報道各位

2019 年 11 月 5 日
株式会社インプレスR&D
https://nextpublishing.jp/

はじめて PWA にトライするエンジニアのために！

『Try PWA』発行

技術の泉シリーズ、11 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『Try PWA』(著者:渋田 達也)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『Try PWA』
https://nextpublishing.jp/isbn/9784844396963

著者:渋田 達也
小売希望価格:電子書籍版 1600 円(税別)／印刷書籍版 1800 円(税別)
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 120 ページ
ISBN:978-4-8443-9696-3
発行:インプレス R&D

＜＜発行主旨・内容紹介＞＞

本書はモバイル向け Web サイトをネイティブアプリのように使える仕組みである PWA(Progressive Web Apps)の概要と簡単な実装方法を 1 ステップずつ解説したチュートリアルです。

Firebase Hosting を使った配信や Firebase を使った Web Push の実装、AWS Lambda で実装する Serverless Push Server などについて解説しています。

(本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

PWA の概要をネイティブアプリと比較しながら紹介

第2章　PWAの概要やツールの紹介

2.1　PWAの概要

まずはPWAとはどういうものかを解説します。PWAとは「Progressive Web Apps」の略称です。PWAはGoogleが提唱した、Webアプリケーションのひとつの姿です。

ネイティブアプリとWeb、そしてPWA

PWAとはなにかをざっくりと説明すると、「ネイティブアプリとWebのいいとこ取りをしたもの」です。次の図は「Progressive Web Apps - PWA Roadshow」という動画のスクリーンショットです。これはYouTubeで公開されています。

出典：Progressive Web Apps - PWA Roadshow（https://youtu.be/z2JgN6Ae-Bo?t=166）

図2.1:

オレンジ色（左上）がネイティブアプリで青色（右下）がWebです。また縦軸のCapabilityはできること、Reachはどれだけの人に届くかという指標です。見ていただければわかるのですが、ネイティブはできることが多く、リーチ力が弱い。Webはリーチ力が強く、できることが少ない。PWAはこのふたつの指標をどちらも満たす可能性があります。

FIREとは

PWAはWebにおけるUXの目指す姿でもあります。

すでにご存知の方もいらっしゃるかもしれませんが、FIREという指標が存在します。これは、「Fast、Integrated、Reliable、Engaging」の頭文字を取ったものです。今ではFIREはWebについての指標という扱いになっていますが、以前はPWAの指標でした（PWAのときはIntegratedを除く3つ）。筆者はこの扱いの変化から、GoogleはPWAで目指していたことをWebの当たり前にしたいのではないのかと考えます。

ではFIREについて軽く解説します。まずFIREを構成する4単語を筆者は次のように訳しています。

・Fast：速さ
・Integrated：デバイスへの統合
・Reliable：動作の信頼性
・Engaging：魅力的であること

それぞれ少し紹介します。

Fastは文字通り速さです。これは読み込みのスピードやスムーズに操作できるという速さを表しています。

IntegratedはWebアプリケーションの体験をデバイスに合わせることです。たとえば、PWAの特徴とも言えるホーム画面からの起動です。ホーム画面にWebアプリケーションを追加することで、ユーザーはブラウザーを起動しなくてもホーム画面にいるアイコンをタップし、Webアプリケーションを利用できます。ホーム画面に配置されていることでネイティブアプリケーション同様の起動体験を提供できます。

Reliableは動作の信頼性です。これは主にネットワーク接続についての信頼性を示しています。ネットワークに接続していないと起動できないアプリケーションをユーザーはどう思うでしょうか？せっかくホーム画面に追加しているのに起動できないアプリケーションを、アプリケーションと認めてもらえるでしょうか？少し言い過ぎかもしれませんが、Reliableはこのように「アプリケーションが常に開ける」という信頼性に関する指標になります。

Engagingはアプリケーションの体験が魅力的であることです。どのようなアプリケーションが魅力的であると言えるでしょうか？サイトにアクセスた時、いきなりプッシュ通知を許可しますか？と聞いてくるサイトは魅力的でしょうか？答えはもちろん違います。これはユーザーが意図していない動作です。このような意図していない動作・機能ではなく、ユーザーの意図通りに動かせる快適なアプリケーションを作ろうという指標になります。

