

Optimizing for  
**return visits**

# Caching Headers...

The old way.

Expires: Wed, 21 Oct 2015 07:28:00 GMT

# The Modern Way: Cache-Control

**# Apache:**

```
<FilesMatch "\.(css|js)$">
Header set Cache-Control "max-age=604800, public"
</FilesMatch>
```

**# PHP:**

```
<?php
  header("Cache-Control: max-age=604800, public");
?>
```

## Cache request directives

Standard `Cache-Control` directives that can be used by the client in an HTTP request.

```
Cache-Control: max-age=<seconds>
Cache-Control: max-stale[=<seconds>]
Cache-Control: min-fresh=<seconds>
Cache-Control: no-cache
Cache-Control: no-store
Cache-Control: no-transform
Cache-Control: only-if-cached
```

## Cache response directives

Standard `Cache-Control` directives that can be used by the server in an HTTP response.

```
Cache-Control: must-revalidate
Cache-Control: no-cache
Cache-Control: no-store
Cache-Control: no-transform
Cache-Control: public
Cache-Control: private
Cache-Control: proxy-revalidate
Cache-Control: max-age=<seconds>
Cache-Control: s-maxage=<seconds>
```

# Asset changes often? No-cache

**Cache-Control: no-cache**

# Asset won't change soon/ever?

**Set a long cache-control**

Cache-Control: public, max-age=2628000

# Asset won't change soon/ever?

**Set a long cache-control**

**Cache-Control: public, max-age=604800, immutable**

# Break immutable w/ a filename

**Last resort, version your filename when it changes**  
path/to/site.123456.js

# In Devtools...

The screenshot shows a browser's developer tools interface, specifically the Network tab and the Headers panel.

**Network Tab:** A table listing network requests. The columns include: Status (S), Method (M), Domain (Do...), File, Cache (Ca...), Time (T), Transfer (Tr...), Size (S), and Duration (0 ms). Requests listed include:

- 200 OK /server-side-... doc... ht 20.... 72 74 ms
- 200 OK bbpress.css?... styl... cs 5.3... 29 161 ms
- 200 OK search-widg... styl... cs 777... 1.2 163 ms
- 200 OK social-logos.... styl... cs 18.... 26 236 ms
- 304 Not Modified style.css?v=1... styl... cs cac... 12 160 ms
- 200 OK style.min.css... styl... cs 5.0... 28 161 ms
- 200 OK jquery-3.3.1 scri... ie 20 RA 297 ms

**Headers Panel:** A detailed view of the selected request (bbpress.css). It includes tabs for Headers, Cookies, Params, Response, Timings, and Security. The Headers tab is active, showing the following response headers:

- cache-control: max-age=315360000
- content-encoding: gzip
- content-type: text/css
- date: Wed, 05 Jun 2019 21:09:23 GMT
- etag: W/"55aef1e8-7486"
- expires: Thu, 31 Dec 2037 23:55:55 GMT



4 March, 2019

## Cache-Control for Civilians

### Table of Contents

The best request is the one that never happens: in the fight for fast websites, avoiding the network is far better than hitting the network at all. To this end, having a solid caching strategy can make all the difference for your visitors.

 *How is your knowledge of caching and Cache-Control headers?*  
— Harry Roberts (@csswizardry) **3 March, 2019**

That being said, more and more often in my work I see lots of opportunities being left on the table through unconsidered or even completely overlooked caching practices. Perhaps it's down to the heavy focus on first-time visits, or perhaps it's a simple lack of awareness and knowledge? Whatever it is, let's have a bit of a refresher.



Hi there, I'm Harry. I am an **award-winning Consultant Web Performance Engineer, designer, developer, writer, and speaker** from the UK. I **write, Tweet, speak, and share code** about measuring and improving site-speed. You **should hire me**.

Follow [@csswizardry](#)

I am accepting new projects for  
Q2 2020

## Cache-Control

Cache-control is a header that tells a browser how to handle cached responses.

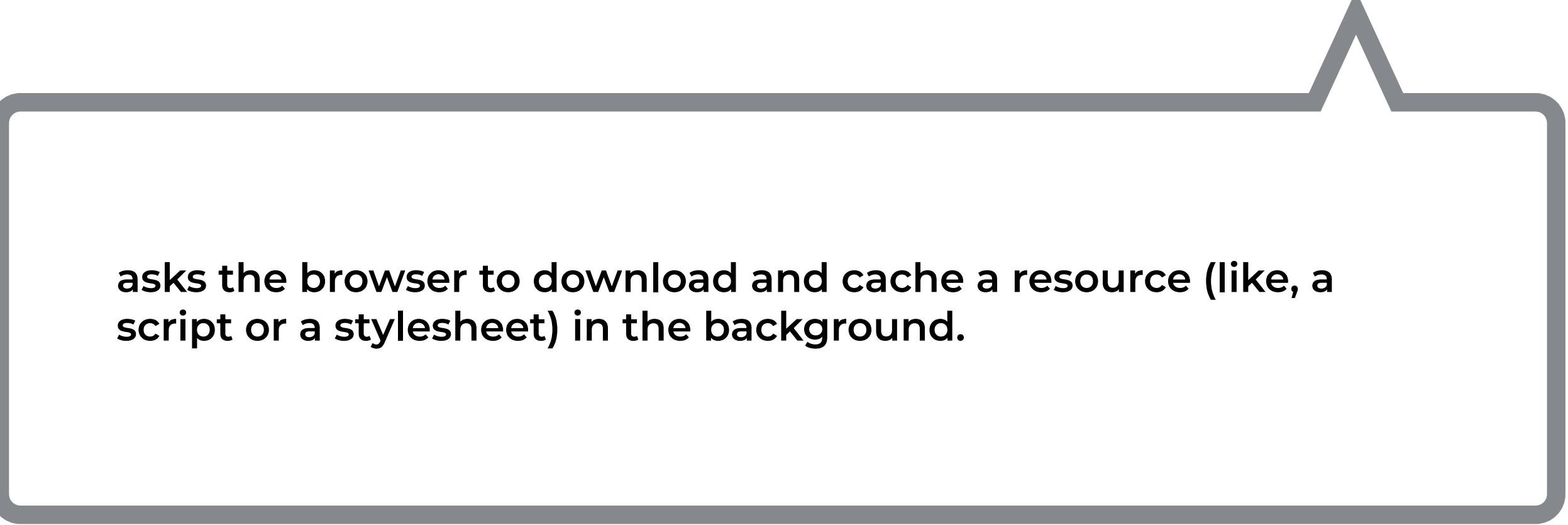


<https://csswizardry.com/2019/03/cache-control-for-civilians/>

A couple more pre's

# Prefetch

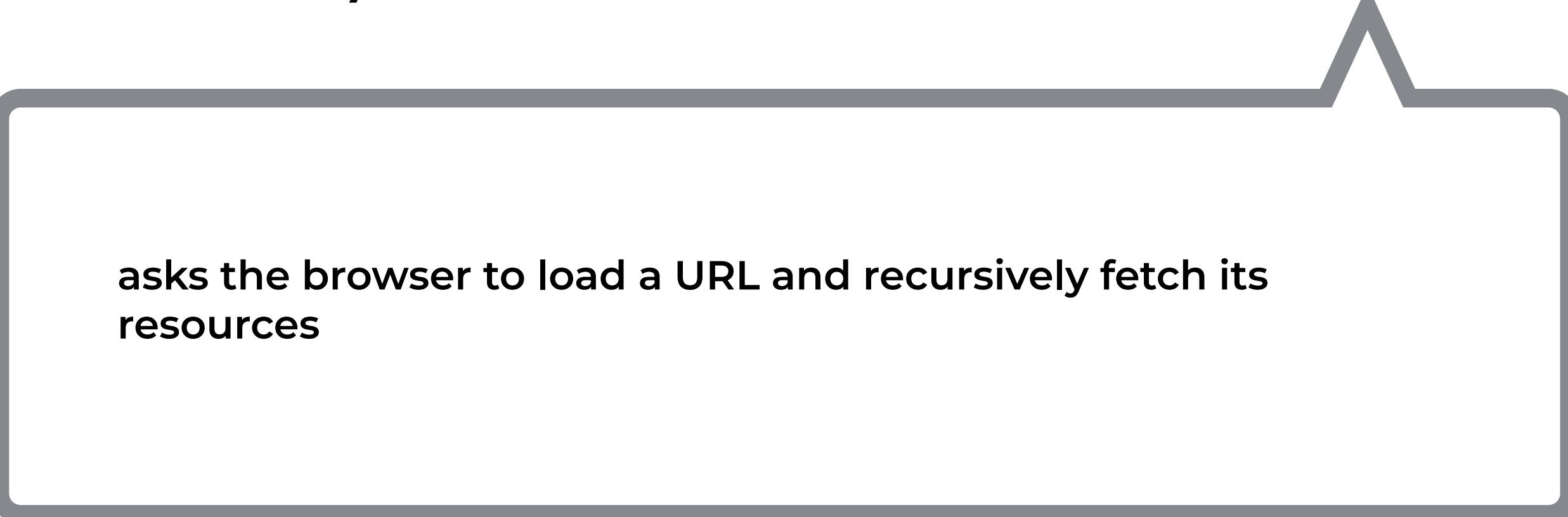
```
<link rel="prefetch" href="/style.css" as="style" />
```



asks the browser to download and cache a resource (like, a script or a stylesheet) in the background.

# Prerender

```
<link rel="prerender" href="https://example.com/  
about.html" />
```



asks the browser to load a URL and recursively fetch its resources

# Service **workers**

# What is a service worker?

- A script that downloads and listens in the background
- A tool for managing requests and responses
- A tool with great ability to work with local caches
- The best way to stabilize low (and no) connectivity

# Worker registration

```
if ('serviceWorker' in navigator) {  
  window.addEventListener('load', function() {  
    navigator.serviceWorker.register('/sw.js')  
  });  
}
```

# □ filament group

We help companies design and build super-fast responsive sites and web apps that are simple to use and accessible to everyone.

Looking for help on a project?

[Work with us!](#)

## Recent & Upcoming

JUN 25 | TORONTO [Smashing Conf](#)

[Move Fast & Don't Break Things](#)

Scott Jehl



## Filament Group wins Agency of the Year!

We're honored to have been voted [Agency of the Year](#) at the 2015 Net Awards, the web industry's largest peer-voted awards show.

Elements    Console    Sources    Network    Performance    Memory    **Application**    Security    Audits    2 2 | : X

**Application**

- Manifest
- Service Workers
- Clear storage

**Storage**

- ▶ Local Storage
- ▶ Session Storage
- IndexedDB
- Web SQL
- ▶ Cookies

**Cache**

**Service Workers**

Offline  Update on reload  Bypass for network

**www.filamentgroup.com** [Update](#) [Unregister](#)

Source [sw.js](#)  
Received 6/7/2019, 10:30:10 AM

Status #6186 activated and is running [stop](#)

Clients <https://www.filamentgroup.com/webappmanifest.json> [focus](#)

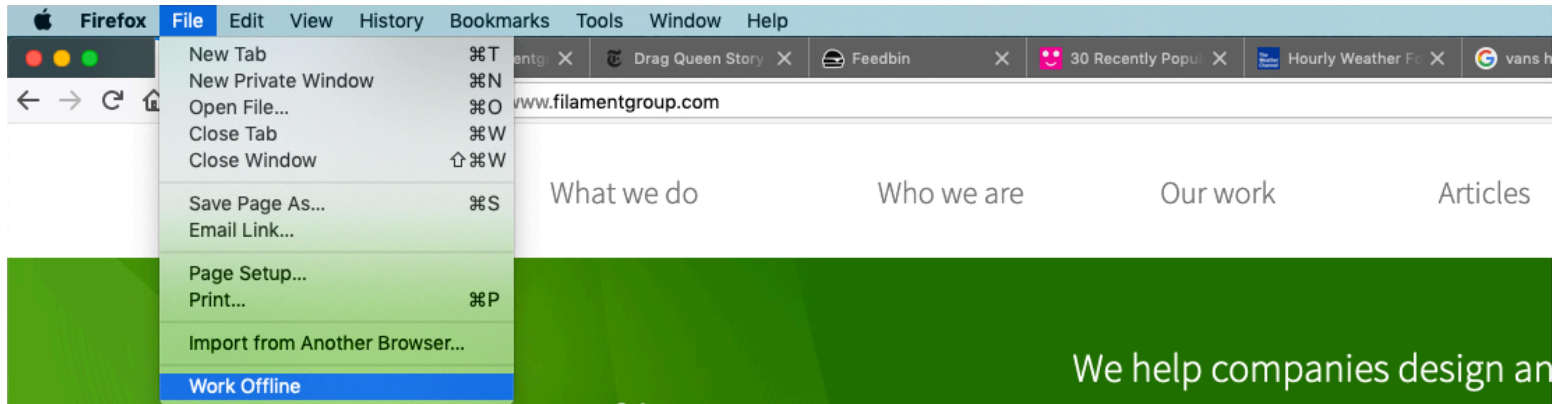
<https://www.filamentgroup.com/> [focus](#)

