jak to osiągnąć, i zagłębimy się w filozofię działania najpierw w trybie offlline.

Powiadomienia z serwera

Progresywne aplikacje webowe mogą wysyłać powiadomienia do swoich użytkowników (nawet dni po opuszczeniu przez nich witryny). Te powiadomienia oferują wspaniałą szansę ponownego angażowania użytkowników i przypominania im, aby wrócili do aplikacji. Powiadomienia w progresywnych aplikacjach webowych działają i wyglądają dokładnie tak, jak powiadomienia aplikacji natywnych, i nie są od nich odróżnialne. Zobacz rozdział 10, aby dowiedzieć się więcej o powiadomieniach z serwera.

Skrót na ekranie startowym

Kiedy użytkownik okazuje zainteresowanie progresywną aplikacją webową, przeglądarka automatyczne sugeruje, że może dodać skrót do jego ekranu startowego – zupełnie nieodróżnialny od aplikacji natywnej (rysunek 1-2). Zobacz rozdział 9, aby nauczyć się jak zagarnąć miejsce na ekranie startowym użytkownika.



Rysunek 1-2 *Natywny wygląd banera instalacji aplikacji*

Progresywne aplikacje webowe uruchamiane z ekranu startowego mogą mieć całkowicie natywny, przypominający aplikacje wygląd. Mogą mieć ekran powitalny przy ładowaniu. Mogą uruchamiać się w trybie pełnoekranowym, bez widocznego interfejsu przeglądarki lub telefonu. Mogą nawet zablokować się w konkretnej orientacji ekranu (ważne wymaganie dla gier). Szczegóły przedstawimy w rozdziale 9.

Lyft – więcej platform, więcej pasażerów

Oprócz zalet dla korzystających z nich użytkowników, progresywne aplikacje webowe oferują dodatkowe korzyści dla firm, które przyjmują to rozwiązanie.

Lyft, popularna usługa transportu samochodowego, jest firmą, która całkowicie polega na przychodach z aplikacji mobilnych.

Ponieważ część bieżących wysiłków firmy Lyft dotyczy pozyskania większej liczby użytkowników, firma znalazła się w konieczności obsługi rosnącej liczby urządzeń i mobilnych systemów operacyjnych. W miarę ewoluowania aplikacji firma Lyft musiała uznać obsługę starszych wersji systemów iOS i Android jako przestarzałą lub stanąć przed rosnącymi kosztami utrzymania. Zamiast zrezygnować z tych potencjalnych klientów (którzy stanowią 8% użytkowników systemu iOS i 3% użytkowników systemu Android), firma Lyft zbudowała progresywną aplikację webową.

Dzięki zastosowaniu progresywnych aplikacji webowych zespół firmy Lyft mógł zmniejszyć koszty techniczne i operacyjne związane z obsługą wielu aplikacji i urządzeń. Co ważniejsze, udało się dotrzeć do nowych użytkowników systemów iOS i Android, a także wcześniej ignorowanej publiczności: użytkowników systemów Windows Mobile i Amazon Fire.

Karta, WWW i service worker

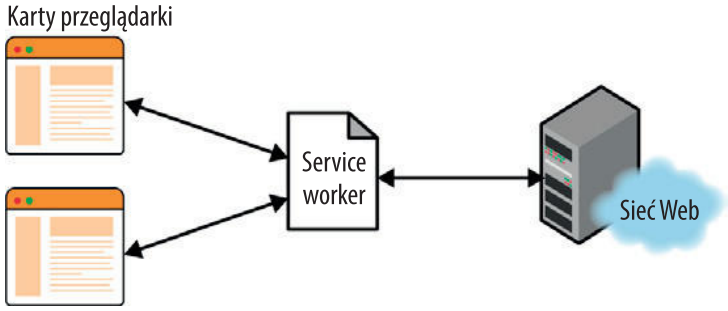
Sercem każdej progresywnej aplikacji webowej jest skrypt service worker.

Skrypty service worker prezentują zmianę w sposobie postrzegania programowania webowego. Warto poświęcić kilka minut na zrozumienie, gdzie odbywa się ich działanie, aby poznać ich potencjalne możliwości.

Przed technologią service worker mieliśmy kod albo po stronie serwera, albo w oknie przeglądarki. Skrypty service worker wprowadzają nową warstwę.

Service worker to skrypt, który można zarejestrować w celu kontrolowania jednej lub wielu stron witryny. Raz zainstalowany service worker znajduje się poza jakimikolwiek oknami czy kartami przeglądarki.

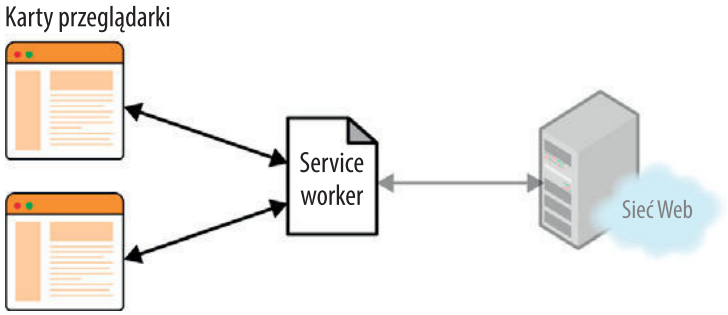
Z tego miejsca service worker może nasłuchiwać zdarzeń z wszystkich kontrolowanych przez siebie stron i odpowiednio na nie reagować. Zdarzenia, np. żądania plików z sieci WWW, mogą być przechwytywane, modyfikowane, rozprowadzanie i zwracane do strony (rysunek 1-3).



Rysunek 1-3 *Karta, WWW i service worker*

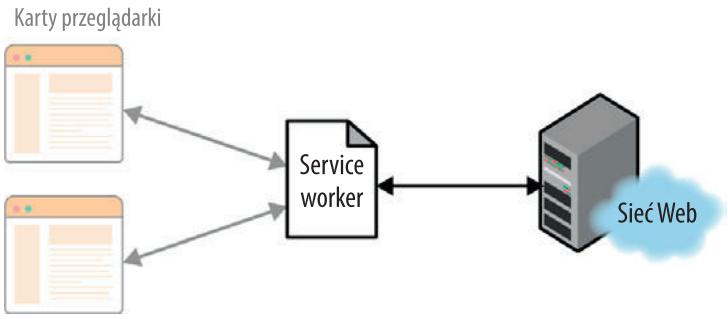
Oznacza to, że istnieje warstwa między stroną a siecią WWW, która może odpowiadać na żądania niezależnie od połączenia sieciowego. Warstwa, która działa nawet wtedy,

gdy użytkownik jest w trybie offline. Ta warstwa może wykrywać stan offline lub wolne odpowiedzi z serwera i zwracać zamiast tego zbuforowaną zawartość (rysunek 1-4).



Rysunek 1-4 *Strony komunikujące się ze skryptem service worker, gdy użytkownik jest w trybie offline*

Podążając dalej za tą logiką, oznacza to, że nawet jeśli użytkownik zamknie wszystkie karty działającej aplikacji w swojej przeglądarce, nadal jest warstwa, która może komunikować się z serwerem (rysunek 1-5). Może odbierać i wyświetlać powiadomienia z serwera lub upewniać się, że dowolna akcja dokonana przez użytkownika zostanie dostarczona do serwera (nawet jeśli użytkownik wszedł do windy zaraz po podjęciu akcji, a następnie zamknął aplikację przed odzyskaniem łączności).



Rysunek 1-5 *Skrypt service worker komunikujący się z serwerem po opuszczeniu strony przez użytkownika*

Zobaczymy, dlaczego skrypty service worker są sercem każdej progresywnej aplikacji webowej. Ich trwała natura pozwala progresywnym aplikacjom webowym spełniać oczekiwania dotyczące działania aplikacji. Są brakującym ogniwem między tym, co mogły zrobić tylko aplikacje natywne, a tym, co mogą zrobić progresywne aplikacje webowe.

Jednak prawdopodobnie ich największą siłą jest to, że skrypty service worker są po prostu plikami w języku JavaScript. Są pisane tak, jak każdy inny kod JavaScript pisany od lat.

Korzyści ze zrozumienia skryptów service worker i skojarzonych z nimi technologii opisanych w tej książce dla programistów są ogromne. Pozwalają nam wykorzystać istniejącą wiedzę o technologiach sieci WWW, takich jak JavaScript, HTML i CSS, aby programować aplikacje webowe, które mogą naprawdę konkurować z natywnymi aplikacjami mobilnymi, a nawet je prześcignąć.

Rozdział 2

Pierwszy skrypt service worker

Konfigurowanie projektu przykładowego

W tej książce przyjęliśmy praktyczne podejście do nauki progresywnych aplikacji webowych.

Zaczynając od tego rozdziału, weźmiemy prostą aplikację webową fikcyjnego hotelu Gotham Imperial i będziemy ją udoskonalać rozdział po rozdziale. Każdy rozdział jest oparty na pracy wykonanej w poprzednim i stanowi jej rozwinięcie, a na koniec każdego rozdziału będziemy mieć gotową do wdrożenia, działającą aplikację webową.

Na końcu książki okaże się, że wzięliśmy tę prostą witrynę WWW i przekształciliśmy ją we w pełni funkcjonalną progresywną aplikację webową.

W celu prześledzenia przykładów kodu i samodzielnego ich modyfikowania można sklonować kod źródłowy aplikacji na komputer lokalny. Kod znajduje się w *repozytorium GitHub Gotham Imperial Hotel*[1].

Zauważ, że konieczna jest możliwość uruchomienia oprogramowania Git, Node.js i NPM na komputerze lokalnym w celu sklonowania kodu i uruchomienia go lokalnie. Jeśli uruchomienie któregoś z tych elementów nie jest możliwe, można przestudiować tę książkę bez nich (pobierając kod źródłowy bezpośrednio z repozytorium GitHub i uruchamiając go na zdalnym serwerze), ale tego nie polecam.