この4つを突き詰めるとWebアプリケーションとしてのUXが高まります。FIREを意識して開発することはユーザーへの思いやりとも言えるかもしれません。

どういうものがPWA？

PWAのチェックリストをGoogleが公開しています。
https://developers.google.com/web/progressive-web-apps/checklist

PWA の実装方法を丁寧に解説

図3.5:

もちろん、自分で用意しても問題ありません。

画像が準備できたので、192pxサイズと512pxサイズのアイコンを作りましょう。せっかくなので、2章で紹介したこのサービスでアイコンを作っていきます。

- Favicon Generator- https://realfavicongenerator.net/

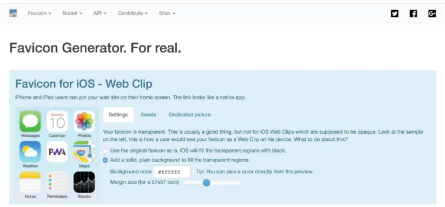
まず、画面右側の「Select your Favicon picture」で用意した画像を選択します。260px×260px以上の画像が望ましいと書いてありますが、今回は512pxのアイコンが必要なので、512px×512pxの画像を使う方がよいでしょう。

選択すると、画像によっては確認のモーダルが表示されます。確認して問題なければモーダル右下の「Confirm ~」を押しましょう。

次の画面は各OSごとのアイコンを設定する画面です。

各OSごとのアイコンを設定をする画面

図3.6:

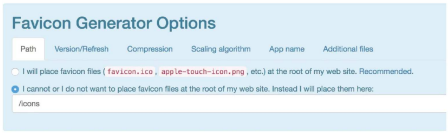


ここは一番下の「Favicon Generator Options」以外は自由に設定しても問題ありません（面倒であればそのままでもOKです）。

Favicon Generator OptionsのPathタブを次の画像のように`/icons`と入力してください。これはアイコンを置いておくディレクトリーのパスを指定するためです。

アイコンセットを配置するディレクトリーの設定

図3.7:



問題なければ「Generate your Favicons and HTML code」を押し、アイコンセットや必要なmetaタグ、Web App Manifestなどがまとめて生成されます。今回は、一番左のタブの「HTML5」のものを使います。

まずは「Favicon package」ボタンをクリックし、アイコンセットをダウンロードします。ダウンロードしたアイコンセット解凍し、中身をVue.jsのプロジェクトの`/public/icons`ディレクトリーに入れます。

※`icons`ディレクトリーはまだ作られていないので自分で作る必要があります。

次にmetaタグを`/public/index.html`にコピペします。

次のようにtitleタグの下に追加すればよいでしょう。

※タグの種類や中身は記載されているものから更新されている可能性もあります。

```
<link rel="icon" href="<%= BASE_URL %>favicon.ico">
<title>try-pwa-simple</title>
<link rel="apple-touch-icon" sizes="180x180" href="/icons/apple-touch-icon.jpg">
<link rel="icon" type="image/png" sizes="32x32" href="/icons/favicon-32x32.jpg">
<link rel="icon" type="image/png" sizes="16x16" href="/icons/favicon-16x16.jpg">
<link rel="manifest" href="/icons/site.webmanifest">
<link rel="mask-icon" href="/icons/safari-pinned-tab.svg" color="#5bbad5">
<link rel="shortcut icon" href="/icons/favicon.ico">
<meta name="msapplication-TileColor" content="#2d89ef">
<meta name="msapplication-config" content="/icons/browserconfig.xml">
<meta name="theme-color" content="#ffffff">
```

PWA でプッシュ通知を実現するための技術要素も紹介

第4章　Web Pushの実装解説

この章ではWeb Pushの実装を解説します。

Web Pushをするためにはプッシュ通知を送るためのサーバーが必要です。そのため、この章ではサーバーの実装とクライアントの実装を行き来します。サーバーもNode.jsで実装するため、すべてJavaScriptでの実装になります。

