PWAs: Don’t build it all at oncce

One of the things that has been tempting when working on a PWA for my layout experiments has been to do everything at once and work on them in parallel.

Reading Jason Grigsby’s Progressive Web Applications book from A Book Apart reminded me that we need to take it easy, plan ahead and do incremental deployment of PWA features.

The basics

What do I mean by this? There are three technical items that we know we need to turn a site into a PWA:

- Host the site using HTTPS
- A Web Manifest
- A Service Worker

The basics are easy. We can add an SSL certificate using free tools like let’s encrypt.

The (basic) Manifest is also fairly straight forward, you can generate it manually using instructions like thos on Google developers or generate it automatically using one of many Web Manifest Generators. I suggest PWA Builder, mostly because it will help you generate both the manifest and a basic service worker that you can later tinker with.

The Service Worker is also fairly easy. One thing that we need to remember is that we don’t need to do everything the first time. PWA Builder will give you options for creating your Service Worker and the code to insert into your site’s entry page to use it.

Thinking things through

> These are not all the items worth reviewing an analyzing before you implement

your site as a PWA but it's a good starting point.

For a more thorough discussion of how to gradually roll out a PWA, check Progressive Web Applications.

There is more than the basics when it comes to PWAs and web applications in general. Yes, we can slap a manifest and a service working into a mediocre website and making into a mediocre PWA.

But there are a lot of other things we need to consider when turning our sites into applications even before we write our first line of code.

Some of these considerations have to do with our site as it exists before we implement PWA technologies.

Navigation

The first item to consider, for me, is the site's navigation. If we make the site in to a full screen application then we need to make sure that users can navigate without the browser's chrome available.

Performance matters

Another aspect to consider is how is your site performing before we implement a PWA. My favorite tool is Lighthouse available as a CLI that you can integrate into your existing workflows and as part of DevTools.

To channel my inner Alex Russell, any performance test you run must run in the devices using `chrome://inspect` to debug it remotely. The results from DevTools are good approximations but will never match the results of running in a real device.

The reason we run performance tests is to make sure we're not using the Service Worker as an excuse for slow loading content.

Service Worker: what part of the

swiss army knife do we need?

When planning a Service Worker you have to decide how hard or how easy do you want to build it. Do you want a site that automatically caches all assets on first load? Do we want a Service Worker that will pre-cache certain assets (the shell of an app or the assets needed to load an index page)? Push notifications? User activated background fetch?

The cool thing is that we can do all these things over time. The idea behind a service workers is that you can build them out over time and users will only see the new features.

We can start with a basic service worker that will cache all resources to a single cache. We can progress from there. Every time you change the service worker, browsers will treat them as a brand new worker and will update whatever needs to change.

There are also tools like Workbox.js that will automate service worker creation and usage. It makes it easier to creaate multiple caches, using different caching strategies and it gives you access to newer technologies built on top of service workers.

It also gives you more time to develop your strategy for how you will implement the worker.

Frameworks and PWAs

If you're using a framework, you can still consider evaluate PWAs Angular and React both provide PWA implementation for new apps/sites... Angular through the CLI and the `@angular/pwa` package and React through the `create-react-app` tool. In my limited research I wasn't able to figure out if this is only for new applications or if we'd be able to update an existing one to make it a PWA but if you're familiar with the tools you should be familiar with the tools and the communities where you can find additional information.

Links and resources

- Progressive Web Apps: Escaping Tabs Without Losing Our Soul
- Responsive Web Design

- Designing Responsive Progressive Web Apps
- Progressive web sites: a future that’s native to the web
- Web App Manifest
- Designer vs. Developer #3 - Adopting the Native Language of the Web
- Angular
    - Angular service worker introduction
    - Progressive Web Apps for Angular 6 and beyond
- React
    - Making a Progressive Web App
    - How to build a PWA with Create-React-App and custom service workers
- Remote Debugging
    - Android
    - iOS
- Tools
    - Workbox.js
    - PWA Builder