Christoph Weber - Director of Technical Services
Alex Borsody - Developer

[achieve]

Meet the Progressive Web App Module

See Also

Session recording

Understanding Progressive Web Apps and Why You Should Care

By Mark Shropshire
Friday, 4:00pm - 4:50pm

Module Maintainers

Theodore Biadala (nod_) - Original author

Chris Ruppel (rupl) - D7 branch

Alex Borsody (AlexBorsody) - D8 branch

Christoph Weber (ChristophWeber) - just talks

Progressive Web App Operational Definition

1. **Reliable** — Load instantly and never show an "Offline" screen to the visitor, even in uncertain network conditions.
2. **Fast** — Respond quickly to user interactions with silky smooth animations and no janky scrolling.
3. **Engaging** — Feel like a natural app on the device, with an immersive user experience.

This new level of quality allows Progressive Web Apps to earn a place on the user’s home screen.

Progressive Web App - technical parts

1. HTTPS
2. Service Worker
3. manifest.json

Why create a Progressive Web App?

- **One code base** for web site and mobile apps. - *Save cost.*
- Better user experience overall. - *Increase Engagement.*
- Bypass the app store. *Save Time and Cost.*
- Search Engine Optimization. - *Speed = SEO.*

A PWA leverages technologies based on the web that the majority of developers are familiar with, opening up the development of native feeling apps to a much wider audience.

The Building Blocks: Manifest

manifest.json


```
{
"name":"Tandem Careplanning",
"short_name":"Tandem",
"Display":"fullscreen",
"background_color":"#757eff",
"theme_color":"#89278e",
"description":"Tandem Careplanning",
"Lang":"en"
,"icons":[{"src":"\/sites\/default\/files\/pwa\/cgd-logo-512.pngcopy.png","sizes":"1
92x192","type":"image\/png"},{"src":"\/sites\/default\/files\/pwa\/cgd-logo-512.pn
g","sizes":"512x512","type":"image\/png"}],
"start_url":"\/",
"scope":"\/"
}
```

Manifest documentation

https://developers.google.com/web/fundamentals/web-app-manifest/

Service Workers

What is a Service Worker

A Service Worker is a JavaScript file that runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and (possibly) delivering push messages.

1. (Nascent) multi-threading
2. Fine grained control over network events, caching strategy, and device capabilities.

Things to note about Service Workers

- The service worker can't access the DOM directly. To communicate with the page, the service worker uses the postMessage() method to send data and a "message" event listener to receive data.
- A service worker is a programmable network proxy that lets you control how network requests from your page are handled.
- Service workers only run over HTTPS. Because service workers can intercept network requests and modify responses, "man-in-the-middle" attacks could be very bad.
- The service worker becomes idle when not in use and restarts when it's next needed. You cannot rely on a global state persisting between events. If there is information that you need to persist and reuse across restarts, you can use IndexedDB databases.
- Service workers make extensive use of promises.

Introducing:
The pwa module

pwa module

The Drupal 7 module is very much complete with a serviceworker.js file that serves pages saved in the Cache API from configs stored in Drupal, there are also configs exclude pages from cache in the Drupal admin serviceworker.js config panel.

The Drupal 8 version now mirrors and surpasses the Drupal 7 version. In particular, https://www.drupal.org/project/pwa/issues/3066848 adds better iOS support.

What does the pwa module do?

The main benefit of this module is the use of Service Worker for caching and offline capabilities. Once the Service Worker is active, page loading is faster:

- All JS and CSS files will always be served from cache while being refreshed in the background.
- All pages are fetched from the network (as before) and a copy is kept in cache so it will be available when offline.
- Images are cached unless the save-data header is detected in order to be mindful of bandwidth usage and cache size. A fallback image should appear for any uncached image.

What else does the pwa module do?

The module will also create a configurable `manifest.json` file to make the website installable on supporting mobile devices. Out of the box, the module fulfils enough PWA requirements that the "add to home screen" prompt is automatically triggered when a visitor returns often enough to your website (not supported by iOS right now).

It provides a perfect PWA Lighthouse audit score by default as well.

A Deep Dive into Precaching

Knowing an array's contents ahead of time is a tall order.

We prepopulate the Service Worker with known assets by internally requesting the URLs you whitelist and extract assets out of the DOM, then assemble them in the pre-cache automatically.

That means the `install` event should have a reasonable chance of caching a usable page including CSS and JS, while only requiring a site admin to specify a list of URLs they want cached like so:

```
/
/about
/offline
```

https://chrisruppel.com/blog/progressive-web-app-drupal/

The `fetch` Listener

If the pre-caching doesn't cover everything, the visitor's next few page loads will continue populating the cache using `staleWhileRevalidate` strategy, storing the latest copy of a file each time it's fetched.

