Styling the Navigation: Declarative Route and Navigation Matching in CSS

July 30, 2026 https://brm.us/styling-the-navigation Leave a comment

INTRO ↠ One of the things that we at Chrome have been thinking about for a while now is a way to do declarative route and navigation matching in CSS. Together with Noam Rosenthal and David Baron I've been working on this for the past few months.

After an initial introduction at the CSS Working Group back in January, and a quick feedback session with developers at CSS Day in June, we think we have something that is ready for further discussion.

At next week's CSS Working Group F2F (face-to-face) meeting in Berlin, we'll be presenting our current line of thinking. This post is a quick intro to the concepts we'll be covering.

~

The Problem: Knowing where you're going

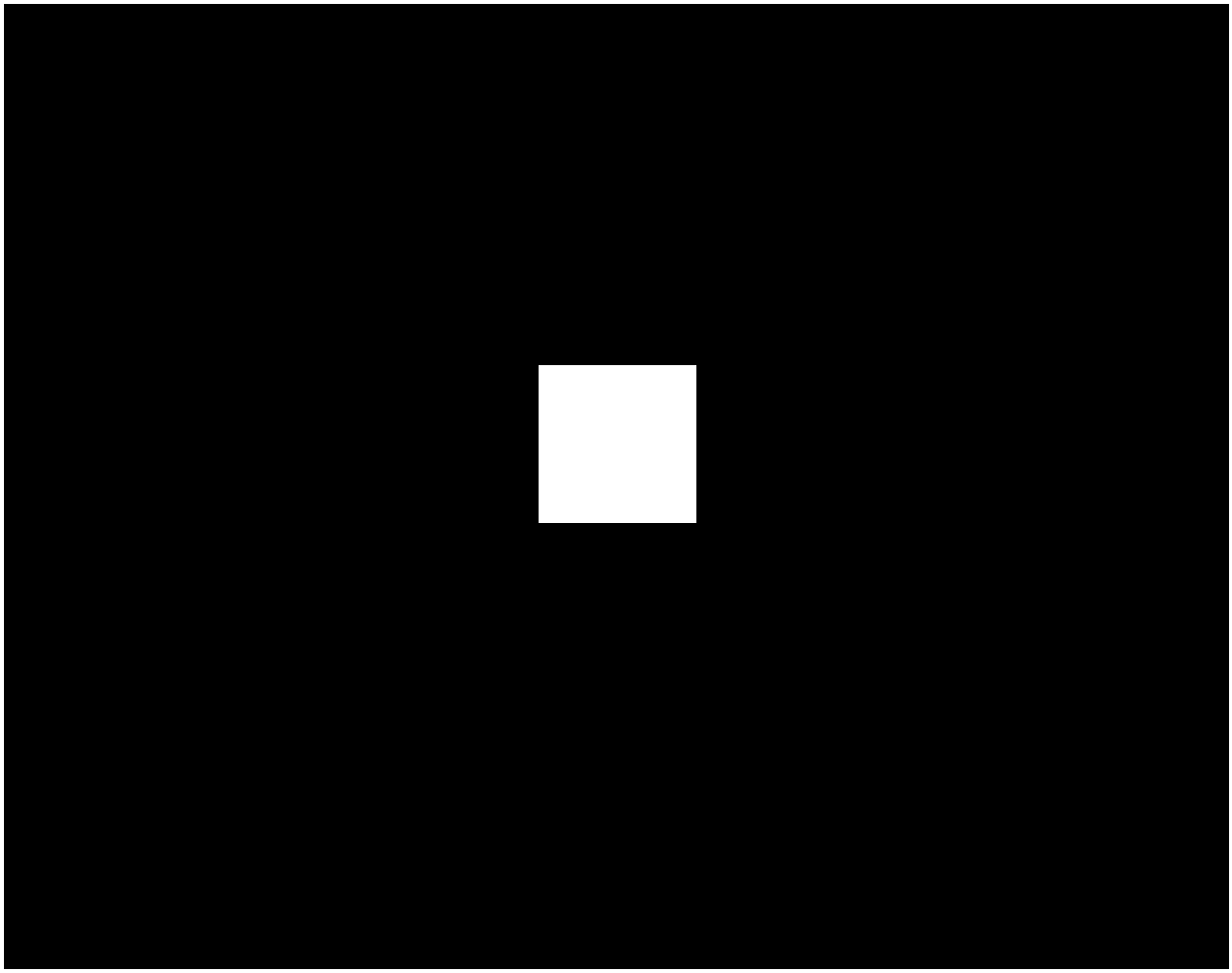
If you've played around with View Transitions, especially for Multi-Page Apps (MPAs), you've probably hit a familiar wall. Very often, you want to style a page differently based on *where you are coming from* and *where you are going to*.

