

PNW Drupal Summit 2025 Sponsors

Platinum

ACQUID



Gold

 Tugboat[®]

Modern Woodmen 
FRATERNAL FINANCIAL

Silver



Bronze

sandstorm[®]



In Kind Sponsors



TheWeeklyDrop

Individual Sponsors

Andrew Wilson, Anonymous, Daniel, Mundra,
Donna Beegle, Eric Wheeler, Erin Rasmussen,
Heather Wozniak, Joshua Mitchell,
Katherine Senzee, Lisa Godare, Todd Smith



Optimizing Drupal for Core Web Vitals: A Developer's Guide

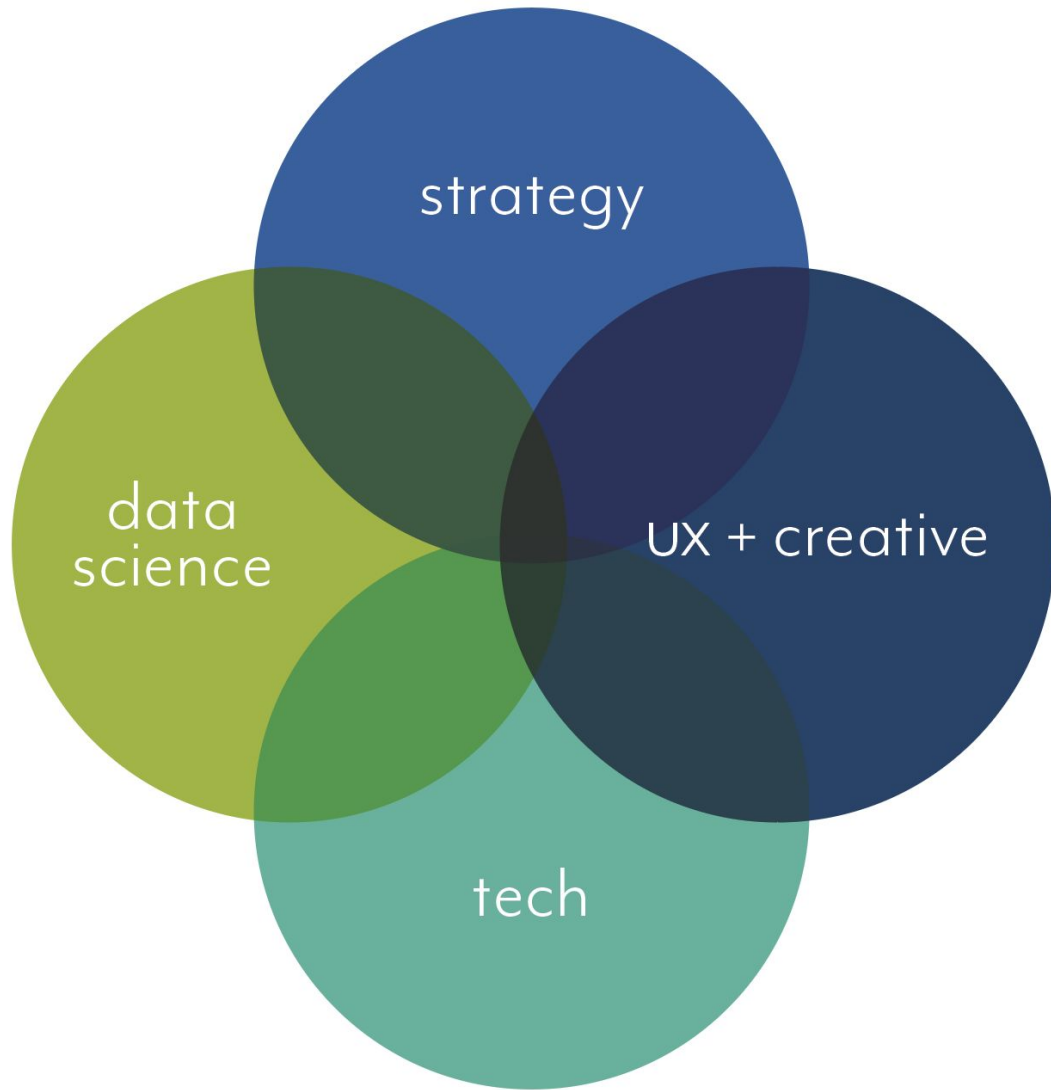
Hello! I'm Madeline



Madeline Jensen
Development Team
Lead

she/her

- **Development Team Lead at Sandstorm Design.**
We do good work for good people.
- **10 years of Drupal development experience,**
background in frontend development.
- **Outside of work** you will find me playing with my 3
big dogs. (German Shepherd, Bernese, and German
Shepherd/Bernie Mix)



More about Sandstorm.