Na początku otwórz wiersz polecenia komputera (konsolę), zmień katalog na ten, do którego chcesz pobrać kod, a następnie uruchom następujące polecenia:

```
git clone -b ch02-start git@github.com:TalAter/gotham_imperial_hotel.git
cd gotham_imperial_hotel
npm install
```

Te polecenia powodują sklonowanie kodu źródłowego aplikacji webowej Gotham Imperial Hotel, zmianę na odgałęzienie o nazwie `ch02-start`, a następnie zainstalowanie zależności koniecznych do jej uruchomienia.

1 *https://pwabook.com/gihrepo*

Następnie możemy pójść dalej i uruchomić serwer lokalny, aby obsługiwał witrynę do przeglądarki za pomocą następującego polecenia:

```
npm start
```

Jeśli otworzymy teraz adres *http://localhost:8443/* w przeglądarce, powinniśmy zobaczyć aplikację webową Gotham Imperial Hotel.

Jeśli aplikacja webowa nie ładuje się do przeglądarki, należy upewnić się, że:

- Oprogramowanie Git, Node.js i NPM jest zainstalowane i można go używać z wiersza polecenia (np. Terminal lub iTerm w systemie macOS; Wiersz poleceń systemu Windows lub Cygwin w systemie Windows).
- Wszystkie poprzednie kroki zostały wykonane.

Jeśli nadal występują problemy z uruchomieniem aplikacji, można poprosić o pomoc w naszej witrynie śledzenia problemów[2].

Możemy teraz otworzyć projekt w ulubionym środowisku IDE lub edytorze, aby czytając książkę, przekształcać tę witrynę w progresywną aplikację webową.

Ponieważ kod z poszczególnych rozdziałów jest oparty na zmianach wprowadzonych w poprzednich rozdziałach, kod na początku każdego rozdziału wymaga uwzględnienia wszystkich tych zmian. W razie pominięcia pewnych ćwiczeń kodowania w książce, a nawet całych rozdziałów, możemy zawsze uzyskać kod w stanie, w którym powinien być na początku danego rozdziału. Służą do tego dwa następujące polecenia wprowadzane w wierszu polecenia:

```
git reset --hard
git checkout ch04-start
```

Te polecenia spowodują zresetowanie wszystkich zmian dokonanych lokalnie, a następnie pobranie odgałęzienia ze zmianami dokonanymi przed tym rozdziałem. Koniecznie należy zmienić nazwę odgałęzienia w drugim poleceniu na nazwę rozdziału, w którym obecnie jesteśmy. Na przykład, jeśli zaczynamy rozdział 6, uruchamiamy polecenie `git checkout ch06-start`, które pobiera odgałęzienie zawierające wszystkie zmiany dokonane w pierwszych pięciu rozdziałach.

Witamy w hotelu Gotham Imperial

Projekt, który będzie towarzyszyć naszej podróży w celu odkrywania progresywnych aplikacji webowych to witryna WWW fikcyjnego hotelu Gotham Imperial.

Ta prosta witryna zawiera dwie strony:

1. Stronę główną zawierającą informacje o hotelu, mapę, listę nadchodzących wydarzeń oraz formularz do dokonywania nowych rezerwacji.

2 *https://pwabook.com/gihissues*

2. Stronę My Account (moje konto) zawierającą listę rezerwacji użytkownika, nadchodzących wydarzeń oraz formularz do dokonywania nowych rezerwacji.

Te dwie strony, chociaż są proste, zawierają większość elementów, z których składają się zarówno strony zorientowane na zawartość, jak i aplikacje webowe bardziej przypominające aplikacje.

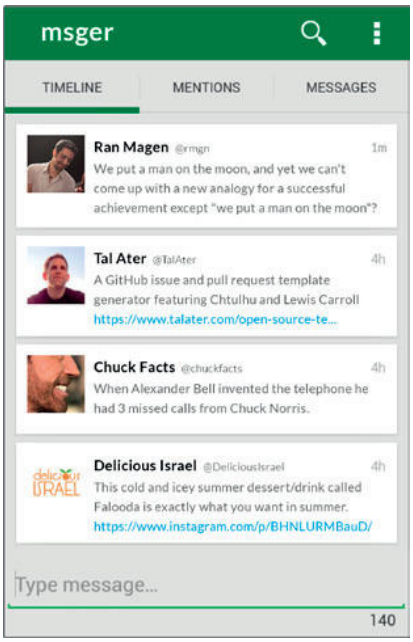
Podczas czytania tej książki weźmiemy tę prostą witrynę WWW i przekształcimy ją we w pełni funkcjonalną progresywną aplikację webową.

Różne wyzwania, różne podejścia

Poznając różne funkcje, które umożliwiają budowanie progresywnej aplikacji webowej, czasami oderwiemy się od aplikacji Gotham Imperial Hotel i zbadamy te same pomysły w innym kontekście.