# In the worker...

```
var CACHE_NAME = 'my-site-cache-v1';
var urlsToCache = [
  '/styles/main.css',
  '/script/main.js'
];
self.addEventListener('install', function(event) {
  // Perform install steps
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});
```

# Offline-first, then network

```
self.addEventListener('fetch', function(event) {  
  event.respondWith(  
    caches.match(event.request)  
      .then(function(response) {  
        // Cache hit - return response  
        if (response) {  
          return response;  
        }  
        return fetch(event.request);  
      })  
    );  
});
```



## Recent & Upcoming

JUN 25 | TORONTO [Smashing Conf](#)

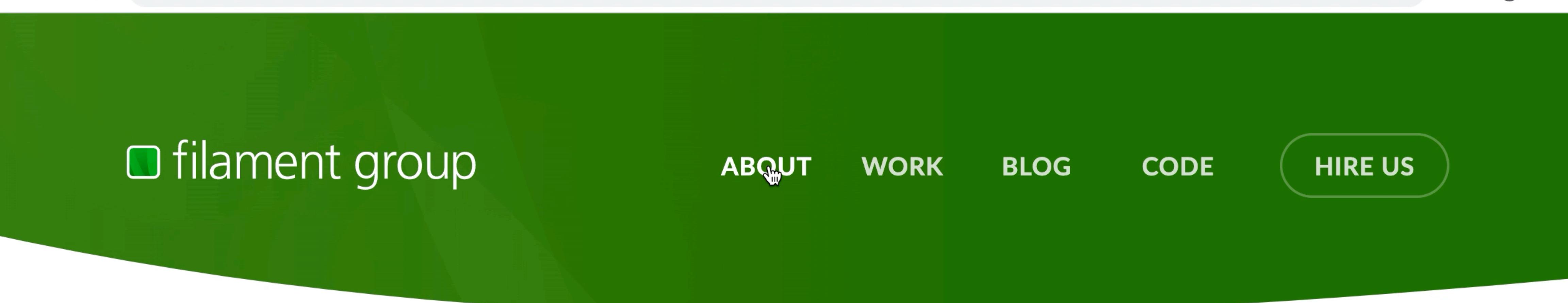
**Move Fast & Don't Break Things**

Scott Jehl



## Filament Group wins Ager

We're honored to have been vote  
the 2015 Net Awards, the web ind  
awards show.



# who we are

Filament Group is a design and front-end development studio. Since 2001 we've been designing and developing a wide range of sites and apps for big companies, innovative startups, and open-source projects.

# Completing the PWA

← → ⌂ 🔒 https://www.filamentgroup.com/webappmanifest.json

```
{  
  "lang": "en",  
  "dir": "ltr",  
  "name": "Filament Group",  
  "description": "We help companies design and build super-fast responsive sites and web apps that are simple to use and accessible to everyone.",  
  "short_name": "Filament",  
  "icons": [  
    {  
      "src": "/images/icons/icon-192.png",  
      "sizes": "192x192",  
      "type": "image/png"  
    },  
    {  
      "src": "/images/icons/icon-256.png",  
      "sizes": "256x256",  
      "type": "image/png"  
    },  
    {  
      "src": "/images/icons/icon-384.png",  
      "sizes": "384x384",  
      "type": "image/png"  
    },  
    {  
      "src": "/images/icons/icon-512.png",  
      "sizes": "512x512",  
      "type": "image/png"  
    }],  
  "start_url": "/",  
  "display": "standalone",  
  "orientation": "natural",  
  "theme_color": "#247200",  
  "background_color": "#247200"  
}
```