- + **25+ yr old** digital experience design agency
- + 15+ yrs building **digital experiences for associations, nonprofits & healthcare organizations**
- + **4,600+ hrs** of usability & UX research
- + **CPACC** (Accessibility) certified
- + **DEIB Specialist** (Diversity, Equity, Inclusion, and Belonging)
- + **WBENC certified** (women-owned)



Agenda

- 01/** Understanding Core Web Vitals
- 02/** Common Performance Bottlenecks
- 03/** Theme-Level Optimizations
- 04/** Caching Strategies
- 05/** Measuring, Monitoring & Iterating
- 06/** Q&A

1

Understanding Core Web Vitals

FUN FACT

Lighthouse and **Page Speed**
Insights are not the same thing. 🤯

When to use **Lighthouse**



- Benchmarking performance **before and after** an optimization or a deployment.
- Identifying performance issues in your **own codebase**.
- **Experiment safely** — simulate improvements without affecting production traffic.

When to use **PageSpeed Insights**



- Identifying issues in your **caching strategy**.
- Monitoring the impact of **3rd party scripts** running on your site.
- **Track Core Web Vitals** using real-world field data.

Metrics for Core Web Vitals

Tip: Don't get lost in the acronyms, just focus on **moving the needle**.



- **LCP** (Largest Contentful Paint)
- **TTFB** (Time to First Byte)
- **FCP** (First Contentful Paint)
- **CLS** (Cumulative Layout Shift)
- **TBT** (Total Blocking Time)
- **INP** (Interaction to Next Paint)
- **FID** (First Input Delay)
- ...probably more!

2

Common Performance Bottlenecks (found in Drupal sites)

It's not Drupal's fault

Drupal has all the tools and friends it needs to be performant.



Render Blocking Resources

Common Performance Bottleneck

The Problem

- Large, global **main.css** or **main.js** files load on **every page**
- Scripts not being **deferred** or **loaded async**
- css/js not **minified**

Related Metrics

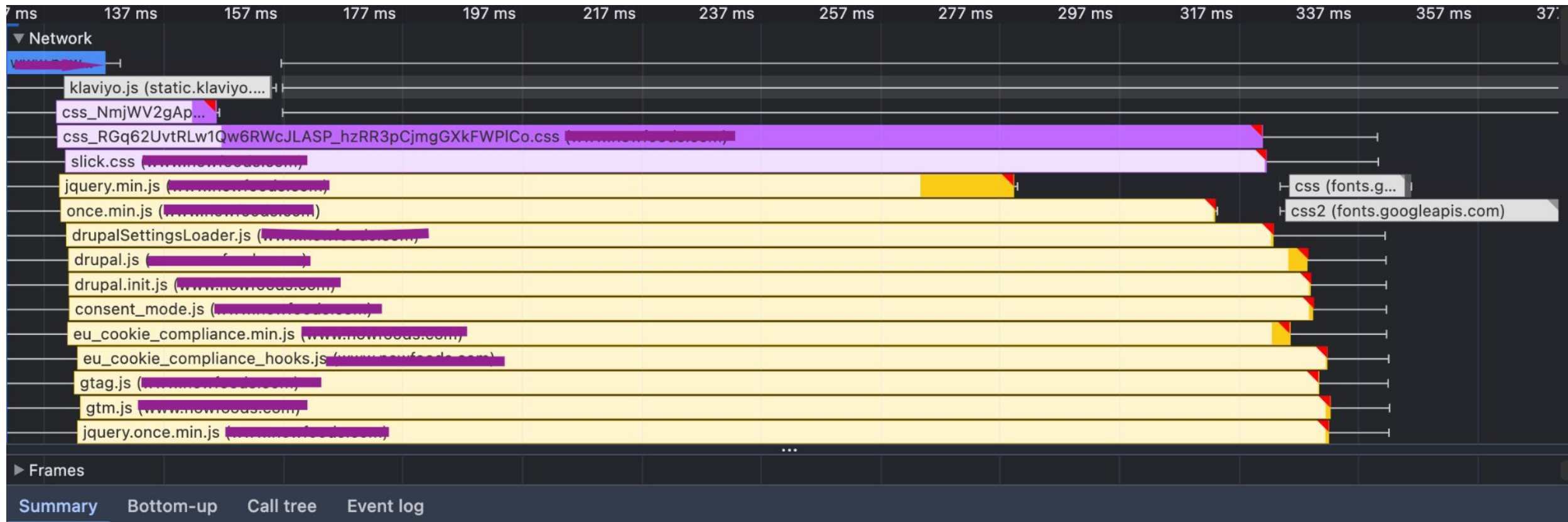
- **FCP** (First Contentful Paint)
- **LCP** (Largest Contentful Paint)
- **TBT** (Total Blocking Time)