サーバー自体はローカル環境でも動くように、Express.jsを使い実装します。最終的には、サーバーはAWS Lambdaで稼働させます。AWSは触れないけどプッシュ通知はやってみたい、という方でも読み進められるでしょう。

どうしてLambdaなのか？ですが、単純に筆者がAWSをメインで使っていて、サーバーレスのプッシュ通知サーバー作れたらおもしろいなという理由です。ただ、サーバーレスであるメリットは大いにあります。プッシュ通知は常時使われているような機能ではなく、必要なときに必要なだけ送信できればOKです。サーバーレスは常時稼働しているサーバーと比べて、負荷やコスト面における優位性があります。今回のプッシュ通知のように必要なときに必要なだけ使えるので、プッシュ通知の機能だけを持ったサーバーをマイクロサービスとしてサーバーレスで作るのはメリットが多いのです。

では、実装の解説に入っていきましょう。この章では次の要素が出てきます。それぞれの詳細な解説はしていないのでご了承ください。

・Firebase
・AWS
・Vue.js
・Serverless Framework
・Express.js

この章は次のような順で解説します。

1．全体像解説
2．クライアントとサーバーのやりとりを円滑にするための準備
3．Push通知を行うための下準備
4．プッシュ通知サーバーの実装解説
5．クライアントの実装解説
6．Lambdaへのデプロイ

またプッシュ通知はAndroid端末、またはPCのGoogle Chromeを対象として解説します。iOSは対応していないので、Android端末をお持ちでない方はPCで動作をご確認ください。

この章のサンプルコードのリポジトリーは次のとおりです。ご活用ください。リポジトリーは最終型の状態になっているので注意が必要です。

クライアント：
https://github.com/mya-ake/try-pwa-apis/tree/master/packages/try-pwa-web-push-client
サーバー：
https://github.com/mya-ake/try-pwa-apis/tree/master/packages/try-pwa-web-push-server

全体像解説

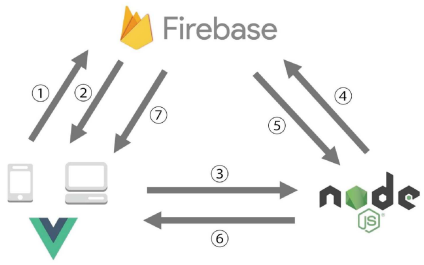
まず、作るものの全体像から解説します。現状プッシュ通知を送るためには、Firebase Cloud Messaging（以降FCM）を利用する必要があります。これはWebに限らずAndroidのネイティブアプリでも同様です。iOSの場合はAPNSというAppleのプッシュ通知サービスを利用します。iOSでのプッシュ通知はまだsafariに実装されていないので、しばらく様子見です。

今回も例に漏れずFCMを使っていきます。

今回はシンプルにクライアントがプッシュ通知を送るボタンを押すと、その端末にプッシュ通知を送るという処理を実装します。この実装におけるプッシュ通知を送るときは、次の図のような順でリクエストが流れていきます。登場人物は、左下のスマホやPC、Vue.jsのロゴがあるところがクライアントです。右下のNode.jsのロゴがあるところがサーバーです。上のFirebaseのロゴがFCMです。この3つが今回の登場人物になります。

FCM利用時のプッシュ通知を送るためのフロー図

図4.1:



＜＜目次＞＞

5.3 Service Worker の注意点
5.4 その他のイベント
5.5 まとめ

＜＜著者紹介＞＞

渋田 達也
福岡で活動している web 系のエンジニア。サーバーレスなどのバックエンドのこともしたりするが最近はフロントエンドがメインとなっています。witter では@mya_ake という ID で Web のフロントエンド周りの話を中心につぶやいたりしてます。

＜＜販売ストア＞＞

電子書籍:
Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER
印刷書籍:
Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス
※ 各ストアでの販売は準備が整いしだい開始されます。
※ 全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/
株式会社インプレスR&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。
※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/
株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp