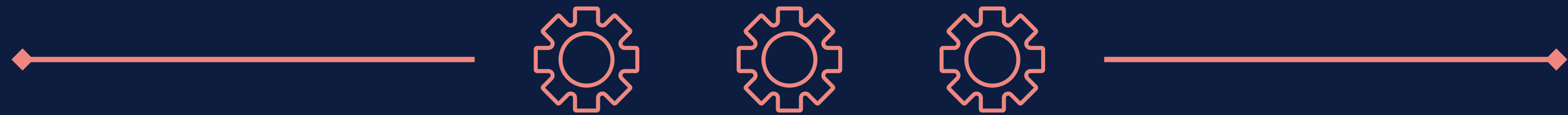


# NOW THAT'S WHAT I CALL SERVICE WORKER

Jeremy Wagner — <https://jlwagner.net/>

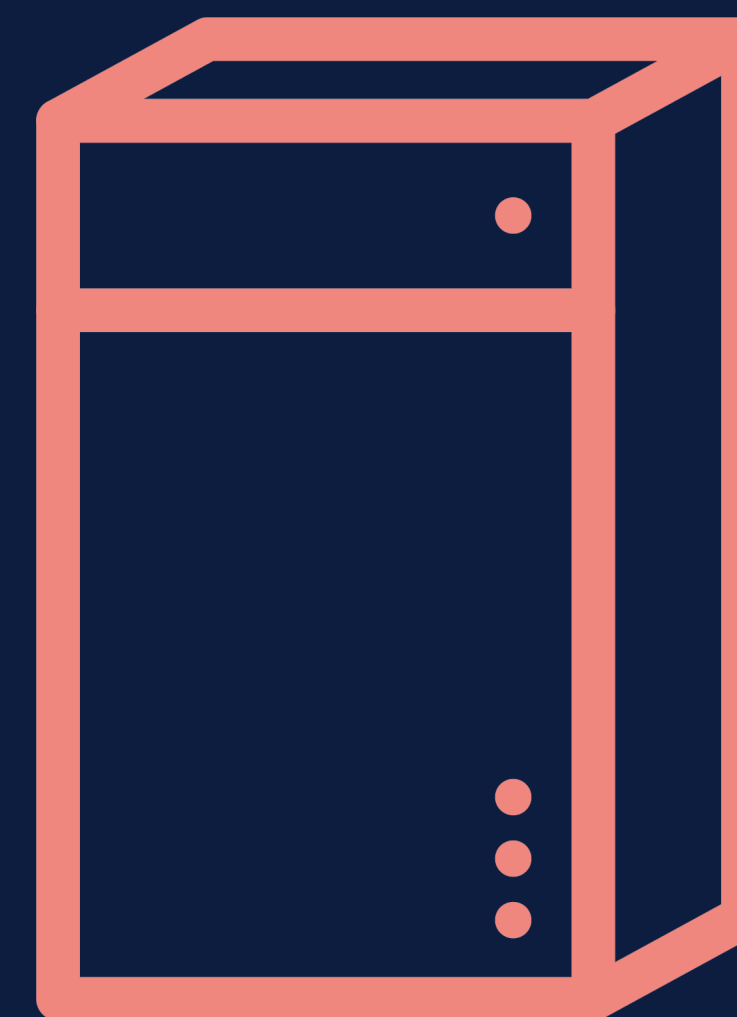
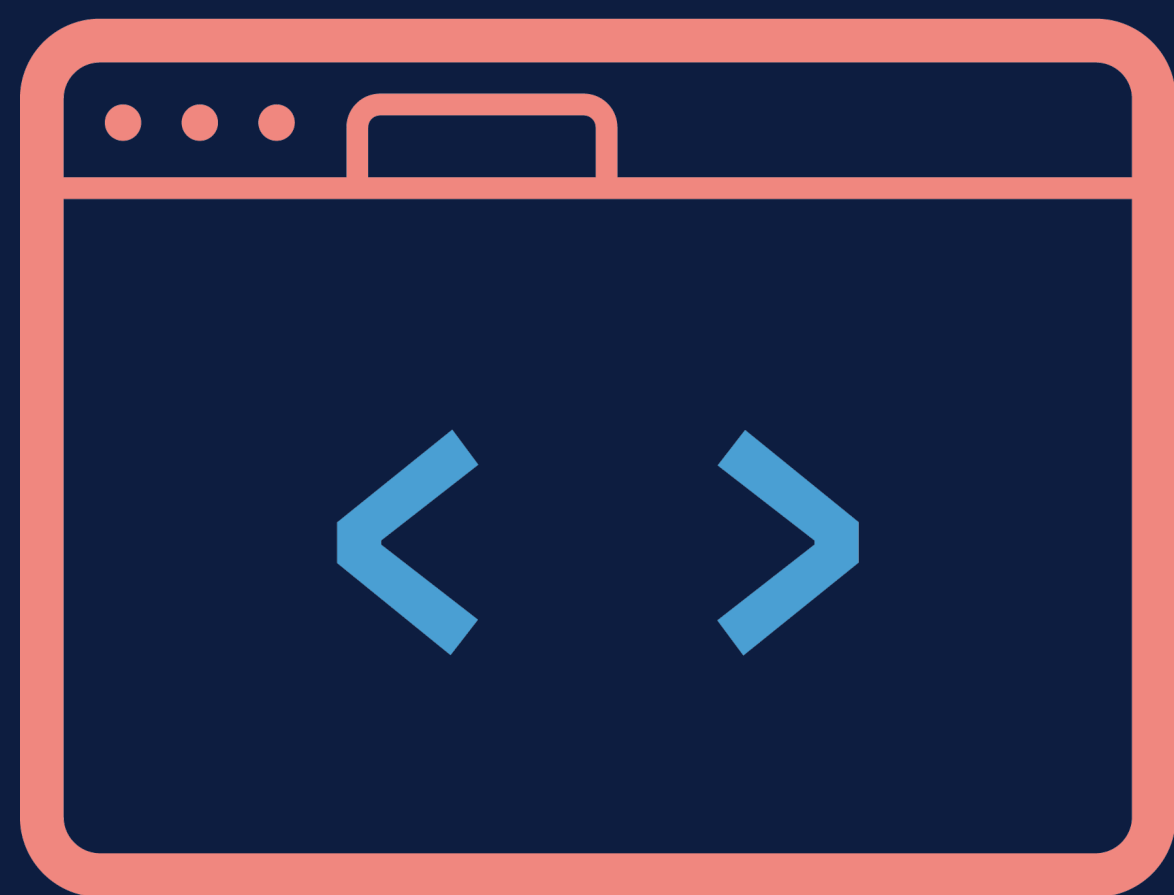
NDC Porto — Porto, Portugal — April 2022