Some Solutions

- Use **libraries.yml** or **SDC** to bundle your css/js into components.
- Use **attach_library** to only load those assets at the template level.
- **ABTAO**: Always be thinking about order.

Render Blocking Resources

Chrome Dev Tools > Performance panel tells a story



Layout Shifts - Above the Fold

Common Performance Bottleneck

The Problem

- Not prioritizing **above the fold** to load first. (ie. nav, hero image and styles)
- Relying on **third party libraries** that load before your own code and styles.
- Scripts not being **deferred** or **loaded async**

Related Metrics

- **FCP** (First Contentful Paint)
- **LCP** (Largest Contentful Paint)
- **TBT** (Total Blocking Time)

Some Solutions

- **Group libraries** by sections (global, nav, hero, footer, etc)
- **preload** important requests like fonts.
- **defer** footer, carousel or other interactive logic.
- **ABTAO**: Always be thinking about order. (AGAIN)

Too Many Network Requests

Common Performance Bottleneck

The Problem

- Not using a **CDN** like Cloudflare or Imperva
- Low **cache age** on Drupal content
- Invalidating Drupal cache tags

Related Metrics

- **TBT** (Total Blocking Time)
- **FID** (First Input Delay)
- **FCP** (First Contentful Paint)

Some Solutions

- **Implement Cloudflare** to cache css, js, and html
- **Be mindful** of blocks and views that break Drupal cache tags.

3

Theme Optimizations

TIP:

Only load what you need.

(No need to load all of bootstrap only for a modal, right?)

```
1 // Basic elements and structure
2 @import 'base/fonts';
3 @import 'base/inputs';
4 @import 'base/typography';
5 @import 'base/styles';
6 @import 'base/media';
7 @import 'base/new-buttons';
8 @import 'config/global-patterns';
9
10 // COMPONENTS
11 @import 'components/account';
12 @import 'components/accordion';
13 @import 'components/awards';
14 @import 'components/global-header';
15 @import 'components/navbar-toggler';
16 @import 'components/site-branding';
17 @import 'components/search-form';
18 @import 'components/search-results';
19 @import 'components/menu';
20 @import 'components/menu--header-utility';
21 @import 'components/menu--main';
22 @import 'components/menu--secondary';
23 @import 'components/content-header';
24 @import 'components/global-footer';
25 @import 'components/main-content';
26 @import 'components/breadcrumbs';
27 @import 'components/mailchimp-signup';
28 @import 'components/forms';
29 @import 'components/site-alerts';
30 @import 'components/layout-builder';
```

