Predgovor

Sećam se kako sam se upoznao sa Johnom Wargom. Govorio sam na konferenciji PhoneGap Day u Portlandu, Oregonu. Pogledom sam potražio Briana LeRouxa i video sam ga da razgovara sa nekim koga nisam prepoznao. Nagnuo sam se i pitao prijatelja: „Hej, ko je tip u odelu?“, na šta je prijatelj odgovorio: „Oh, to je John Wargo. On je analitičar.“ Odelo je izdvojilo Džona na ovom skupu zaluđenika, jer je većina prisutnih nosila majice sa smešnim sloganima. To je bilo pre vremena kada je Mark Zakerberg popularizovao izgled „progamera u duksericama sa kapuljačom“.

Ako se dobro sećam, na mojoj majici je bilo prase sa oblačićem na kome je pisalo: „Mmmm... slanina“, i dok se sećam toga vremena, to nije bilo čak ni odelo koje je John nosio – to je bio samo blejzer. Međutim, nije preterivanje reći da je bio najbolje obučen čovek na konferenciji, ali skrećem pažnju.

„Čoveće!“, pomislio sam, „Analitičar.“ Vidite, tek sam nedavno prešao iz industrije telekomunikacija na razvoj softvera. U telekomu, analitičari nemaju najveću reputaciju. Onda sam razgovarao sa Džonom. Nastavio je da mi postavlja pronicljiva pitanja o mom radu na PhoneGapu i zajednici Apache Cordova.

U ovom trenutku se odupirem želji da ne sudim o knjizi po naslovnoj strani, a nikada nisam bio dobar u tome da izbegnem da se našalim.

Džon je postao važan član i edukator zajednice Apache Cordova i napisao ne jednu, već četiri autoritativne knjige na tu temu, drugačiji od ove koju sada držite u svojim rukama o progresivnim veb aplikacijama.

Jedan od često citiranih principa PhoneGap/Cordova projekta bio je da na kraju „prestane da postoji“, ali drugi manje pominjani princip je bio da se „od veba napravi prvoklasna razvojna platforma“. Sada se to obećanje da će biti prvoklasna platforma ispunilo zahvaljući progresivnim veb aplikacijama. To što je bio veliki deo zajednice Apache Cordova tako dugo stavlja Johna u sjajnu poziciju, jer je bio gde treba da u svakom koraku vidi evoluciju mobilnog veba.

Ako jednog dana naletite na Džona na konferenciji ili na potpisivanju knjige, dođite sa preporukom gde da nabavite svežu, ukusnu krofnu. Ovo je jedna od dve zajedničke osobine koje imamo. Drugi je suva, sarkastična duhovitost.

Doneli ste odličnu odluku kada ste nabavili ovu knjigu. Da tek počinjem na svom putu učenja ka ovladavanju progresivnim veb aplikacijama, nema mnogo ljudi kojima bih verovao više nego Džonu da me odvedu na taj put. Želim da čujete njegov glas dok čitate ovu knjigu kao i ja.

—Simon MacDonald, zastpunik programera, Adobe
Novembar 2019.

Uvod

Kada prave aplikacije za stone računare, laptopove, pametne telefone i tablete, programeri generalno imaju dve opcije za korišćenje: izvorne (eng. nativ) aplikacije napravljene posebno za ciljnu platformu ili veb aplikacije koje se u potpunosti mogu izvršavati na skoro svakom sistemu zbog sloja apstrakcije koji pružaju veb čitači (eng. browsers). Izrada izvornih aplikacija za bilo koju ciljnu platformu je dugotrajan i skup pristup, posebno kada je vaša aplikacija za više tipova sistema (stoni računari, pametni telefoni, televizori, itd.).

Veb aplikacije su bile izazovne jer je korisničko iskustvo moglo dramatično da varira u zavisnosti od toga sa kog tipa sistema je korisnik pristupio aplikaciji. Čitači za desktop računare su moćni, ali čitači mobilnih uređaja imaju ograničenja zbog smanjene veličine ekrana, brzine procesora, propusnog opsega mreže i još mnogo toga. Mnoga od ovih ograničenja su nestala, ali i dalje postoji znatna razlika između mogućnosti izvorne aplikacije i veb aplikacije.

Veb programeri imaju na raspolaganju mnogo alata i tehnologija koje im pomažu da naprave bogate, privlačne aplikacije. Tokom godina, pojavile su se različite tehnologije kao što su Sun Microsistems Java[1] i Adobe Flash[2] sa očekivanjem da će promeniti svet u korist veb aplikacije – pružajući korisnicima zanimljiva iskustva. Oba su to radila, ali su na kraju nestala iz čitača iz dobrog razloga.

Ono što je programerima i korisnicima potrebno je način da se omogući da veb aplikacije rade više kao izvorne aplikacije. Da smo to imali, naše veb aplikacije bi porasle i omogućile nam da lakše isporučimo aplikacije na više platformi preko čitača, umesto da ručno pravimo izvorne aplikacije za svaku podržanu platformu.