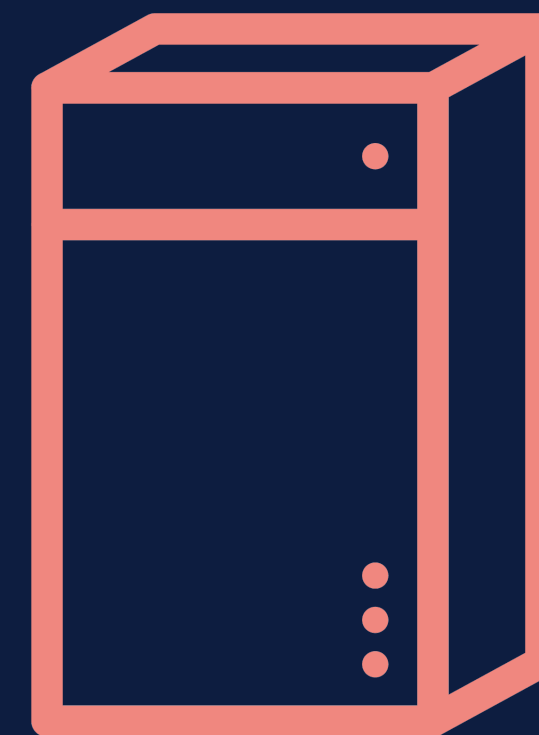
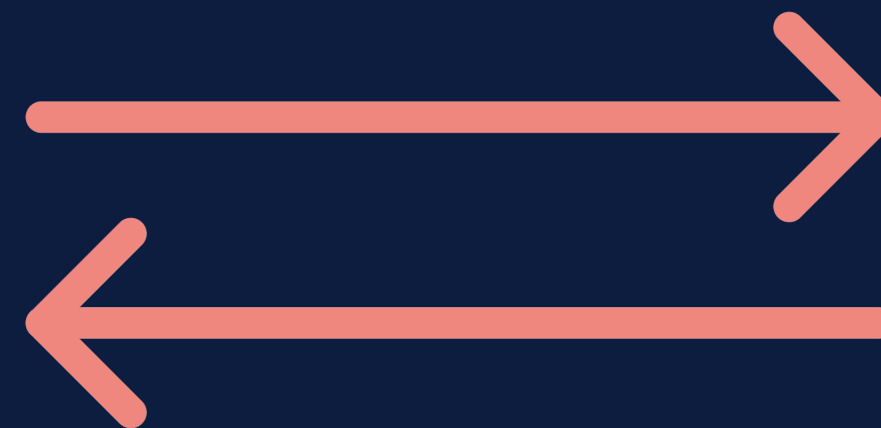
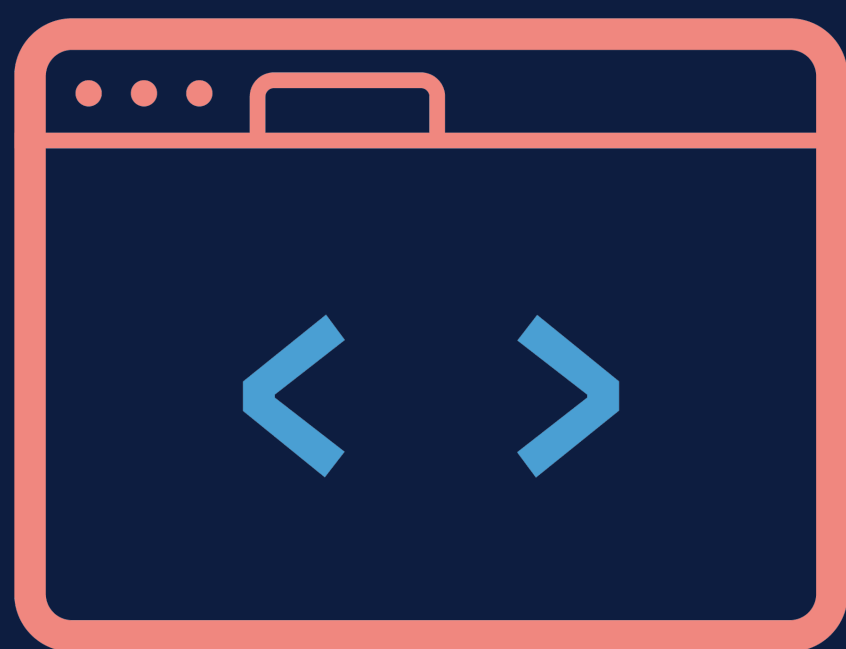


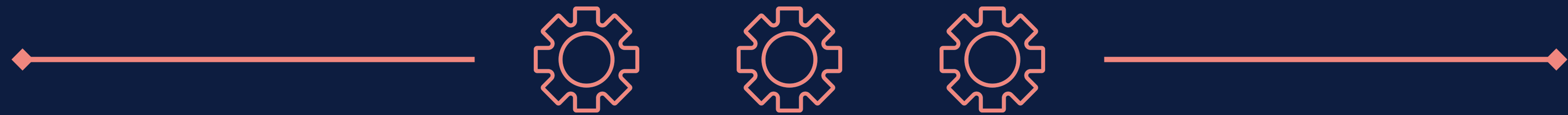


*A quick service worker primer.*

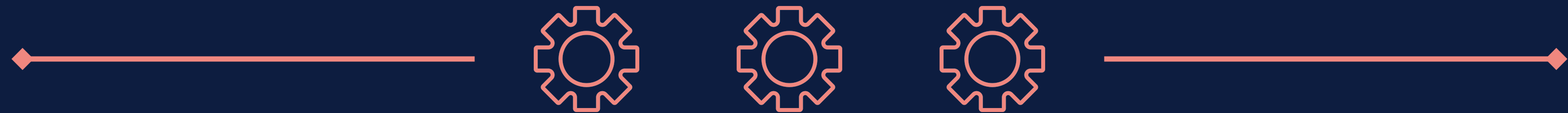








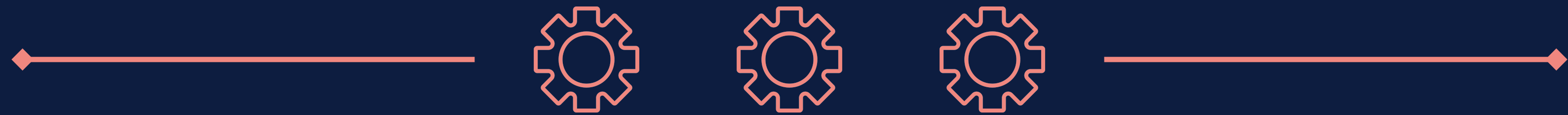
*A tale of two navigation patterns.*





*Streams!*



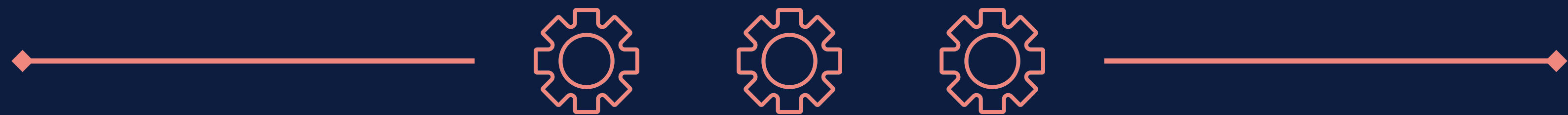


*Where are you going with this?*



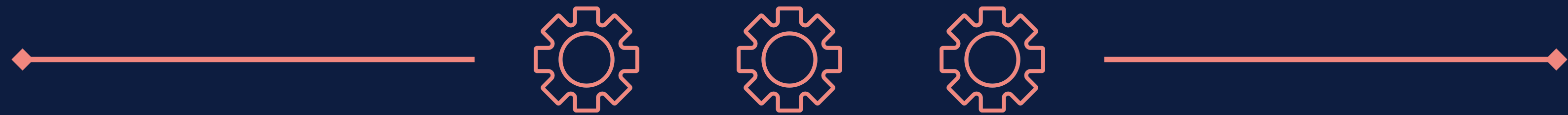






*Streams! Again!*





*The anatomy of a website.*



Want to get in touch? That's cool — just so long as you're not looking to recruit me for a new job, asking me to write for your blog, asking to write on my blog (hey, I've had it happen), or sending me spam. Otherwise, the contact form below should get your message to me in no time.