Przykładowa aplikacja hotelu przypomina bardziej tradycyjną witrynę firmową, natomiast te uwagi pozwolą spojrzeć na podobne wyzwania z perspektywy aplikacji, która bliżej przypomina tradycyjną aplikację natywną. Poznając różnice i podobieństwa tych podejść, zyskamy lepsze zrozumienie sposobów dopasowania poszczególnych funkcji do różnych projektów, a także czerpania korzyści z poszczególnych nowych funkcji przez różne firmy.

msger, czyli fikcyjna aplikacja komunikatora, którą będziemy poznawać w tych uwagach, pozwala użytkownikom publikować 140-znakowe wiadomości i wyświetla strumień ostatnich wiadomości użytkownika. Gdy pojawia się nowa wiadomość, jest dodawana na górze strumienia wiadomości, przesuwając starsze wiadomości w dół (rysunek 2-1).



Rysunek 2-1 *Przykładowa aplikacja komunikatora*

Poznawanie kodu

Zanim zaczniemy, zapoznamy się z podstawową strukturą kodu aplikacji. Wewnątrz głównego katalogu projektu znajdują się dwa najważniejsze katalogi:

public
: Zawiera cały kod po stronie klienta witryny, a także inne pliki potrzebne do jego uruchomienia (np. obrazy i arkusze stylów).

server
: Zawiera kod po stronie serwera, który obsługuje witrynę, śledzi rezerwacje, wysyła powiadomienia itp.

Wszystkie przykłady kodu w książce będą obejmować katalog public, ale od czasu do czasu warto zajrzeć do katalogu server – szczególnie w rozdziale 10.

Najpierw słowo o kodzie

Jeśli patrzymy na początkowy stan kodu aplikacji, widzimy, że jest utrzymany dość prosto. Często idziemy na kompromis z najlepszymi praktykami, a nawet zwykłym rozsądkiem, aby osiągnąć czytelność i możliwość przejrzystej demonstracji kluczowych zasad, których będziemy się uczyć.

Do czasu ukończenia tej książki będziemy mieć szansę poprawienia większości tego kodu i mam nadzieję nauczenia się nie tylko, jak budować progresywne aplikacje webowe od podstaw, ale także jak można poprawić istniejący projekt, aby zmienić go w progresywną aplikację webową.

Zdecydowałem, aby nie używać wielu nowoczesnych konstrukcji języka ES2015, co pozwoli skupić się na temacie książki, a nie na nowej składni, która może być nieznana niektórym Czytelnikom. W celu poznania korzyści zastosowania standardu ES2015 do kodu z tej książki zobacz dodatek A.

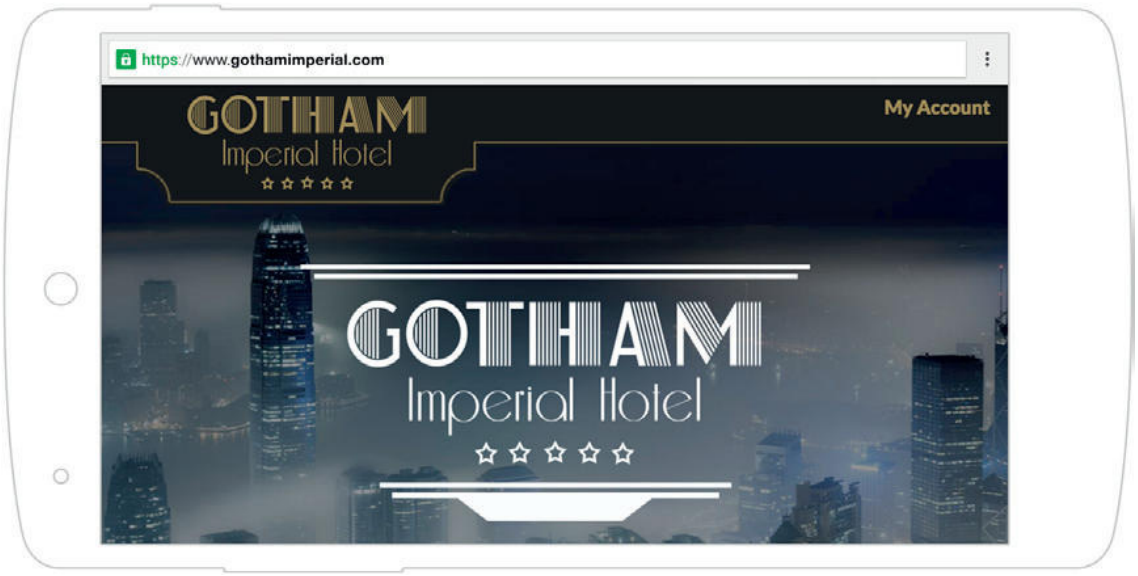
Obecne działanie w trybie offline

Po ukończeniu poprzedniej sekcji powinniśmy mieć teraz kopię aplikacji webowej Gotham Imperial Hotel oraz lokalny serwer WWW, na którym ta aplikacja może działać.