Tokom godina, veb čitači, posebno oni koji rade na mobilnim uređajima kao što su pametni telefoni i tableti, počeli su da nude više izvornih mogućnosti veb aplikacijama. Na primer, moderne veb aplikacije mogu pristupiti sistemu datoteka uređaja i dozvoliti aplikaciji zasnovanoj na čitaču da zna geolokaciju uređaja. Ovo omogućava veb aplikacijama da rade više kao izvorne aplikacije, ali su i dalje postojala ograničenja. Uslužni radnici (eng. service workers) su relativno nova tehnologija koja olakšava veb aplikacijama da premoste jaz između izvornih i veb mogućnosti, uklanjajući mnoga ograničenja iz veb aplikacija.

Ovo je knjiga o uslužnim radnicima.

Da, znam, naslov kaže da je knjiga o progresivnim veb aplikacijama (PWA), i jeste, ali knjiga se fokusira na to kako koristiti uslužne radnike da bi se poboljšale mogućnosti veb aplikacije i kreirale PWA.

Postoji nekoliko knjiga koje se fokusiraju na uticaj PWA; kako se prave PWA koji oduševljavaju i inspirišu korisnike da više koriste aplikaciju. Ovo nije takva knjiga.

Ova knjiga je u najvećoj mogućoj meri fokusirana na tehnologije koje omogućavaju PWA i kako da ih koristite za poboljšanje vaših veb aplikacija kako bi se u vašim veb aplikacijama pružilo iskustvo nalik na izvorno.

Ovde sam sa 15 godina iskustva u razvoju mobilnih uređaja (napisao sam prvu knjigu o razvoju BlackBerryja pre mnogo godina), i, kao što je Simon rekao u predgovoru, PWA su sledeći korak u pretvaranju veba u prvoklasnu razvojnu platformu—posebno za mobilne aplikacije. Moje

---

1 https://en.wikipedia.org/wiki/Java_(software_platform)

2 https://en.wikipedia.org/wiki/Adobe_Flash

interesovanje za pisanje ove knjige fokusira se na uticaj PWA na programere mobilnih uređaja, ali sve ovde važi i za veb aplikacije koje se izvršavaju na desktop čitaču.

Ako ste pročitali neku od mojih knjiga, već znate da ovaj tekst ne sadrži fraze ili tekst na nekom stranom jeziku (ne želim da vas teram da odložite knjigu da biste potražili neku nejasnu frazu na latinskom ili francuskom). Ova knjiga takođe ne sadrži reference na pop kulturu (pa, osim jedne koju opisujem u potpunosti, tako da se nećete osećati zapostavljeno ako je već ne znate).

Za razliku od mojih prethodnih tehničkih knjiga, koje su bile fokusirane na tehnologije i kako ih koristiti (sa mnogo primera koda), ova knjiga je zasnovana na projektima. Dok budete prolazili kroz poglavlja, na različitim mestima ćete početi sa jednom od tri kompletne, samostalne veb aplikacije, a zatim ih pretvoriti u PWA koristeći uslužne radnike i druge tehnologije.

Primeri aplikacija za ovu knjigu napisani su tako da budu jednostavni i laki za razumevanje, tako da nisu posebno fensi. Mogao sam da ih napravim kao neverovatne, reaktivne, moderne veb aplikacije, koristeći JavaScript okvire kao što su VueJS[3] ili React,[4] ali umesto toga sam ih napravio koristeći običan, ravan HTML, CSS i JavaScript. Ovaj pristup uklanja mnogo složenog koda iz aplikacija i ostavlja samo ono što vam je potrebno da biste razumeli temu.

Možete pronaći sav izvorni kôd za primere aplikacija na GitHub-u na https://github.com/johnwargo/learning-pwa-code. Da bih čitaocima olakšao korišćenje štampane verzije ove knjige, dodao sam datoteku `resources.md`[5] u folder sa izvornim kodom za svako poglavlje; navodeći sve veze koje se koriste u poglavlju. Umesto da kucate dugačke URL adrese iz sadržaja poglavlja, možete jednostavno otvoriti datoteku resursa u čitaču i brzo pristupiti bilo kojoj vezi iz poglavlja.

Za razliku od spremišta (eng. repositories) izvornog koda za druge knjige, izvorni kôd knjige ne sadrži samo prethodnu i narednu verzija aplikacija. Umesto toga, dok dovršavate odeljak poglavlja, naći ćete modifikacije izvornog koda samo za taj odeljak u zasebnoj datoteci. Ovo vam omogućava da lakše napravite aplikaciju prateći sadržaj poglavlja imajući dostupan kôd samo tog odeljka u slučaju da imate problem i želite da uporedite završeni kôd sa vašim. Ovaj pristup bi trebalo da pojednostavi vaše praćenje, posebno ako pravite greške u kucanju ili greške dok radite.

Ako imate bilo kakvih pitanja ili komentara o knjizi ili ako pronađete grešku u tekstu ili kodu, pošaljite ih kroz oblast za probleme u GitHub spremištu knjige na https:// github.com/johnwargo/learning-pwa-code/issues. Tamošnji kôd je otvorenog koda, pa ga slobodno koristite u svojim aplikacijama (bilo bi posebno lepo ako biste referencirali izvor koda kako bi drugi mogli da saznaju više o ovoj knjizi).