**Name?**

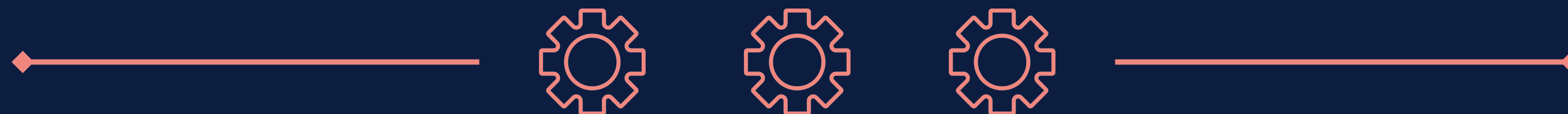
**Email?**

**What's on your mind?**

**Send message**



*Precache!*



```
npm install \
workbox-navigation-preload \
workbox-strategies \
workbox-routing \
workbox-precaching \
workbox-streams \
--save
```

```
// ./src/sw.js
import * as navigationPreload from "workbox-navigation-preload";
import { CacheFirst, NetworkFirst, StrategyHandler } from "workbox-strategies";
import { registerRoute, Route } from "workbox-routing";
import { matchPrecache, precacheAndRoute } from "workbox-precaching";
import { strategy as composeStrategies } from "workbox-streams";
```

```
// ./src/sw.js
import * as navigationPreload from "workbox-navigation-preload";
import { CacheFirst, NetworkFirst, StrategyHandler } from "workbox-strategies";
import { registerRoute, Route } from "workbox-routing";
import { matchPrecache, precacheAndRoute } from "workbox-precaching";
import { strategy as composeStrategies } from "workbox-streams";

// Enable navigation preload
navigationPreload.enable();
```

Speed up service worker with na


+

8

← →

https://developer.chrome.com/blog/navigation-preload/

☰ ☆ 📧 >> ≡

 Chrome Developers

🔍 Search docs, blogs and more

🏠  
Home

📁  
Docs

💬  
Blog


Take a stroll down memory lane and celebrate #100CoolWebMoments since Chrome's first release.  
[Discover](#) [Dismiss](#)

Blog

🔗

# Speed up service worker with navigation preloads

Published on Wednesday, February 15, 2017 • Updated on Friday, September 7, 2018

 Jake Archibald  
Human boy working on web standards at Google

## # TL;DR

- In some situations, [service worker boot-up time can delay a network response](#).
- A new feature, [navigation preload](#), released in Chrome 59, fixes this by allowing you to make the request in parallel with service worker boot-up.
- You can distinguish preload requests from regular navigations using a header,

Table of contents

[TL;DR](#)  
[The problem](#)  
["Navigation preload" to the rescue](#)  
[Activating navigation preload](#)  
[Using the preloaded response](#)  
[Custom responses for preloads](#)  
[Changing the header](#)  
[Getting the state](#)

We serve cookies on this site to analyze traffic, remember your preferences, and optimize your experience. [More details](#)

Ok, Got it.

<https://developer.chrome.com/blog/navigation-preload/>

SW boot

Navigation request

SW boot

Navigation request

```
// ./src/sw.js
import * as navigationPreload from "workbox-navigation-preload";
import { CacheFirst, NetworkFirst, StrategyHandler } from "workbox-strategies";
import { registerRoute, Route } from "workbox-routing";
import { matchPrecache, precacheAndRoute } from "workbox-precaching";
import { strategy as composeStrategies } from "workbox-streams";

// Enable navigation preload
navigationPreload.enable();
```

```
// ./src/sw.js
import * as navigationPreload from "workbox-navigation-preload";
import { CacheFirst, NetworkFirst, StrategyHandler } from "workbox-strategies";
import { registerRoute, Route } from "workbox-routing";
import { matchPrecache, precacheAndRoute } from "workbox-precaching";
import { strategy as composeStrategies } from "workbox-streams";

// Enable navigation preload
navigationPreload.enable();

// Establish cache names
const runtimeCacheName = "jlwagnernet-runtime";
const contentCachename = "jlwagnernet-content";
```

```
// Prior code omitted...
```

```
// ...
```

```
precacheAndRoute([
  {
    url: "/partial-header.php",
    revision: __PARTIAL_HEADER_HASH__
  },
  {
    url: "/partial-footer.php",
    revision: __PARTIAL_FOOTER_HASH__
  },
  {
    url: "/offline/index.php",
    revision: __OFFLINE_FALLBACK_HASH__
  },
  ...self.__WB_MANIFEST
]);
```

```
// Prior code omitted...
```

```
// ...
```

```
precacheAndRoute([
  {
    url: "/partial-header.php",
    revision: __PARTIAL_HEADER_HASH__
  },
  {
    url: "/partial-footer.php",
    revision: __PARTIAL_FOOTER_HASH__
  },
  {
    url: "/offline/index.php",
    revision: __OFFLINE_FALLBACK_HASH__
  },
  ...self.__WB_MANIFEST
]);
```

```
// Prior code omitted...
// ...

const contentStrategy = new NetworkFirst({
  cacheName: contentCacheName,
  plugins: [
    {
      requestWillFetch: ({ request }) => {
        const headers = new Headers();

        headers.append("X-Content-Mode", "partial");

        return new Request(request.url, {
          method: "GET",
          cache: "default",
          headers
        });
      },
      // Error handler omitted...
    }
  ]
});
```

```
// Prior code omitted...
// ...

const contentStrategy = new NetworkFirst({
  cacheName: contentCacheName,
  plugins: [
    {
      requestWillFetch: ({ request }) => {
        const headers = new Headers();

        headers.append("X-Content-Mode", "partial");

        return new Request(request.url, {
          method: "GET",
          cache: "default",
          headers
        });
      },
      // Error handler omitted...
    }
  ]
});
```

```
// Prior code omitted...
// ...

const contentStrategy = new NetworkFirst({
  cacheName: contentCacheName,
  plugins: [
    {
      requestWillFetch: ({ request }) => {
        const headers = new Headers();

        headers.append("X-Content-Mode", "partial");

        return new Request(request.url, {
          method: "GET",
          cache: "default",
          headers
        });
      },
      // Error handler omitted...
    }
  ]
});
```

```
<?php
// Get all includes
require_once($_SERVER["DOCUMENT_ROOT"] . "/includes/utils.php");
require_once($_SERVER["DOCUMENT_ROOT"] . "/includes/site-header.php");
require_once("./content.php");
require_once($_SERVER["DOCUMENT_ROOT"] . "/includes/site-footer.php");

// Check if navigation reload is enabled
$isPartial = isPartial();

// Get assets
$assets = getAssets();

// Includes
if ($isPartial === false) {
    siteHeader($assets, $isPartial);
}

content($isPartial);

if ($isPartial === false) {
    siteFooter($assets);
}
?>
```

```
<?php
function isPartial () {
    return (
        isset($_SERVER["HTTP_SERVICE_WORKER_NAVIGATION_PRELOAD"]) &&
        $_SERVER["HTTP_SERVICE_WORKER_NAVIGATION_PRELOAD"] === "true"
    ) || (
        isset($_SERVER["HTTP_X_CONTENT_MODE"]) &&
        $_SERVER["HTTP_X_CONTENT_MODE"] === "partial"
    );
}
?>
```

```
// Prior code omitted...
// ...

const contentStrategy = new NetworkFirst({
  cacheName: contentCacheName,
  plugins: [
    {
      // requestWillFetch handler omitted...
      handlerDidError: async ({ request }) => {
        const cacheMatch = await caches.match(request.url, {
          cacheName: contentCacheName
        });

        if (cacheMatch === undefined) {
          return await matchPrecache("/offline/index.php");
        }

        return cacheMatch;
      }
    }
  ]
});
```