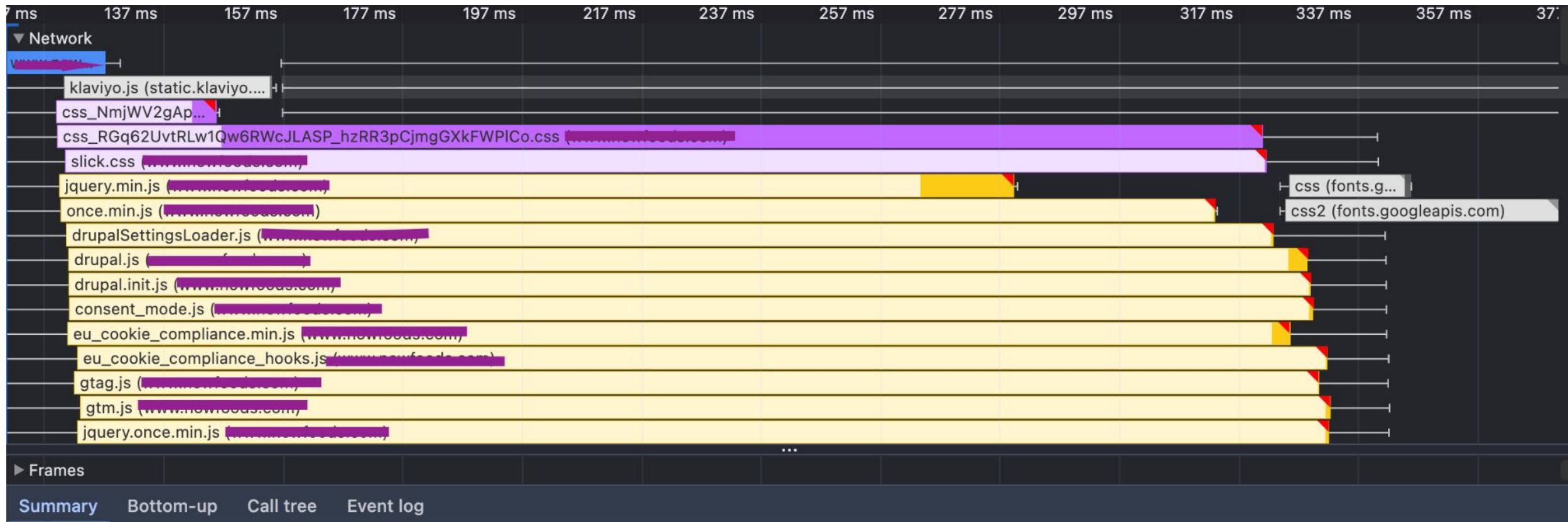
If your main.scss looks like this....

Start chunking it out.

(ABTAO) Always be thinking about order!

Render Blocking Resources

Chrome Dev Tools > Performance panel tells a story



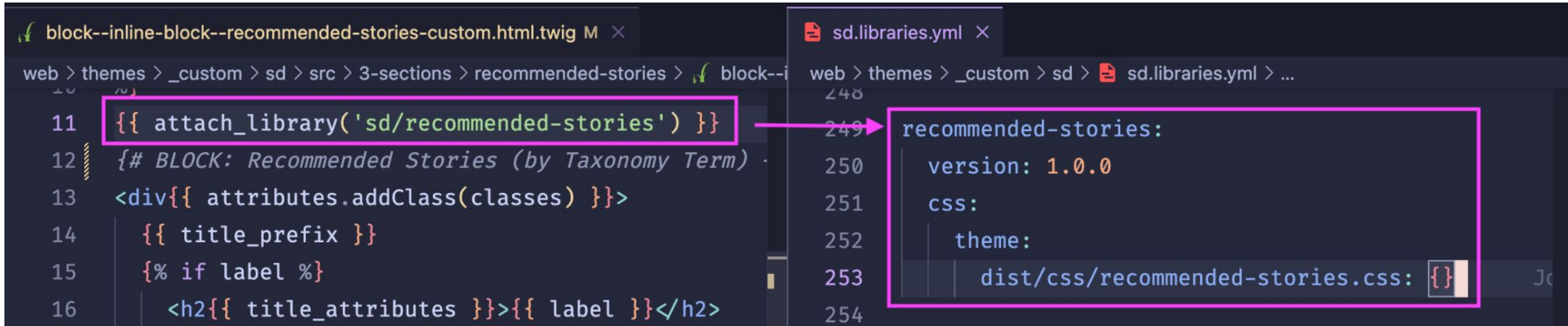
You are going to need a new build strategy.