Both the install and fetch listener use the `no-cors` mode to fetch assets, making it friendly to third-party requests. This means it can cache assets on your primary domain in addition to CDNs like Google Fonts.

Additionally, the `fetch` listener checks for the `Save-Data` header, and avoids caching images when it is present.

https://chrisruppel.com/blog/progressive-web-app-drupal/

Server Requirements

The pwa Drupal module requires **PHP 7.2** or greater. We will continue to modify this minimum requirement to avoid supporting any EOL version of PHP.

Your web server **must** support secure connections using **HTTPS**. This is a requirement of the W3 specification and is not a choice made by the module maintainers.

Browser Support

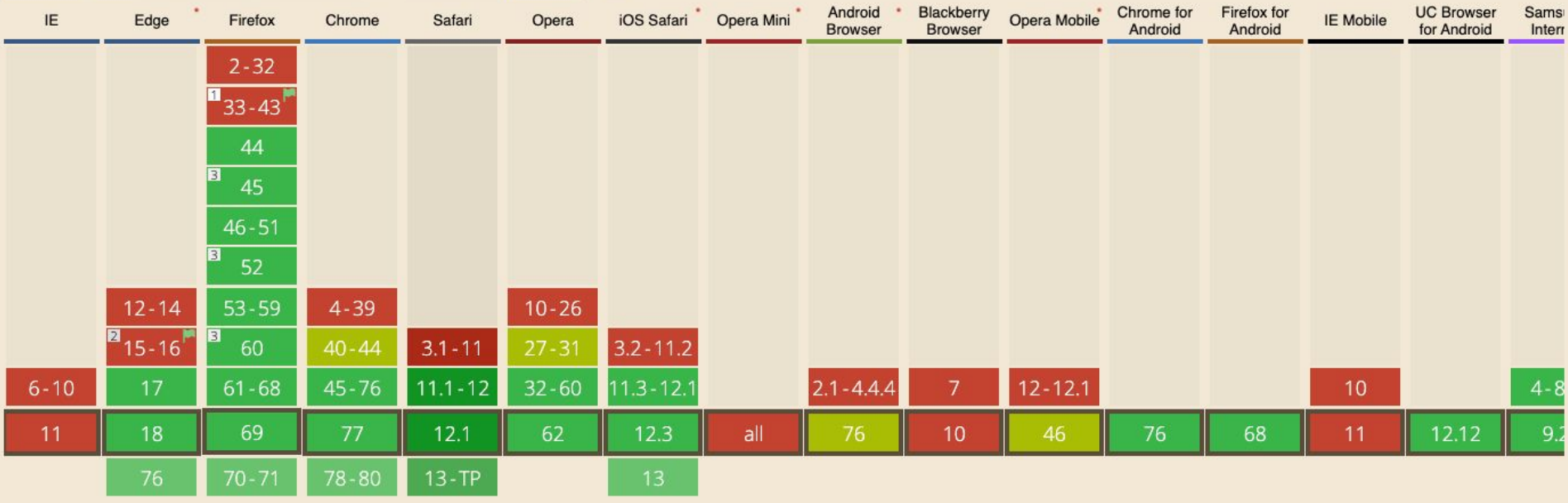
Service Workers - WD

Usage: Global

% of all users: 92.09% + 0.39% = 92.48%

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?



Browser Support Gotchas

Safari (desktop and mobile) does not support push notifications. Nor does the pwa module.

Safari (desktop and mobile) does not completely honor settings in manifest.json. The D8 module has a patch in the issue queue to add metatags to better support Apple devices.
https://www.drupal.org/project/pwa/issues/3066848

Safari (desktop and mobile) does not support all display options in manifest.json. standalone -> fullscreen, minimal-ui -> browser

Debugging Service Workers

Chrome:

https://developers.google.com/web/fundamentals/codelabs/debugging-service-workers/

Mobile Safari:

Service Workers on iOS devices are nearly unmanageable. Once you test on your iPhone you may get persistent issues from lingering obsolete Service Worker code. Be warned!

Android:

Same as Chrome

What's Next?

Broadcast cache update:
Display cached page (might be stale)
Get updated content from server, and broadcast available updated content to page.

For example:
Display cached page, play a small spinner.
If broadcast of updated content arrives, remove spinner and automatically refresh the page.
Else, stop spinner after a 2s or so.

Thank you