Oh no. It looks like you're offline, and that sucks. While you can't load any other pages on this website — or any other website for that matter — you *are* able to visit these pages you've already been to:

## *Contact*

14 JANUARY, 2022

[JLWAGNER.NET](#)

Want to get in touch with me? I guess you could do that here.

## *I went and wrote a book*

27 OCTOBER, 2021

[JLWAGNER.NET](#)

I wrote Responsible JavaScript for A Book Apart. I hope you'll consider buying it, and if you do, I hope you like it.

## *Welcome*

14 JANUARY, 2022

[JLWAGNER.NET](#)

The home page of technical writer, author, and speaker Jeremy Wagner.

Access caches from the window

+

8

←

→

🔒

https://developer.chrome.com/docs/workbox/access-caches-from-the-window/


📄

☆

🔖

»

☰

 Chrome Developers

🔍

Search docs, blogs and more

🏠  
Home

📁  
Docs

💬  
Blog

Documentation > Workbox > Use cases and recipes

▶ Introduction to Workbox and service workers

▶ What you need to know

▶ Ways to use Workbox

▼ Use cases and recipes

Using workbox-window

Caching resources during runtime

Forcing a network timeout

Access caches from the window

Serving cached audio and video

Managing fallback responses

Handling service worker updates with immediacy

Retrying requests when back online

Using plugins

Workbox Modules

API Reference

Migration Guides

Workbox on GitHub

🔗

# Access caches from the window

Published on Tuesday, December 7, 2021

With all of this work we've been doing outside of the `window`, you might think that `Cache` instances can only be accessed in the service worker scope. The fact is that you can access `Cache` instances in *both* the service worker scope *and* in your web app's traditional code, running in the `window`. This makes it easier for the user to directly interact with a service worker cache, or update the user interface based on cache state.

One potential use case is to offer a "save for offline" feature for pages the user may want to read later, but know they may be offline at that time. The Glitch embed below shows how to do this with Workbox.

## Tears in Rain

*"I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched C-beams*

save-for-offline-test

Share

View Source

🔗

🐙

<https://developer.chrome.com/docs/workbox/access-caches-from-the-window/>

```
// Prior code omitted...  
// ...
```

```
const navigationHandler = composeStrategies([  
  () => matchPrecache("/partial-header.php"),  
  ({ event }) => contentStrategy.handle(event),  
  () => matchPrecache("/partial-footer.php")  
]);
```

```
// Prior code omitted...
```

```
// ...
```

```
registerRoute(({request}) => request.mode === 'navigate', navigationHandler);
```

```
// Prior code omitted...
```

```
// ...
```

```
const staticAssets = /\.(m?js|css|webp|avif|svg|jpe?g|png|gif|ico)$/i;
```

```
const staticRoute = new Route(({ url }) => {
```

```
  return staticAssets.test(url);
```

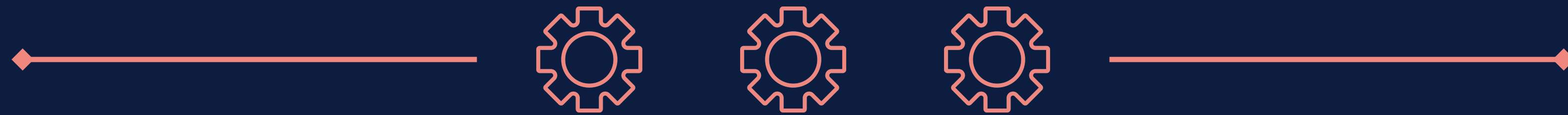
```
}, new CacheFirst({
```

```
  cacheName: runtimeCacheName
```