# Linking that up!

```
<link rel="manifest" href="/webappmanifest.json">
```

# Custom Request Headers

# Look, no cookies!

```
const version = "65";
self.addEventListener('fetch', event => {
if (event.request.headers.get('Accept').indexOf('text/html') !== -1) {
    var newReq;
    var reqHeaders = new Headers();
    for( var i in event.request.headers ){
        reqHeaders.set( i, event.request.headers[i] );
    }
reqHeaders.set( 'X-CACHED', version );
    newReq = new Request( event.request.url, {
        method: "GET",
        headers: reqHeaders
    } );
}
```

New visitor? Let's push some stuff.

```
<If "%{HTTP:x-cached} !~ /65/ && %{DOCUMENT_URI} =~ /\.html/">
```

```
    H2PushResource add site.css critical
```

```
</If>
```

# Not just **no** cookies; **better** than

```
function updateStaticCache() {  
    // code in this function will cache all the static assets!  
    ...  
}  
  
self.addEventListener('install', event => {  
    event.waitUntil(updateStaticCache()  
        .then( () => self.skipWaiting() )  
    );  
});
```



ABOUT WORK BLOG CODE

HIRE US

# Tuning Performance for New and “Old” Friends

---

POSTED BY  
**SCOTT JEHL**  
02/19/2019

For performance reasons, we often configure our sites to deliver their code in slightly different ways for new and returning visitors. For example, for new visitors, we might choose to inline or http2-push important resources along with a page’s HTML so that it will be ready to render immediately on arrival. Additionally, for a new visit we might set



<https://filamentgroup.com/lab/cache-state/>

# Caching Inlined Content

# Cache API: Inlining without the waste.

```
<style id="css">
  body { background: green; }
</style>
<script>
  var css = document.getElementById("css").innerHTML;
  caches.open('static').then(function(cache) {
    cache.put("site.css", new Response( css,
      {headers: {'Content-Type': 'text/css'}})
  ));
}) ;
</script>
```

[Home](#) / [Articles](#) /

# A Pretty Good SVG Icon System



Author  
Chris Coyier

43 Comments  
[Go to Comments](#) →

Published  
Jun 23, 2017

Updated  
Aug 16, 2017

Ship custom analytics today with [Keen.io](#).

Share: [Twitter](#) [Facebook](#)

I've long advocated SVG icon systems. Still do. To name a few benefits: vector-based icons look great in a high pixel density world, SVG offers lots of design control, and they are predictable and performant.

I've also often advocated for a SVG icon system that is based on `<symbol>`s (an "SVG sprite") and the `<use>` element for placing them. I've changed my mind a little. I don't think that is a bad way to go, really, but there is certainly a *simpler* (and perhaps a little *better*) way to go.



<https://css-tricks.com/pretty-good-svg-icon-system/>

# Inline-Cache for SVG?

```
<div id="icons">
  <svg> ... </svg>
</div>
<script>
  var svg =
    document.getElementById("icons").innerHTML;
    caches.open('static').then(function(cache) {
      cache.put("icons.svg", new Response( svg,
        {headers: {'Content-Type': 'image/svg+xml'}})
    ));
  });
</script>
```

# Inlining or Caching? Both Please!

---

POSTED BY  
**SCOTT JEHL**  
11/12/2018

Last week, I was finishing a section of my slides for a presentation at the [Performance.Now\(\) conference](#). In it, I was exploring patterns that enable the browser to render a page as fast as possible by including code alongside the initial HTML so that the browser has everything it needs to start rendering the page, without making additional requests.

Our two go-to options to achieve this goal are inlining and server push ([more on how we use those](#)), but each has drawbacks: inlining prevents a file from being cached for reuse, and server push is still a bit experimental, with [some browser bugs](#) still being worked out. As I was preparing to describe these



It's easier to  
**make a fast website**  
than it is to  
**keep a website fast...**