Ako pronađete grešku u kodu i želite da je sami popravite, uradite to, a zatim pošaljite zahtev za povlačenje (eng. pull request) spremištu, a ja ću pogledati. Obično sam brz u odgovarima, tako da bi trebalo da se javim za koji dan. Budite uviđajni — GitHub je javni forum i svi bi trebalo da se ponašamo prema ljudima tamo kao što očekujemo da se drugi ophode prema nama.

Napravio sam javnu veb lokaciju za knjigu na https://learningpwa.com. Tamo ću vremenom objaviti greške i, nadamo se, srodan sadržaj. U očekivanju eventualnih žalbi, dok ovo pišem, sajt nije PWA. Nije PWA jer ne mora biti PWA. To je samo marketinška stranica za knjigu i nema obaveštenja (eng. notification) za čitače koja želim da šaljem posetiocima, niti postoji potreba za keširanjem sadržaja sajta radi poboljšanja performansi ili korišćenja van mreže. Samo zato što mogu da napravim sajt u PWA ne znači da treba. Koristite moć PWA samo za one sajtove kojima su zaista potrebne.

---

3 https://vuejs.org/

4 https://reactjs.org/

5 https://github.com/johnwargo/learning-pwa-code/blob/master/chapter-01/resources.md

Dva od tri primera aplikacije iz knjige su objavljene i na mreži. Prvi je Tip Calculator iz poglavlja 2, kome možete pristupiti na https://learningpwa.com/tipcalc/. Drugi je sajt PWA News iz poglavlja 3, 4 i 5 — tu aplikaciju možete pronaći na https://pwa-news.com. Sajt Learning Progressive Web Apps je statičan sajt koji se nalazi na Netlify[6] tako da će verovatno ostati aktivan sve dok je knjiga javno dostupna. Sajt PWA News zahteva računarske resurse (na strani servera je aplikacija node.js[7]), tako da ću ostaviti sajt sve dok postoji interesovanje za knjigu. Na kraju ću ga ugasiti, ali potpuni izvor za server je uključen u izvorni kôd knjige, tako da ćete uvek moći sami da pokrenete lokalnu kopiju servera ili da hostujete sajt negde drugde.

To je to! Ovaj predgovor sadrži sve što sam želeo da vam kažem o knjizi. Verujem u PWA i mislim da su one budućnost razvoja mobilnih aplikacija. Zaista sam uživao dok sam pisao, i nadam se da ćete i vi uživati čitajući, isto toliko ili više.

Registrujte svoju primerak knjige Learning Progressive Web Apps na InformIT sajtu za praktičan pristup ažuriranjima i/ili ispravkama čim postanu dostupne. Da biste započeli proces registracije, idite na informit.com/register i prijavite se ili napravite nalog. Unesite ISBN broj knjige engleskog izdanja (9780136484226) i pritisnite na Submit. Potražite na kartici Registered Products (registrovani proizvodi) vezu za pristup bonus sadržaju pored ovog proizvoda i pratite tu vezu da biste pristupili svim dostupnim bonus materijalima. Ako želite da budete obavešteni o ekskluzivnim ponudama za nova izdanja i ažuriranja, označite polje za prijem e-pošte od nas.

6 https://www.netlify.com/

7 https://nodejs.org/

Sadržaj

Predgovor ix
Uvod x
Izrazi zahvalnosti xiii
O autoru xiv

1 Uvod u progresivne veb aplikacije 1
Prvo, malo istorije 2
PWA su... 2
Pravljenje progresivne veb aplikacije 4
PWA tržišni uticaj 6
PWA i prodavnice aplikacija 7
Zaključak 7

2 Datoteke manifesta veb aplikacije 9
Sačuvaj na početnom ekranu 10
Omogućavanje instaliranja veb aplikacije 16
Anatomija manifesta veb aplikacije 17
Zadavanje imena aplikacije 18
Podešavanje ikona aplikacija 18
Konfigurisanje režima prikaza 19
Zadavanje početne URL adrese instalirane aplikacije 23
Podešavanje opcija aplikacije 25
Dodatne opcije 25
Kontrola iskustva instalacije 26
Priprema za kodiranje 27
Node.JS 27
Git Client 28
Visual Studio Code 29
Instalacija aplikacije u akciji 29
Dodavanje opcije Service Worker 29
Dodavanje datoteke veb manifesta 32
Pokretanje aplikacije 32
Poboljšanje procesa instalacije 35
Rešavanje problema 41
Generisanje manifesta 42
Zaključak 42

3 Uslužni radnici 43
PWA vesti 43
Predstavljamo uslužne radnike 44

Priprema za kodiranje 46
Preduslovi 47
Kretanje po izvornom kodu aplikacije 48
Konfigurisanje API-ja servera 49
Pokretanje servera 51
Registrovanje uslužnog radnika 52
Opseg dejstva uslužnog radnika 59
Životni ciklus uslužnog radnika 60
Forsirana aktivacija 62
Traženje dodatnih kartica čitača 62
Uočavanje promene uslužnog radnika 63
Forsiranje ažuriranja uslužnog radnika 64
`ready` obećanje uslužnog radnika 65
Zaključak 65