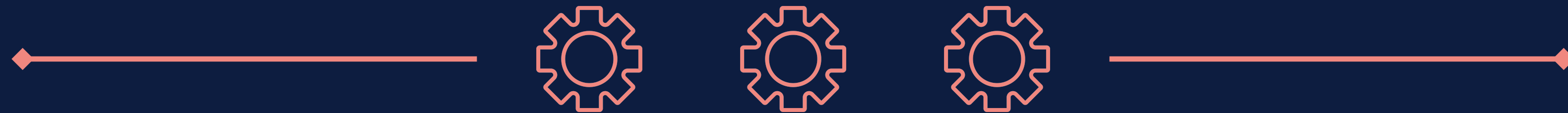
```
})));
```

```
registerRoute(staticRoute);
```

```
// That's all!
```



*Is this all worth it?*

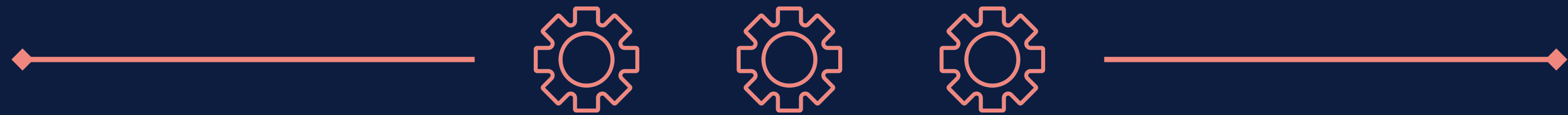




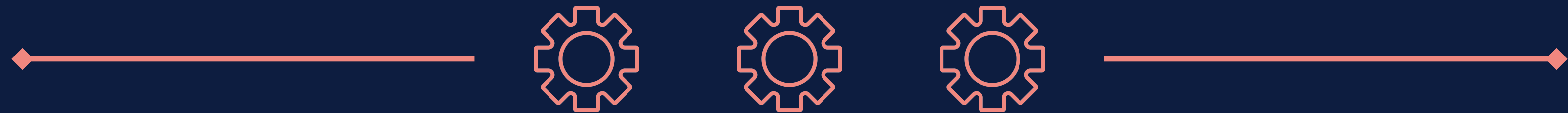
*No service worker.*



*With service worker.*



*What's the field data story?*



GitHub - GoogleChrome/web-vitals

150% ☆ >> ≡

Product Team Enterprise Explore Marketplace Pricing

Search / Sign in Sign up

GoogleChrome / web-vitals Public

Notifications Fork 218 Star 4.7k

<> Code

Issues 20

Pull requests 3

Actions

Projects

Wiki

Security

Insights

main 3 branches 20 tags

Go to file Code

About

philipwalton Merge pull request #205 from skychx/main ed70ed4 on Feb 27 198 commits

docs	Initial commit	2 years ago
src	Prevent TTFB from reporting after bfcache restore	3 months ago
test	Prevent TTFB from reporting after bfcache restore	3 months ago
.eslintrc	Update dev dependencies	3 months ago
.gitignore	Update dependencies	13 months ago
CHANGELOG.md	Update CHANGELOG	3 months ago
LICENSE	Update license	2 years ago
README.md	chore: fix readme typo	2 months ago
base.d.ts	Fix missing type declaration files	17 months ago

Essential metrics for a healthy site.

[web.dev/vitals](#)

Readme

Apache-2.0 License

Code of conduct

4.7k stars

83 watching

218 forks

Releases

20 tags

<https://developer.chrome.com/blog/navigation-preload/>

◀———— *FCP* ———▶

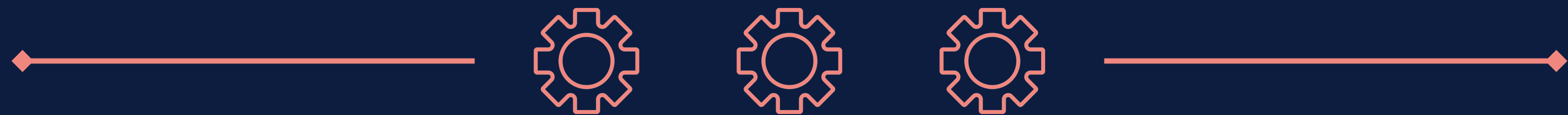
◀———— *90th percentile* ———▶

◀———— *LCP* ———▶

◀———— *90th percentile* ———▶

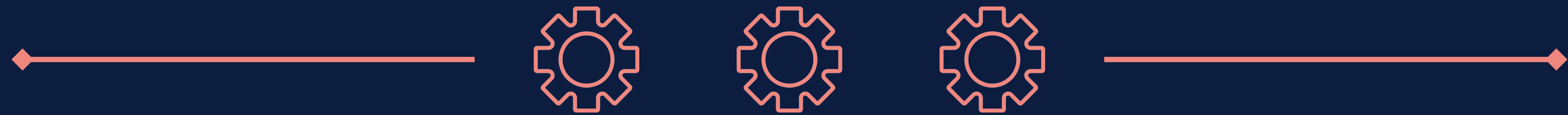
◀ *TTFB* ▶

◀ *90th percentile* ▶



*Caveats.*





*Every page is entitled to a title.*



```
<title>
```

```
Contact &mdash; jlwagner.net
```

```
</title>
```

```
<title>
```

```
<?php
```

```
if ($isPartial === false) {
```

```
    echo $title . "-jlwagner.net";
```

```
}
```