W celu upewnienia się, że pracujemy nad kodem w stanie potrzebnym na początku tego rozdziału, uruchom następujący wiersz polecenia:

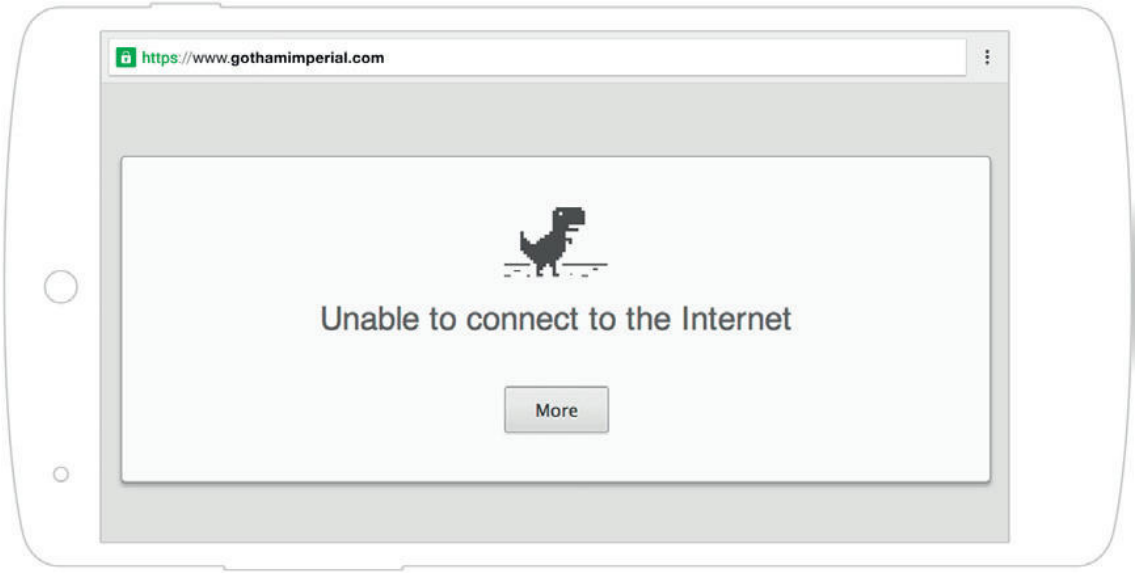
```
git reset --hard
git checkout ch02-start
```

Następnie uruchom polecenie **`npm start`**, aby uruchomić lokalny serwer WWW, na którym będzie działać ta witryna, i otwórz ją w przeglądarce (*http://localhost:8443/*). Powinna być widoczna witryna w całej okazałości (rysunek 2-2).



Rysunek 2-2 *Strona główna witryny Gotham Imperial Hotel*

To jest dzisiejsza sieć WWW. Bogata, piękna i przydatna. Rzeczywiście to jest dzisiejsza sieć WWW dla nas – programistów. Jako programiści zwykle patrzymy na witryny z względnie nowoczesnego komputera stacjonarnego, laptopa lub urządzenia przenośnego. Mamy niezawodne połączenie albo z serwerem lokalnym, albo z serwerem deweloperskim w bliskiej odległości. Natomiast nasi użytkownicy mogą postrzegać nasze aplikacje webowe znacznie inaczej niż my. Co dzieje się, gdy użytkownik odwiedza naszą aplikację webową, gdy jest w trybie offline (rysunek 2-3)?



Rysunek 2-3 *Nasza przykładowa aplikacja webowa odwiedzana przez użytkownika w windzie*

Niestety, dla wielu użytkowników tak właśnie wygląda dzisiejsza sieć WWW. Skrypty service worker w końcu pozwalają nam coś z tym zrobić.

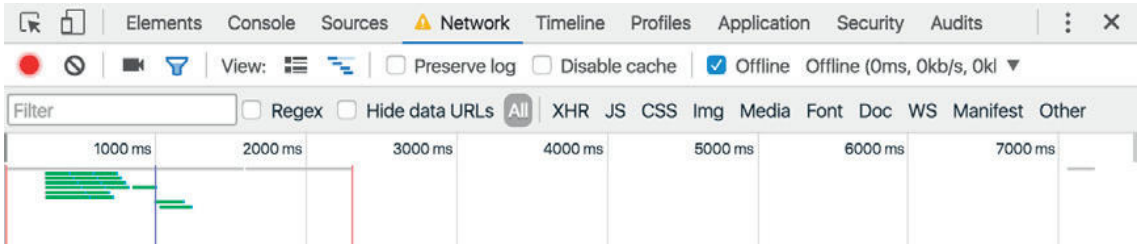
Symulowanie stanu offline

Podczas pracy nad przykładową aplikacją w tej książce często będzie potrzebne symulowanie stanu offline. Ponieważ stan offline w istocie oznacza, że użytkownik nie może dosięgnąć serwera, jednym ze sposobów symulowania tego stanu jest wyłączenie serwera projektowego.

W wierszu polecenia, gdzie działa lokalny serwer, naciśnij kombinację klawiszy Ctrl+C, aby zakończyć działanie serwera. Teraz ponowne załadowanie aplikacji w przeglądarce pozwoli zobaczyć jej działanie, gdy użytkownik jest w trybie offline.