4 Keširanje resursa 67
Interfejs keša uslužnih radnika 67
Priprema za kodiranje 68
Keširanje resursa aplikacije 70
Upravljanje kešom 77
Vraćanje objekta podataka u slučaju greške 82
Dodavanje stranice van mreže 85
Implementacija dodatnih strategija keširanja 90
Samo-keš memorija 90
Prvo-mreža, zatim-keš 91
Prvo-mreža, ažurirajte keš 92
Zaključak 96

5 Rad van mreže uz sinhronizaciju u pozadini 97
Uvod u sinhronizaciju u pozadini 98
Oflajn sinhronizacija podataka 101
Izbor baze podataka za sinhronizaciju 103
Izrada baze podataka 103
Izrada skladišta 104
Dodavanje podataka 105
Brisanje objekata 106
Iteracija kroz podatke pomoću kursora 107
Priprema za kodiranje 108
Poboljšanje aplikacije PWA News 108
Priprema uslužnih radnika za sinhronizaciju u pozadini 109
Ažuriranje veb aplikacije za korišćenje pozadinske sinhronizacije 109

Kompletiranje uslužnog radnika 116
Suočavanje sa poslednjim šansama 122
Zaključak 125

6 Push obaveštenja 127
Uvod u push obaveštenja 127
Arhitektura daljinskog obaveštenja 130
Priprema za kodiranje 132
Generisanje ključeva za šifrovanje 132
Provera podrške za obaveštenja 136
Proveravanje dozvole za obaveštenja 136
Dobijanje dozvolje za obaveštenja 137
Lokalna obaveštenja 140
Opcije obaveštenja 142
Prijava na obaveštenja 145
Odjava od obaveštenja 151
Daljinska obaveštenja 153
Obrada isteka prijave 159
Slanje obaveštenja push servisima 160
Zaključak 162

7 Prenošenje podataka između uslužnog radnika i veb aplikacija 163
Priprema za kodiranje 164
Slanje podataka iz veb aplikacije uslužnom radniku 165
Slanje podataka od uslužnog radnika do veb aplikacije 166
Dvosmerna komunikacija pomoću `MessageChannel` 169
Zaključak 177

8 Procena, automatizacija i primena 179
Procena kvaliteta PWA pomoću Lighthouse 179
Priprema za kodiranje 180
Korišćenje dodatka Lighthouse 180
Korišćenje Lighthouse Tools u čitaču 185
Korišćenje Lighthouse Node modula 187
PWABuilder 188
Korišćenje korisničkog intefejsa PWABuildera 189
Kreiranje primenjivih aplikacija 193
Korišćenje PWABuilder CLI 195
PWABuilder i Visual Studio 196
PWA i Microsoft prodavnica 200
Zaključak 203

9 Automatizacija uslužnog radnika pomoću Google Workboxa 205
Predstavljamo Workbox 205
Generisanje uslužnog radnika za prekeširanje 206
Dodavanje prekeširanja postojećem uslužnom radniku 213
Kontrolisanje keš strategija 216
Zaključak 222

Indeks 223

1

Uvod u progresivne veb aplikacije

Progresivna veb aplikacija, PWA (eng. Progressive Web App) je veb aplikacija koja koristi specijalne mogućnosti čitača koje omogućavaju aplikaciji da deluje više kao izvorna ili mobilna aplikacija kada se pokreće na odgovarajućim čitačima. To je to, to je sve što treba da se kaže–Ostatak ove knjige je u tome da vam pokaže kako da ih napravite (sa mnogo primera koda, naravno).

Programeri grade PWA prvenstveno koristeći dve tehnologije dostupne u većini modernih čitača: datoteke manifesta veb aplikacije (eng. web app manifest files) i uslužne radnike (eng. service workers). Poglavlje 2, „Datoteke manifesta veb aplikacije", govori vam skoro sve što treba da znate o datotekama manifesta veb aplikacije. Preostala poglavlja sadrže usluge radnika i šta možete da učinite sa njima.

Ne postoji standard ili telo standarda za PWA; PWA je samo veb aplikacija napravljena da deluje na određen način. Koristite onoliko ili malo PWA funkcionalnosti koliko želite u svojim veb aplikacijama. Možete imati veb aplikacije koje koriste neke PWA mogućnosti, ali nisu PWA i PWA koje koriste samo neke PWA mogućnosti.

PWA ima sopstveni logotip koji vodi zajednica, prikazan na slici 1.1. Možete čak i da kreirate sopstvenu verziju logotipa, navodeći bilo koju boju za *W* deo koristeći alatku na https://diekus.net/logo-pvinter.



Slika 1.1 PWA logotip zajednice

Nažalost, postoje različiti pogledi na to šta su PWA; dozvolite mi da vam objasnim svoje viđenje...

Prvo, malo istorije

Google i drugi tvorci čitača stalno dodaju posebne funkcije svojim čitačima, nastojeći da obezbede više alatki koje programeri mogu da koriste za poboljšanje veba. Godine 2015, projektant Frances Berriman i inženjer Google Chroma Alex Russell skovali su termin *Progresivne veb aplikacije* da opisuju, prema Vikipediji, „aplikacije koje koriste prednosti novih funkcija koje podržavaju savremeni vab čitači, uključujući uslužne radnike i manifesti veb aplikacija, koji korisnicima omogućavaju nadogradnju veb aplikacija na progresivne veb aplikacije u svom matičnom operativnom sistemu (OS).“

Google Chrome je u to vreme bio vodeći čitač koji je podržavao potrebne tehnologije, tako da je Google počeo snažno da promoviše ovaj novi pristup pravljenja veb aplikacija.