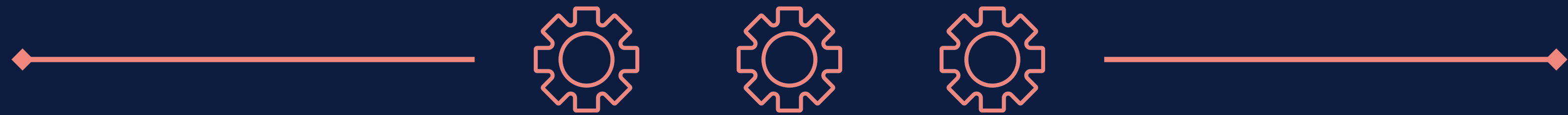
```
?>
```

```
</title>
```

```
<?php
if ($isPartial === true) {
    ?>
    <script>
        document.addEventListener("DOMContentLoaded", () => {
            const { title } = JSON.parse(document.getElementById("page-data").textContent);

            document.title = `${title}-jlwagner.net`;
        });
    </script>
    <?php
}
?>
```

```
<script type="application/json" id="page-data">
  {
    "title": "Contact"
  }
</script>
```



*Spinners gonna spin.*



jlwagner.net

Talks — Contact — Settings

---

```
/* This rule assumes your header partial  
   ends with an open <main> element. */  
main:empty {  
    content: "Loading...";  
}
```

```
/* This rule assumes your header partial  
   ends with an open <main> element. */  
main:empty {  
    content: "Loading...";  
}
```

Can I use

? ⚙ Settings

✕ Feature: NetworkInformation API: effectiveType

⌵

NetworkInformation API: effectiveType

Usage

% of all users ⌵ ?

• Africa

76.36%

Global

72.08%

Current aligned

Usage relative

Date relative

Filtered

All ⚙

Chrome for Android	Chrome	Safari on iOS*	Opera Mini*	Firefox	Samsung Internet	Edge*	UC Browser for Android	IE	Safari	Opera	Firefox for Android	Android Browser
	4-60					12-18				10-47		
	61-99	3.2-15.3		2-98	4-15.0	79-99		6-10	3.1-15.3	48-82		2.1-4
100	100	15.4	all	99	16.0	100	12.12	11	15.4	83	99	100
	101-103			100-101					TP			

```
<script>
```

```
    const effectiveType = navigator?.connection?.effectiveType;
```

```
    if (effectiveType !== "4g") {
```

```
        document.documentElement.classList.add("slow");
```

```
    }
```

```
</script>
```

```
<div id="loading" aria-label="Loading page...">  
  <div class="dot" aria-hidden></div>  
  <div class="dot" aria-hidden></div>  
  <div class="dot" aria-hidden></div>  
</div>
```

