Progressive Web "Apps" with 11ty

@JeffPosnick • JAMstack NYC • Jan. 14, 2020
bit.ly/11ty-pwa-slides

or...

What if 11ty,
but in a service worker?

Some terminology

Progressive Web Apps

web.dev/installable

- Websites that meets a baseline for installability.
- Requires a web app manifest and a service worker.

Web App Manifests

web.dev/add-manifest

- Metadata in JSON format.
- Title, icons, splash screen, colors, and more.

Service Workers

web.dev/reliable

- An event-driven JavaScript network proxy deployed alongside your site.
- Has access to the Cache Storage API, and can respond to HTTP requests programmatically.

Cache Storage API

web.dev/service-workers-cache-storage

- Different from the HTTP cache.
- Primitives are `Request` & `Response` objects.

Workbox

web.dev/workbox

- Build tools for generating service workers.
- Runtime interfaces for caching, routing, and more.

11ty

www.11ty.dev

- *Eleventy is a simpler static site generator.*
- JavaScript-based alternative to Jekyll.
- Rich templating language support.

Nunjucks

mozilla.github.io/nunjucks

- A templating system with browser & Node.js support.
- Syntax is close enough to Liquid for those migrating from Jekyll.

PWAs' Superpower:

Loading that's reliable, fast, and reliably fast.

How?
Cache-first HTML.

What's with the "App" in PWA?

Single-page app architecture works well with cache-first HTML.

App Shell Model

bit.ly/app-shell-loading

- "Generic" HTML is cached and used for all navigations.
- Client-side JavaScript renders page-specific content.
- `gatsby-plugin-offline` and others use this approach.

So what about static sites?

One approach: cache everything *

** Not recommended*

Runtime caching

bit.ly/11ty-runtime-caching

- Pros: repeat visits are cache-first, reliable while offline.
- Cons: first visits are still network-first, broken update model, storage bloat.

Precaching

bit.ly/11ty-precaching

- Pros: initial visits are cache-first, entire site is reliable while offline.
- Cons: "all or nothing" updates, upfront download cost, storage bloat.

A different approach

11ty in your service worker

bit.ly/11ty-sw-rendering

Export post-specific 11ty data as JSON

```
---
pagination:
  data: collections.post
  size: 1
  alias: post
permalink: /_posts/{{ post.fileSlug }}.json
---

{
  "date": "{{ post.date.toISOString() }}",
  "excerpt": {{ post.data.excerpt | dump | safe }},
  "html": {{ post.templateContent | dump | safe }},
  "layout": "{{ post.data.layout }}",
  "tags": {{ post.data.tags | dump | safe }},
  "title": {{ post.data.title | dump | safe}}
}
```

Get Nunjucks to work with Cache Storage API

```
import nunjucks from 'nunjucks/browser/nunjucks';

const CacheStorageLoader = nunjucks.Loader.extend({
  async: true,

  getSource: async function(name: string, callback: Function) {
    try {
      const path = `/_posts/_includes/${name}`;
      const cachedResponse = await matchPrecache(path);
      if (!cachedResponse) {
        throw new Error(`Unable to find precached response for ${path}.`);
      }
      const src = await cachedResponse.text();
      callback(null, {src, path, noCache: false});
    } catch(error) {
      callback(error);
    }
  }
});
```

Tell the service worker how to load page context

```
const postHandler = async (options: RouteHandlerCallbackOptions) => {
  const {params} = options;
  const site = await getSiteData();
  const cacheKey = `/_posts/${params[3]}.json`;
  const cachedResponse = await matchPrecache(cacheKey);
  const context = await cachedResponse.json();
  context.site = site;
  context.content = context.html;

  // Pass content to Nunjucks
});
```

Tell the service worker about 11ty's URLs

```
registerRoute(
  new RegExp('/(\\d{4})/(\\d{2})/(\\d{2})/(.+)\\.html'),
  postHandler
);
```

Service Worker Templating

bit.ly/11ty-sw-templating

- Pros: initial visits are cache-first, entire site is reliable while offline, updates are cheap and apply globally, efficient storage usage.
- Cons: Upfront download cost.

TODOs & Enhancements

- Package things up as a starter kit.
- Support for 11ty shortcodes, custom tags, etc.
- Dynamic rendering of top-level `index.html`
- Lightweight Nunjucks alternatives—`lit-html`?

Additional Resources

- jeffy.info
- bit.ly/static-site-scaffold
- bit.ly/pwa-architecture
- bit.ly/sw-html-payloads

Progressive Web "Apps" with 11ty

@JeffPosnick • JAMstack NYC • Jan. 14, 2020
bit.ly/11ty-pwa-slides