Kiedy jesteśmy gotowi, aby z powrotem „przejść do stanu online", wystarczy ponownie uruchomić `npm start`.

Ta podstawowa metoda działa dobrze w celu symulowania stanu offline podczas programowania. Jednak po wprowadzeniu kodu do środowiska produkcyjnego wyłączanie serwera produkcyjnego za każdym razem, gdy chcemy coś przetestować, nie jest praktyczne. Na szczęście, większość nowoczesnych przeglądarek zawiera narzędzia symulowania stanu offline, a nawet różnych prędkości połączenia (rysunek 2-4). Zobacz „Narzędzia dla programistów" na stronie 58, aby poznać więcej szczegółów.



Rysunek 2-4 *Symulowanie stanu offline w przeglądarce Google Chrome*

Tworzenie pierwszego skryptu service worker

Przejmiemy kontrolę nad doświadczeniem użytkownika w trybie offline.

Zaczynamy od zarejestrowania nowego skryptu service worker dla bieżącej strony. Otwórz plik *js/app.js* i dodaj następujący kod na górze pliku:

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/serviceworker.js")
    .then(function(registration) {
      console.log("Service Worker registered with scope:", registration.scope);
    }).catch(function(err) {
      console.log("Service worker registration failed:", err);
    });
}
```

Ten kod zaczyna się od weryfikacji obsługi skryptów service worker w bieżącej przeglądarce. Następnie rejestruje skrypt service worker za pomocą wywołania metody `navigator.serviceWorker.register`, która przyjmuje dwa argumenty. Pierwszym jest adres URL skryptu service worker. Drugi to opcjonalny obiekt opcji (który tu pominęliśmy, ale zbadamy go dalej w tym rozdziale w punkcie „Pojęcie zasięgu skryptu service worker" na stronie 25).

Dzięki testowaniu obsługi skryptu service worker przed jego użyciem, upewniamy się, że nie wykluczamy korzystania z naszej aplikacji przez użytkowników starszych przeglądarek, a jednocześnie oferujemy udoskonalone działanie dla użytkowników nowocześniejszych przeglądarek. Ta praktyka progresywnego udoskonalania jest ważną zasadą budowania aplikacji (zobacz „Co to jest progresywne udoskonalanie?" na stronie 18).

Wywołanie metody `register` zwraca obiekt *promise* (obietnica). Jeśli obietnica zostanie spełniona, co oznacza, że skrypt service worker został pomyślnie zarejestrowany, zostanie wywołana funkcja zdefiniowana w instrukcji `then`. W razie wystąpienia jakiegokolwiek problemu wykonywana jest funkcja zdefiniowana wewnątrz bloku catch.

Jeśli odświeżymy przykładową aplikację w przeglądarce, powinniśmy zobaczyć komunikat o błędzie w konsoli przeglądarki o treści „Service worker registration failed"[3] (Błąd rejestracji skryptu service worker).

Rejestracja skryptu service worker nie powiodła się, a obietnica obiektu promise została odrzucona, ponieważ nie utworzyliśmy jeszcze pliku *serviceworker.js*.

Utwórz pusty plik o nazwie serviceworker.js i umieść go w publicznym katalogu głównym projektu (tj. public/serviceworker.js). Po odświeżeniu przeglądarki powinniśmy otrzymać komunikat „Service worker registered with scope: *http://localhost*:8443/" (Zarejestrowano skrypt service worker o zasięgu: *http://localhost*:8443/). Chociaż ten service worker jest jedynie pustym plikiem, jest już prawidłowym skryptem i został zarejestrowany pomyślnie.

Możesz mieć pokusę, aby przenieść plik *serviceworker.js* do podkatalogu *js* projektu. Na razie zachowamy go jednak w katalogu głównym. Uzasadnienie poznamy w podrozdziale „Pojęcie zasięgu skryptu service worker" na stronie 25.

Zaczynamy badanie możliwości skryptu service worker. Dodaj następujący kod do pliku *serviceworker.js*:

```
self.addEventListener("fetch", function(event) {
  console.log("Fetch request for:", event.request.url);
});
```

3 Jeśli ten komunikat o błędzie nie jest widoczny, trzeba upewnić się, że działa serwer lokalny, i przeczytać o obsłudze przeglądarek w „Obsługa skryptów service worker w przeglądarce" na stronie 18.

Ten kod dodaje do skryptu service worker możliwość nasłuchiwania zdarzeń. W tym celu wywołuje metodę `addEventListener` na obiekcie `self` (`self` w skrypcie worker odwołuje się do samego skryptu service worker). Dodany obiekt odbiornika zdarzeń (*event listener*) będzie nasłuchiwać wszystkich zdarzeń fetch (pobierz) i przekazywać je do obiektu service worker oraz uruchamiać funkcję, którą definiujemy dalej, przekazując jej obiekt event (zdarzenie) jako pojedynczy argument. Funkcja zdefiniowana do obsługi tych zdarzeń uzyskuje dostęp do obiektu request (żądanie) dostępnego jako właściwość zdarzenia fetch i rejestruje adres URL tego żądania.

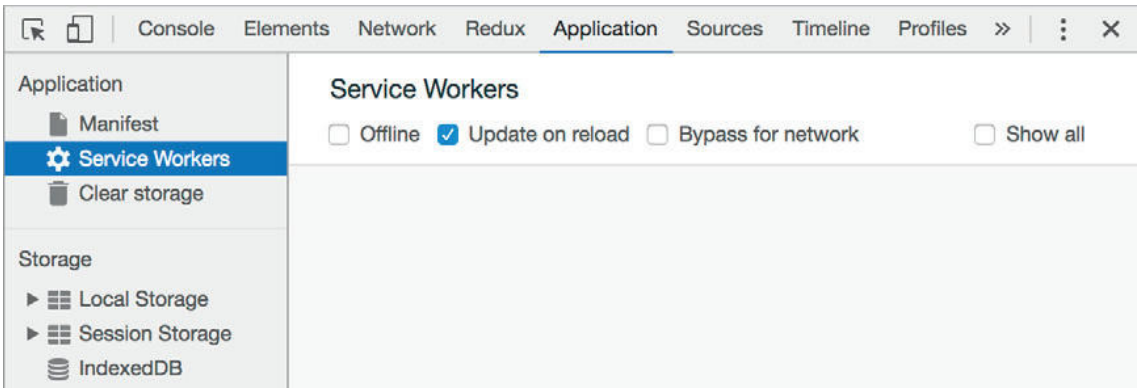
Po odświeżeniu strony powinniśmy teraz zobaczyć, że adres URL każdego żądania dokonywanego na stronie zostaje zarejestrowany w konsoli przeglądarki. (Jeśli nie widać żadnych adresów URL w konsoli, stary pusty skrypt service worker może nadal sprawować kontrolę; patrz ramka „Cykl życia skryptu service worker").

Cykl życia skryptu service worker

Być może zauważyliśmy, że zmiany dokonane w pliku service worker nie zostają zastosowane natychmiast po odświeżeniu przeglądarki. Jest tak, ponieważ stary skrypt service worker jest nadal aktywny, a nowy skrypt service worker pozostaje w stanie oczekiwania do czasu, aż stary skrypt przestanie kontrolować stronę.

Chociaż może to się wydawać strasznie niewygodnie, w rzeczywistości jest bardzo potężną cechą działania skryptów service worker. Zbadamy to bardziej szczegółowo w rozdziale 4.

Dla ułatwienia programowania możemy poinstruować przeglądarkę, aby nowe skrypty service worker przejmowały natychmiastową kontrolę nad stroną. W przeglądarce Chrome, aby to zrobić, możemy otworzyć kartę Application (aplikacja) narzędzi dla programistów i w sekcji Service Workers włączyć opcję „Update on reload" (aktualizuj po ponownym załadowaniu, rysunek 2-5). Dzięki temu będziemy mieć pewność, że przy każdej zmianie skryptu service worker, po odświeżeniu strony nowy skrypt service worker natychmiast przejmie kontrolę nad stroną.



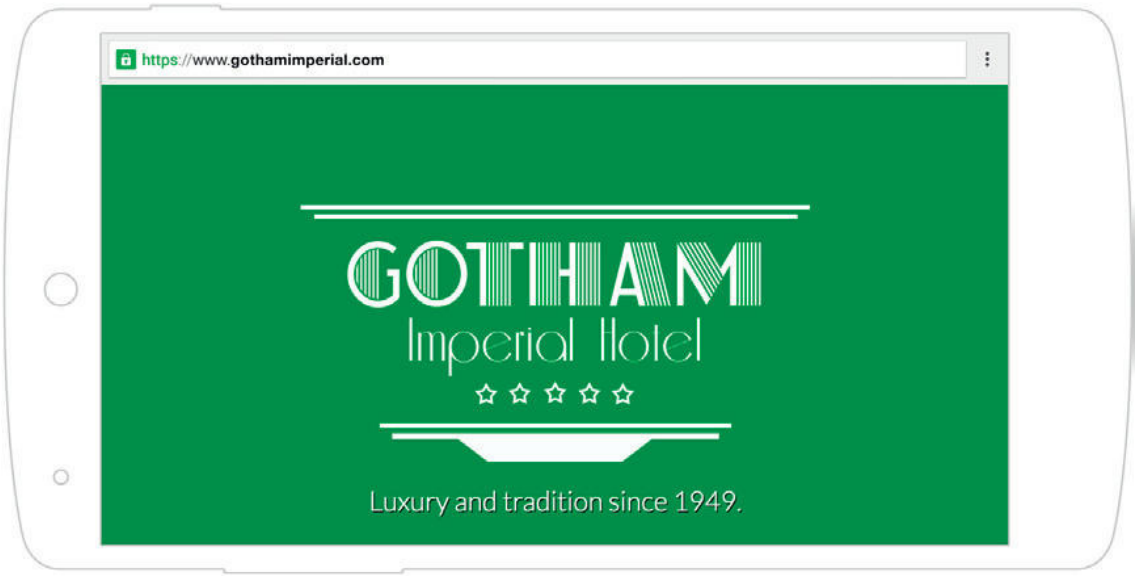
Rysunek 2-5 *Włączanie aktualizacji po ponownym załadowaniu*

Choć początkowo nie wydaje się to zbyt imponujące, warto się zastanowić: poszczególne żądania dokonywane przez naszą stronę (w tym żądania do serwerów innych firm) przechodzą teraz przez skrypt service worker. Wszystkie te żądania mogą być teraz przechwytywane, analizowane, a nawet modyfikowane. Zobaczmy na przykładzie, jak potężna jest ta funkcjonalność.

Zastąp kod w pliku *serviceworker.js* następującym i odśwież przeglądarkę:

```
self.addEventListener("fetch", function(event) {
  if (event.request.url.includes("bootstrap.min.css")) {
    event.respondWith(
      new Response(
        ".hotel-slogan {background: green!important;} nav {display:none}",
        { headers: { "Content-Type": "text/css" }}
      )
    );
  }
});
```

Ten kod nasłuchuje zdarzeń fetch i bada adres URL każdego żądania, aby zobaczyć, czy zawiera on ciąg znaków `bootstrap.min.css`. Jeśli tak jest, zamiast pobierać plik ze zdalnego serwera, skrypt service worker odpowie na żądania nowym obiektem `Response` (odpowiedź) utworzonym w locie i zawierającym niestandardowy arkusz CSS (rysunek 2-6).



Rysunek 2-6 *Nadpisywanie żądania arkusza CSS i modyfikowanie koloru tła*

W zaledwie kilku wierszach kodu JavaScript udało nam się utworzyć skrypt service worker, który przechwytuje żądanie do innego serwera, utworzyć nową odpowiedź w locie i zaprezentować ją przeglądarce jakby pochodziła z tego serwera. W istocie utworzyliśmy serwer proxy wewnątrz przeglądarki.

Obsługa skryptów service worker w przeglądarce

Od czasu opublikowania specyfikacji w roku 2014 przeglądarki zaskakująco szybko zaadaptowały technologię service worker. Pod koniec roku 2015 wszystkie przeglądarki Chrome, Opera, Firefox i Samsung Internet miały dodaną obsługę skryptów service worker.

W chwili publikacji tej książki zespół WebKit pracuje nad wprowadzeniem skryptów service worker do urządzeń iPhone i wszystkich przeglądarek typu Safari, podobnie zespół Microsoft Edge.

W celu poznania aktualnego stanu obsługi skryptów service worker i związanych z nimi technologii w przeglądarkach można odwiedzić stronę WWW Jake'a Archibalda *„Is Service- Worker Ready?”* [4]

Co to jest progresywne udoskonalanie?

Centrum filozofii naszej aplikacji, a także dowolnej nowoczesnej aplikacji webowej jest zasada *progresywnego udoskonalania*.

Progresywne udoskonalanie oznacza włączenie jak największej funkcjonalności, z której mogą skorzystać użytkownicy. Oznacza to programowanie witryn, które nie ulegają awarii tylko dlatego, że przeglądarka użytkownika nie obsługuje pewnej funkcjonalności.

Pomyślmy o progresywnym udoskonalaniu jako o sposobie budowania aplikacji webowej w stylu warstwowym. Zaczynamy od podstawowej zawartości, prostych łączy HTML, obrazów itp. Następnie dla użytkowników korzystających z obsługi JavaScript dodajemy warstwę wzbogacającą łącza, aby pobierać zawartość asynchronicznie i zastąpić statyczne obrazy map interaktywnymi Mapami Google. Dodajemy obsługę trybu offline dla przeglądarek z obsługą skryptów service worker. Wysyłamy powiadomienia z serwera do użytkowników, którzy mogą je odbierać.

Dzięki temu nie tylko możemy pokazać w pełni funkcjonalną aplikację wszystkim użytkownikom, ale także witryna staje się bardziej dostępna dla wszystkich publiczności (w tym użytkowników starszych przeglądarek lub telefonów) i pozwala mechanizmom wyszukiwarek na prawidłowe indeksowanie całej zawartości.

Podczas rejestracji skryptu service worker zaczęliśmy od weryfikacji obsługi przeglądarki. Użytkownicy korzystający z przeglądarek, które mają tę funkcjonalność, będą cieszyć się z poprawionego działania, a reszta użytkowników będzie nadal otrzymywać to, co przedtem. Progresywnie rozbudowujemy naszą aplikację, bez pozbywania się użytkowników.

Nie należy mylić terminu *progresywne udoskonalanie* z *progresywnymi aplikacjami webowymi*. Chociaż progresywne aplikacje webowe powinny być programowane z myślą o progresywnym udoskonalaniu, nie jest to wymaganie technicznie. Możemy zbudować

4 *https://pwabook.com/isswready*