- Organize your javascript and scss/css into **components**.
- Ditch the **main** or **global** single file entrypoint.
- Ask your **compiler** (webpack, vite) to look at each file individually.

```
rollupOptions: {
  input: Object.fromEntries(
    glob
      .sync("./src/**/*.{js,scss}", {
        ignore: ["./src/0-styles/utils/*.scss"] // Exclude
      })
      .map((file) => [
        path.basename(file),
        path.resolve(file),
      ])
  ),
  output: [{
    entryFileNames: ({ name }) => {
      if (name.endsWith('.js')) {
        return `js/${name.replace('.js', '')}.min.js`;
      }
      return `js/${name}.min.js`;
    },
  ]},
```


Use Drupal Libraries to your advantage

Reminder: Drupal has all the tools and friends it needs to be performant.



```
block--inline-block--recommended-stories-custom.html.twig M x
web > themes > _custom > sd > src > 3-sections > recommended-stories > block--i
11 {{ attach_library('sd/recommended-stories') }}
12 {# BLOCK: Recommended Stories (by Taxonomy Term)
13 <div{{ attributes.addClass(classes) }}>
14   {{ title_prefix }}
15   {% if label %}
16   <h2{{ title_attributes }}>{{ label }}</h2>

sd.libraries.yml x
web > themes > _custom > sd > sd.libraries.yml > ...
248
249 recommended-stories:
250   version: 1.0.0
251   css:
252     theme:
253       dist/css/recommended-stories.css: {}
254
```

4 Caching Strategies



Layered Caching Strategy

- **CDN:** Delivers cached assets at the edge; lowers latency and load.
- **Varnish:** Serves cached pages; offloads backend and handles failover.
- **Redis:** Caches data in memory; accelerates Drupal responses.

Test your real cache headers

Chrome Dev Tools

> Network

> Headers

The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The 'General' section displays the request details: Request URL is https://www.unicefusa.org/, Request Method is GET, Status Code is 200 OK (indicated by a green dot), Remote Address is 104.18.5.212:443, and Referrer Policy is strict-origin-when-cross-origin. The 'Response Headers' section lists several headers: Access-Control-Allow-Origin is *, Age is 24917, Alt-Svc is h3=":443"; ma=86400, Cache-Control is max-age=259200, public, and Cf-Cache-Status is DYNAMIC.

Section	Header/Property	Value
General	Request URL	https://www.unicefusa.org/
	Request Method	GET
	Status Code	200 OK
	Remote Address	104.18.5.212:443
	Referrer Policy	strict-origin-when-cross-origin
Response Headers	Access-Control-Allow-Origin	*
	Age	24917
	Alt-Svc	h3=":443"; ma=86400
	Cache-Control	max-age=259200, public
	Cf-Cache-Status	DYNAMIC

Play with Drupal Cache Tags

- Use Devel or PHP to inspect a **node's render array**.
- Edit the page and check **cache tags** to see which items trigger **invalidation**.
- Check if **cache max-age** is what you expect.

```
array:6 [▼  
  "#node" => Drupal\node\Entity\node {#1409 ▶}  
  "#view_mode" => "full"  
  "#cache" => array:5 [▼  
    "tags" => array:2 [▼  
      0 => "node_view"  
      1 => "node:39725"  
    ]  
    "contexts" => array:1 [▶]  
    "max-age" => -1  
    "keys" => array:4 [▶]  
    "bin" => "render"  
  ]  
  "#theme" => "node"  
  "#weight" => 0  
  "#pre_render" => array:1 [▶]  
]
```


TIP:

**Find your cache max-age
sweet spot for your site.**

How far do you think you push the
cache life for your assets?

5

Measuring, Monitoring & Iterating

FAILING TO PLAN

IS PLANNING TO FAIL

FACT(ish):

**You likely will not see
improvements overnight.**

You are going to need a plan.

A Phased Approach

To maintaining a performant Drupal site

Phase 1

- **Benchmark your baseline** before you start.
- **Plan your sprints**, prioritizing high impact but accomplishable changes.
- Implement a **performance monitoring tool** (or process)

Phase 2

- Implement and track changes **with each deployment.**
- **Benchmark before and after deployments.** (If doing manually, you can run Lighthouse).
- Keep in mind real user data in **GPSI is 28 days avg.**

Phase 3

- **Monitor GPSI scores** to tell the real user story over time,
- **Find clues for Lighthouse and GPSI** differences by using **RUM.**
- **Push the boundaries** of your cache age - can you cache a 404 page template for... **forever?**



What should we measure?

- **Page Load Time:** Overall user-perceived speed (LCP, TTFB, fully loaded).
- **Cache Hit Ratio:** Percentage of requests served from cache (CDN, Varnish, Redis).
- **Backend Performance:** PHP execution time, database query count, memory usage.
- **Cache Invalidation Events:** How often content changes trigger cache clears.



What tools do we have to work with?



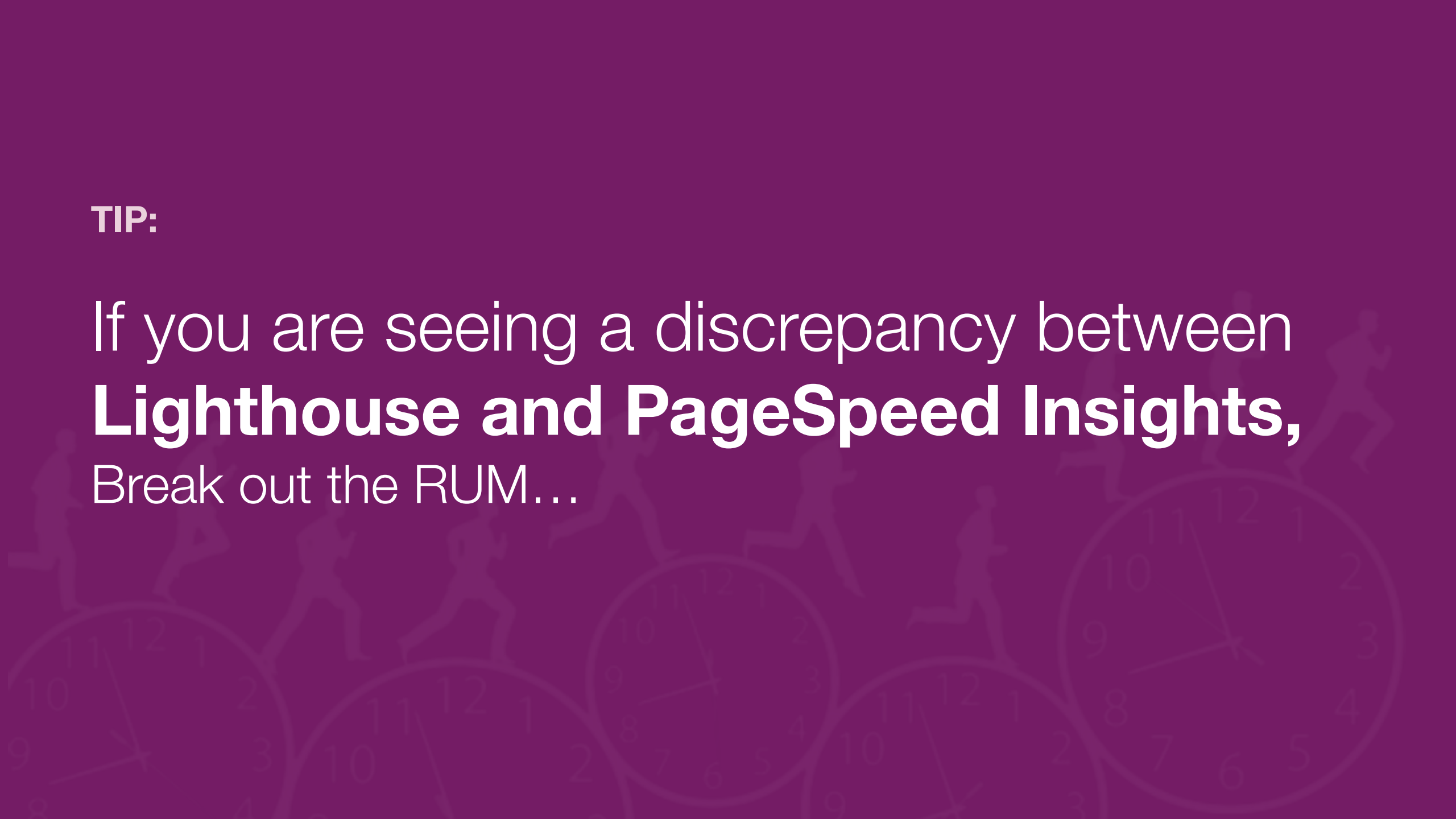
**Google
Lighthouse**



DATADOG

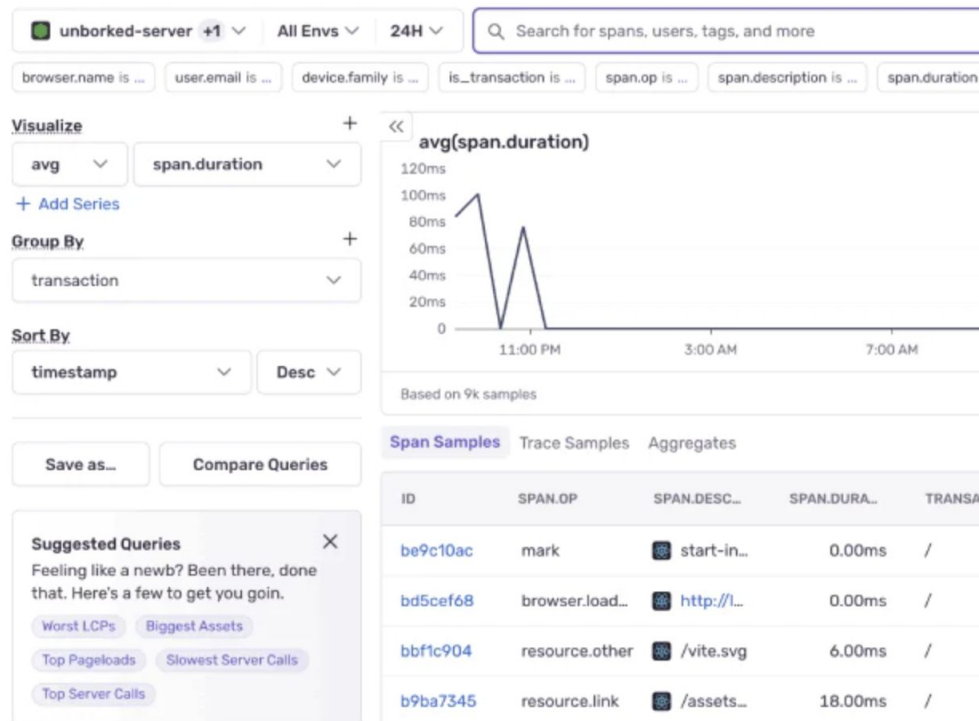
TIP:

If you are seeing a discrepancy between
Lighthouse and PageSpeed Insights,
Break out the RUM...

The background of the slide is a solid purple color. It features a repeating pattern of faint, light-colored clock faces and silhouettes of people running. The clock faces are circular with numbers 1 through 12 and hands. The running figures are in various stages of a stride, scattered across the background.

R.U.M (Real User Monitoring)

Traces ?



Collect info such as:

- **Page load times** as experienced by actual visitors.
- **Core Web Vitals (LCP, FID, CLS)** in real-world conditions.
- **Geography & device impact** — how performance varies by location or device.
- **Capture cache headers** to look for cache invalidations.

When to use **Lighthouse**



- Benchmarking performance **before and after** an optimization or a deployment.
- Identifying performance issues in your **own codebase**.
- **Experiment safely** — simulate improvements without affecting production traffic.

When to use **PageSpeed Insights**



- Identifying issues in your **caching strategy**.
- Monitoring the impact of **3rd party scripts** running on your site.
- **Track Core Web Vitals** using real-world field data.

6 Q&A