PWA su...

Guglove polazne stranice[1] za progresivne veb aplikacije kažu da su PWA

- Pouzdane
- Brze
- Privlačne

Ovi termini opisuju percepciju korisnika o aplikaciji, a ne o tome kako je aplikacija napravljena, koje tehnologije koristi ili šta može da uradi. Tehnologije koje se koriste za pravljenje PWA omogućavaju programerima da grade veb aplikacije koje su pouzdane, brze i privlačne, ali samo ako veb programeri imaju veštine i odvoje vreme da to urade. Mislim da je Guglov opis neosnovan, pošto postoji mnogo neiskusnih veb programera koji mogu da naprave PWA koji nisu pouzdane, brze ili zanimljive, ali aplikacije su i dalje PWA.

Dakle, imajući to na umu, u ovoj knjizi, PWA se odnose na tehnologije koje se koriste i kako ih napraviti. Postoji mnogo onlajn članaka i knjiga dostupnih o tome kako su PWA sve angažovanije i o uticaju koji su imale na uspeh ili prihod preduzeća; najbolje mesto za početak za te informacije je PWA statistika.[2]

Slažem se sa definicijom Jeremy Keitha iz *Šta je progresivna veb aplikacija?*[3] Evo mog mišljenja: PWA su veb aplikacije sa malo posebne magije koja im omogućava da se ponašaju više kao mobilne aplikacije. Koncept veb aplikacija koje funkcionišu više kao mobilne aplikacije čini PWA tako zanimljivim.

Ako pomislite na većinu izvornih mobilnih aplikacija koje koristite na telefonu:

- Mogu se instalirati iz javnih prodavnica aplikacija telefonske platforme (i prodavnica poslovnih aplikacije), stavljajući ikonu aplikacije na početni ekran uređaja.
- Učitavaće se i raditi (iako na potencijalno ograničen način) bez obzira na to da li uređaj ima mrežnu povezanost. Korisnički interfejes (UI) aplikacije je ugrađen u binarni program aplikacije i učitava se lokalno kada otvorite aplikaciju (podaci se preuzimaju sa mreže kada je to moguće).

1 https://developers.google.com/web/progressive-web-apps/

2 https://www.pwastats.com/

3 https://adactio.com/journal/13098

- Mogu da rade u pozadini, čak i da rade ili obrađuju podatke dok radite nešto drugo na uređaju.
- Mogu da primaju push obaveštenja (poruke poslate na uređaj ili aplikaciju od strane spoljnog sistema).
- Uz nešto dodatnog rada, mogu čak da pruže bogato i robusno iskustvo van mreže, keširanje novih zapisa za kasnije otpremanje (eng. upload) i sinhronizovanje podataka sa aplikacijom zasnovanom na serveru.

Da li to opisuje mnoge mobilne aplikacije koje danas koristite? Da li vaša aplikacija za e-poštu (ili bilo koja druga aplikacija) funkcioniše bez obzira na to da li uređaj ima mrežnu vezu? Ako isključite radio i otvorite aplikaciju, da li se korisnički interfejs aplikacije otvara i prikazuje veći deo podataka koji su bili tamo ranije? Kada pošaljete poruku dok niste na mreži, da li aplikacija stavlja podatke u red za kasniji prenos? Da li aplikacija prima obaveštenja (na primer kada je nova pošta dostupna)? Da, tako je; to su osnovne mogućnosti modernih mobilnih aplikacija.

Prelazak na PWA:

- PWA se mogu instalirati: korisnici mobilnih i desktop računara mogu brzo da ih instaliraju na početni ekran ili radnu površinu svog telefona pomoću korisničkog interfejsa za instalaciju koji se nalazi u aplikaciji. Mobilni telefoni su skoro uvek imali mogućnost da kopiraju URL veb lokacije na početni ekran uređaja, ali je bolje instalirati aplikaciju. Više o tome objašnjavam u poglavlju 2.
- PWA keširaju osnovni korisnički interfejs aplikacije na lokalnom uređaju, tako da kada korisnik otvori aplikaciju, korisnički interfejs se brzo učitava pre nego što aplikacija pokuša da se poveže i dobije ažurirane podatke sa mreže. Shodno tome, PWA se deluju brže od tradicionalnih veb aplikacija.
- PWA izvršavaju zadatke u pozadini, omogućavajući keširanje resursa i obradu u pozadini. Tradicionalne veb aplikacije ne mogu to da urade (pa, mogu, ali je potrebno mnogo ručno izrađenog koda ili korišćenje biblioteka koda).
- PWA mogu da primaju push obaveštenja sa pozadinskog servera bez obzira da li je aplikacija pokrenuta.

Kada je Apple najavio iPhone, famozno su promašili kada su pretpostavili da će programeri biti srećni da imaju mogućnost da prave veb aplikacije samo za platformu. Stiv Džobs i kompanija su arogantno pretpostavili da je čitač dovoljan i da može da se prilagodi većini potreba. Zajednica programera se brzo pobunila, prisilila Applea, i tako smo dobili mogućnost da kodiramo izvorne aplikacije na iOS-u.

Pobunili su se jer iako je Safari bio prilično sposoban na iOS-u, a Apple je obezbedio neke odlične alate za programere (ne, ne Xcode, nego, ako se neko seća Dashcodea[4], jednostavno postojale su stvari koje programeri aplikacija nisu mogli da urade u čitaču. Ova ograničenja su takođe bila motivacija za projekat PhoneGap (koji je kasnije postao Apache Cordova); projekat je započet posebno da bi se omogućio razvoj mobilnih aplikacija na više platformi i dao programerima čitača pristup mogućnostima uređaja koje u to vreme nisu bile dostupne u ugrađenom čitaču.

Tokom godina, mnoge od tih mogućnosti su dospele u čitač kroz otvorene standarde koji pokrivaju stvari kao što su akcelerometar uređaja, kamera, kompas, mikrofon i još mnogo toga.

4 https://en.wikipedia.org/wiki/Dashcode

Aplikacije na čitaču su postale sposobnije, ali nekoliko konačnih zahteva koji su čitaču davali jednaku osnovu sa izvornim mobilnim aplikacijama bili su–i ovo bi trebalo da zvuči poznato–mogućnosti instaliranja veb aplikacije na uređaj, keširanje koda i sadržaja lokalno, pokretanje zadataka u pozadini (čak i kada aplikacija nije pokrenuta) i primanje push obaveštenja. Tehnologije koje omogućavaju PWA isporučuju te konačne potrebne mogućnosti čitaču; i, sa ovim mogućnostima, veb aplikacija sada može da deluje više kao izvorna aplikacija.

Pravljenje progresivne veb aplikacije

Nekoliko tehnologija omogućava PWA; bez njih, veb aplikacija jednostavno ne može biti PWA. Te tehnologije su datoteke manifesta veb aplikacije, uslužni radnici i bezbedni protokol za prenos hiperteksta (HTTPS) ili, tačnije, HTTP over Transport Layer Security (TLS). HTTPS omogućava bezbednu komunikaciju između klijenta i servera i nije tema koju obrađujem u ovoj knjizi osim da kasnije opišem da je potreban za većinu PWA mogućnosti i objasnim zašto. Sledeća poglavlja detaljno opisuju kako da koristite datoteke manifesta veb aplikacije i uslužne radnike za pravljenje PWA. Za sada je dobro znati da datoteke manifesta veb aplikacije omogućavaju instalaciju veb aplikacije na lokalnom sistemu na kome se izvršava aplikacija, a uslužni radnici rade sve ostalo da bi se omogućile PWA.

Tabela 1.1 daje mogućnosti aplikacije iz prethodnog odeljka mapirane prema PWA tehnologijama koje ih omogućavaju.

Tabela 1.1 **PWA mogućnosti prema tehnologiji implementacije**

Mogućnost	Veb manifest	Uslužni radnici	TLS (HTTPS)
Instaliraj prečicu	Da	Obavezno, ali nije uključeno	Da
Sadržaj keša	Ne	Da	Da
Pozadinska obrada	Ne	Da	Ne
Push obaveštenja	Ne	Da	Da

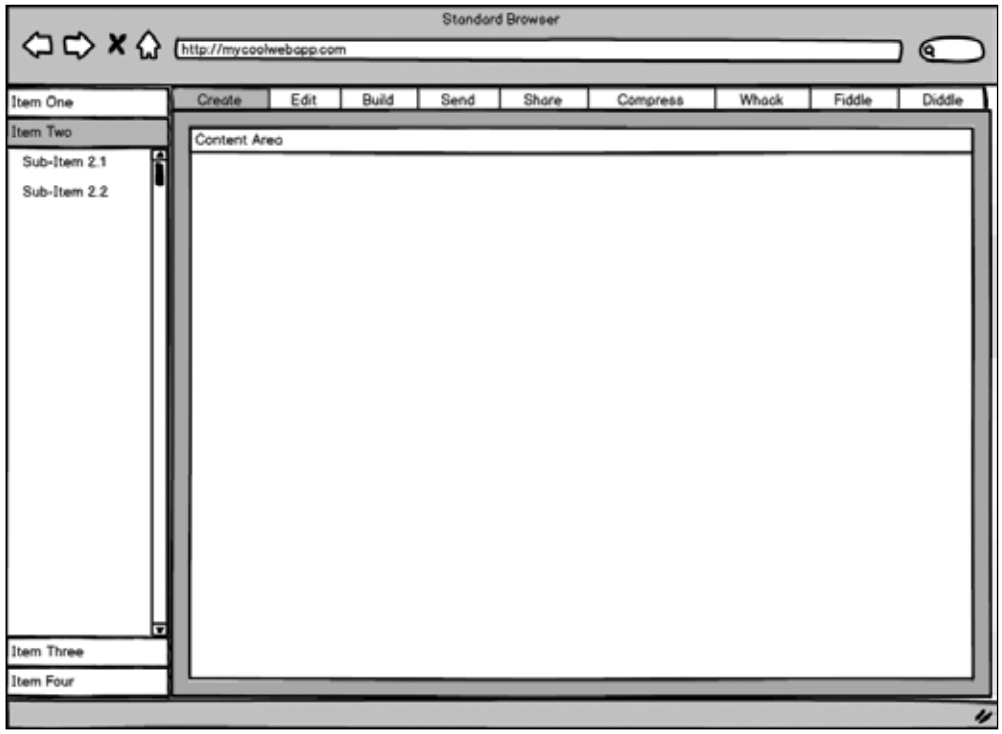
Programer mora da uradi dosta posla da bi veb lokacija bila zanimljiva, primamljiva i korisna. To znači da napravite stranice sajta tako da izgledaju dobro i dobro funkcionišu na ekranima ili prozorima čitač različitih dimenzija (širine i visine). Veb programeri moraju da uzmu u obzir mogućnosti čitča na strani klijenta kada prave ove sajtove, a mnogi programeri isporučuju različite mogućnosti na svojim sajtovima na osnovu toga šta radi ili ne radi u čitaču koji se koristi tokom rada. Neki mobilni čitači (poput onih na naprednim telefonima) ne mogu da obrađuju mnogo koda na strani klijenta, tako da JavaScript može biti onemogućen ili ima ograničene mogućnosti na tom uređaju.

Proces koji programeri koriste za pravljenje veb stranica za ove scenarije naziva se *progresivno poboljšanje*, gde se programer prvo fokusira na sadržaj, a zatim postavlja slojeve na dodatnu funkcionalnost za adekavatne čitače. To znači da aplikacija jednostavno radi, prikazujući potreban sadržaj, bez obzira na to da li čitač podržava dodatne napredne mogućnosti (kao što su fensi CSS transformacije) koje je programer dodao za robusnije čitače. Ako te mogućnosti postoje, onda korisnik dobija potpuno iskustvo. Ako nisu, korisnik može barem da vidi sadržaj.

Veb aplikacije su interaktivnije, kao što je Vikipedija (veb sajt) u odnosu na Gmail (veb aplikacija). Veb aplikacije koriste jezike za skriptovanje na strani klijenta kao što je JavaScript (ili njegov

bratski jezik TypeScript) da bi pružile dinamičnije i uzbudljivije iskustvo na strani klijenta. Kod veb aplikacije, kada prvi put pritisnete URL aplikacije, školjka (eng. shell) aplikacije se preuzima i prikazuje pre nego što aplikacija ode na server za podatke aplikacije.

Možete videti primer ovoga na slici 1.2. Školjka aplikacije sastoji se od bočne navigacije i gornje trake sa dugmadima, dok se sadržaj u oblasti sadržaja (unutar sivog okvira) dinamički menja na osnovu izbora koje korisnik čini u aplikaciji.



Slika 1.2 Struktura školjke veb aplikacije

PWA progresivno poboljšavaju veb aplikacije–zato se zovu progresivne veb aplikacije.

Dakle, pretpostavimo da imamo dinamičku veb aplikaciju koja se bavi interakcijom korisnika i dinamičkim podacima. Aplikacija zahteva sposobniji čitač i svakako znači da je uključen JavaScript (da bi se pružila dinamička priroda aplikacije). Aplikacija se pokreće u čitaču i koristi mrežnu vezu sistema za preuzimanje ljuske, a zatim, ili možda paralelno, preuzima podatke aplikacije sa servera. Kada čitač koji pokreće aplikaciju nema mrežnu vezu, čitač prikazuje spiner ili poruku o grešci, a korisnik jednostavno ne može da koristi aplikaciju.

Kao što mu i samo ime govori - progresivan (PWA), poboljšaćete aplikaciju kada dodate datoteku veb manifesta da biste omogućili instalaciju na strani klijenta i dodate uslužnog radnika koji će omogućiti keširanje, obradu u pozadini i podršku za push obaveštenja. U čitačima koji podržavaju te mogućnosti, PWA se može instalirati i omogućava korisniku da pokrene aplikaciju kada nema mrežne veze (keširanje uslužnog radnika), čuva nove ili ažurirane zapise za kasnije otpremanje kada postoji mrežna veza (keširanje uslužnog radnika u kombinaciji sa pozadinskom obradom), i obaveštava korisnike kada stignu nove push poruke (pozadinska obrada uslužnog radnika).

Šta na sistemima koji ne podržavaju datoteke veb manifesta i/ili uslužne radnike? Aplikacija radi na isti način kao pre nego što ste dodali datoteku veb manifesta ili uslužnog radnika u aplikaciju. Kada pristupite aplikaciji, sadržaj i podaci aplikacije se traže u realnom vremenu sa jednog ili više servera.

PWA tržišni uticaj

Dakle, kakve uticaje PWA imaju na tržište? Razvojne organizacije su brzo usvojile ovu tehnologiju jer progresivna priroda pristupa omogućava razvojnim timovima da dramatično utiču na performanse (za svakog korisnika koji koristi čitač koji ga podržava) za svoje korisnike uz samo nekoliko malih, jednostavnih promena. Kada korisnici pokreću aplikaciju u čitaču koji ne podržava PWA tehnologije, neće primetiti ništa drugo–aplikacija jednostavno nastavlja da radi kao i uvek.

Performanse veb aplikacija, posebno na mobilnim uređajima, odjednom izgledaju bolje za neke aplikacije. To je zato što mnogi programeri aplikacija implementiraju datoteke veb manifesta i uslužne radnike, a ove aplikacije sada aktivno upravljaju svojim ponašanjem na strani klijenta (koristeći pozadinsku obradu i keširanje) u korist korisnika.

Korisnici se više angažuju sa aplikacijama koje podržavaju PWA. To je samo zato što su aplikacije brže (brže se učitavaju) i rade čak i kada mobilni uređaj na kome radi nema mrežnu vezu. Aplikacija možda nema sve svoje funkcije kada je van mreže, ali i dalje radi i to je verovatno sve što je potrebno da bi korisnici bili srećni.

Moderni okviri veb aplikacija i alati za razvoj na više platformi su u velikoj meri usvojili PWA. Na primer, React[5], veoma popularna JavaScript biblioteka za pravljenje korisničkih interfejsa (UI), automatski unapred učitava svoj početni uzorak projekta sa uslužnim radnikom. Po difoltu je onemogućeno, ali sve što morate da uradite da biste React aplikaciju pretvorili u PWA je da promenite jednu liniju koda, kao što je opisano u Listingu 1.1.

Listing 1.1 **Primer React aplikacije index.js**

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

// Ako želite da vaša aplikacija radi van mreže i da se učitava brže,
// možete dole da promenite unregister() u register().
// Obratite pažnju da to stvara određene probleme.
// Više o uslužnim radnicima pročitajte ovde: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

U poslednje vreme mnogo radim koristeći Ionic Framework. Ionic je višeplatformski korisnički interfejs i komponentni okvir za pravljenje mobilnih i desktop aplikacija. Do nedavno, Ionic aplikacije su obično bile napravljene pomoću Angulara, ali Ionic tim je nedavno prešao na veb

5 Naučite React, u izdanju Mikro knjige: https://www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-427-1

komponente i Stencil (njihov otvoreni kompajler veb komponenti) i objavio PWA Toolkit koji pruža PWA-prvi pristup veb aplikacijama.

Ono što je najvažnije, PWA je lakše (mnogo, mnogo lakše) kreirati nego izvorne mobilne aplikacije. To znači da ove aplikacije koštaju manje za pravljenje i održavanje. Kada pakujete PWA u aplikaciju Apache Cordova, Ionic Capacitor ili GitHub Electron, odgovarate na potrebe korisnika koji žele izvornu verziju za mobilne uređaje ili desktop računare.

Tehnologija je toliko popularna da je istraživačka firma Gartner predvidela da će do 2020. 50% potrošačkih mobilnih aplikacija biti PWA (dao bih vam onlajn referencu za ovo, ali ta izjava je svuda na mreži). Ali to ste znali, zar ne? Već ste kupili, pozajmili ili ukrali ovu knjigu da biste naučili kako da napravite PWA.

PWA i prodavnice aplikacija

Programeri prave izvorne mobilne aplikacije iz mnogo razloga, ali jedan od njih je da mogu da distribuiraju svoje aplikacije preko javne prodavnice aplikacija platforme za mobilne uređaje. Iako možete da instalirate PWA tako što ćete samo otvoriti veb aplikaciju u mobilnom čitaču (sve o tome ćete saznati u sledećem poglavlju), prodavci mobilnih platformi su obučili svoje korisnike da traže aplikacije u prodavnicama aplikacija. To znači da će korisnici koji traže vašu aplikaciju prvo da je traže u prodavnici aplikacija na svom telefonu.

Neke zanimljive stvari se dešavaju u ovom prostoru. Početkom 2019. Google je najavio Trusted Web Activity[6] (TWA) za Android. TWA omogućava programerima da spoje PWA u matičnu mobilnu aplikaciju za Android. TWA je u suštini Chrome čitač bez hromiranja (bez korisničkog interfejsa čitača), preko celog ekrana u Android aplikaciji. Uz to, dobijate PWA upakovan sa čitačem u matičnoj aplikaciji–svež i spreman za primenu preko Google Play prodavnice.

Microsoft je zauzeo drugačiji pristup, najavljujući sredinom 2018. da programeri mogu da objave svoje PWA u Microsoft Storu. Ako ne objave svoju aplikaciju u prodavnici, Microsoft će to ipak učiniti umesto njih (ako PWA ispunjava određene kriterijume). Više o ovome ćete saznati u 8. poglavlju, „Procena, automatizacija i primena“.

Zaključak

U ovom poglavlju dao sam vam kratak uvod u progresivne veb aplikacije. Naći ćete mnogo više materijala na mreži i mnoge autore koji se bave tržištem i primenom PWA. Pošto je ovo zaista knjiga o uslužnim radnicima, želim da dođem do kodiranja što je pre moguće.

U sledećem poglavlju pokazaću vam sve o tome kako da svoj PWA učinite instaliranim pomoću uslužnih radnika i datoteka veb manifesta. Nakon toga, ostatak knjige je o uslužnim radnicima i svemu što možete da uradite uz njihovu pomoć.

6 https://developers.google.com/web/updates/2019/02/using-twa