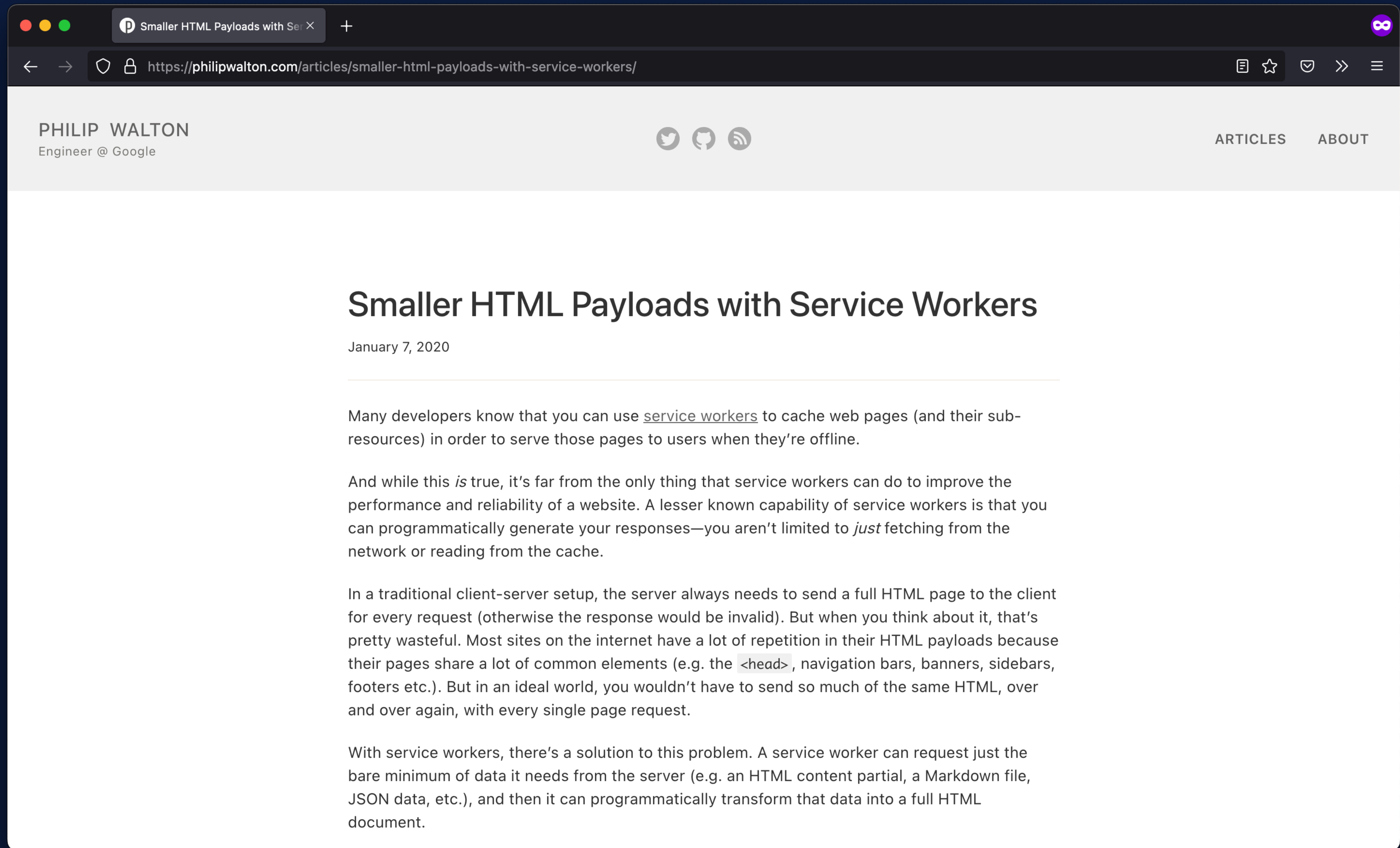
```
/* Don't show the spinner initially. */  
#loading {  
    display: none;  
    height: 2rem;  
}  
  
/* Loading animation styles and other  
    particulars omitted for brevity... */  
  
/* Show the spinner if stuff's slow. */  
.slow #loading {  
    justify-content: center;  
    display: flex;  
    align-items: center;  
    align-content: center;  
}
```

jlwagner.net

Talks — Contact — Settings

---





<https://philipwalton.com/articles/smaller-html-payloads-with-service-workers/>

Beyond SPAs - alternative archi X

https://developer.chrome.com/blog/beyond-spa/

Chrome Developers

Search docs, blogs and more

Home

Docs

Blog

Blog

# Beyond SPAs - alternative architectures for your PWA

Published on Wednesday, May 23, 2018 • Updated on Wednesday, May 23, 2018

Jeff Posnick  
Web DevRel at Google

Prefer a video to an article? You can watch the presentation on which this was based instead:

Beyond single-page apps: alternative architectures for yo...

Watch later

Share

Beyond single

Table of contents

- Let's talk about... architecture?
- Stack Overflow PWA
- Multi-page Apps (MPAs)
- Reliably fast
- Enabling Technologies: Service Workers + Cache Storage API
- "Isomorphic" JavaScript
- The server
  - Routing
  - Server-side templating
  - Templating language
- The service worker
  - Workbox
  - Routing
  - Static asset caching
  - Streaming
  - Runtime caching
  - Sharing code keeps things in sync
- Dynamic, progressive enhancements
  - Page metadata
  - Offline UX
- Common pitfalls

<https://developer.chrome.com/blog/beyond-spa/>

Now THAT'S What I Call Service Worker

← → https://alistapart.com/article/now-thats-what-i-call-service-worker/

ARTICLES EVENTS TOPICS WRITE FOR US LANGUAGE (ENGLISH)

# A LIST APART

## Now THAT'S What I Call Service Worker!

by [Jeremy Wagner](#) · March 18, 2021

Published in [Browsers](#), [Code](#), [JavaScript](#), [User Experience](#)

The [Service Worker](#) API is the [Dremel](#) of the web platform. It offers incredibly broad utility while also yielding resiliency and better performance. If you've not used Service Worker yet—and you couldn't be blamed if so, as [it hasn't seen wide adoption as of 2020](#)—it goes something like this:

1. On the initial visit to a website, the browser [registers](#) what amounts to a client-side proxy powered by [a comparably paltry amount of JavaScript](#) that—like a Web Worker—runs on its own thread.
2. After the Service Worker's registration, you can intercept requests and decide how to respond to them in the [Service Worker's fetch\(\)](#) event.

What you decide to do with requests you intercept is a) your call and b) depends on your website. You can [rewrite requests](#), [precache static assets](#) during install, [provide](#)

[offline functionality](#), and—as will be our eventual focus—[deliver smaller HTML](#)

Northwestern  
INFORMATION DESIGN AND STRATEGY  
School of Professional Studies

Northwestern's Online MS in Information Design and Strategy. Choose from tracks in content strategy, data science and analytics, and learning design.

Share this:

Become a patron

This site is monetized using Coil. If you enjoy the content, consider supporting us by signing up for a Coil Membership. Here's how...

Get Coil to access

<https://alistapart.com/article/now-thats-what-i-call-service-worker/>



*Thank you.*