Imagine a simple flow:

- When navigating from the index page to the about page, you want the whole site to slide to the left.
- When navigating back from the about page to the index page, you want it to slide to the right.
- When clicking a thumbnail in a list to go to a detail page, you want that specific clicked thumbnail to transition into the hero image.

Right now, doing this requires JavaScript to intercept the navigation, figure out the URLs using `pageswap` and `pagereveal`, find the `NavigateEvent.sourceElement`, and dynamically set the View Transition Types based on those values.

To see it in action, check out this **this demo** that uses the components mentioned above.



It works, but the script required to do it right grows over time and can become quite complicated. We wanted to see if styling the navigation experience can be done declaratively instead, in CSS.

~

Say Hello to Route and Navigation Matching 👋

The [CSS Route and Navigation Matching Specification (`css-navigation-1`)](#) introduces a few new key components to CSS that allow you to style the navigation and links declaratively.

> "CSS conditioned on the status of navigating between particular URLs."

Here is a quick *(and abbreviated)* overview of what we are proposing:

~

1. Define Routes with `@route`

Instead of constantly typing out URL strings, you can define named routes using the `url-pattern()` function.

```
@route --home { pathname: url-pattern('/'); }
@route --about { pathname: url-pattern('/about'); }
@route --detail { pathname: url-pattern('/detail/:id'); }
```

Pattern matching is done using the syntax from the popular [path-to-regexp](#) library, which is also used by JavaScript's `URLPattern`.

(For completeness: You can also use the `protocol`, `hostname`, `port`, `pathname`, `search`, and `hash` descriptors to define a named route.)

~

2. Query the navigation with `@navigation`

This is where the magic happens. You can query the current, ongoing, navigation state using the `@navigation` at-rule, checking the `from` and `to` endpoints.

```
/* Going from home to about? Slide left! */
@navigation (from: --home) and (to: --about) {
  @view-transition {
    navigation: auto;
    types: slide-all-to-left;
  }
}

/* Going back from about to home? Slide right! */
@navigation (from: --about) and (to: --home) {
  @view-transition {
    navigation: auto;
    types: slide-all-to-right;
  }
}
```

You can query against a named route (defined with `@route`) or a direct `url-pattern()`.

(For completeness: We are also introducing `between` and `at` keywords for the “navigation relation”.)

~

3. Select the element that triggered the navigation with `:nav-source`

The `:nav-source` pseudo-class selector allows you to target the exact element that initiated the outgoing navigation. This is modeled after `NavigateEvent.sourceElement`.

For example, If you want to apply a `view-transition-name` only to the specific image that was clicked, you can do this:

```
@navigation (between: --home --detail) {
  @navigation (at: --home) {
    /* Target the clicked link's image */
    :nav-source img {
      view-transition-name: image;
    }
  }
  @navigation (at: --detail) {
    img#hero {
      view-transition-name: image;
    }
  }
}
```

~

4. Advanced link selection with `:link-to()`

The spec also introduces ways to style links based on what they link to using the `:link-to(...)` selector. Its argument is a named route, or `url-pattern()`. This is incredibly powerful for orchestrating complex UI transitions between routes.

```
/* Target the all links that links to the --detail route */
:link-to(--detail) {
  color: hotpink;
}
```

> *Note: Currently not covered is a way to do parameter matching with `:link-to()` — e.g. "select the link that links to a `--detail` route and whose `id` is equal to `6`" — as we are still working on that part to make sure the syntax is easy to use and covers all use cases.*

~

We need your feedback!

Over the past months we have given this a lot of thought and have gone back and forth over the key features and their syntax, and now we think we have something that could work. As mentioned we will be presenting this at next week's CSS Working Group F2F meeting, seeking resolutions to verify with the group that we are down the right path.

Back in June we did an introductory pitch at CSS Day, and a quick read of the room told us this was worth pursuing. Of course we are very open to feedback and would love to hear from you if you have any input.

Are we missing a use case? Do the keywords make sense? Or like one of the open questions we have is about the default base URL be used: should relative links be resolved against the document's location or the stylesheet's location?

Please take a look at the **current shape of the API** and leave your feedback in the **w3c/csswg-drafts#12594** at the CSSWG, or reach out to me (or Noam) on social media.

Let's build this together! 🚀

PS: You can try this out in Chrome Canary with the Experimental Web Platform Features flag enabled 😉

~

Spread the word

Feel free to reshare one of the following posts on social media to help spread the word:

- 🦋 **Bluesky**
- 🦣 **Mastodon**
- 💼 **LinkedIn**

~

🔥 ***Like what you see? Want to stay in the loop? Here's how:***

- 🦋 ***Follow @bram.us on Bluesky***
- 🔸 ***Follow bram.us using RSS***

I can also be found on ***𝕏 Twitter*** *and* ***🐘 Mastodon*** *but only post there sporadically.*

Published by Bramus! Bramus is a frontend web developer from Belgium, working as a Chrome Developer Relations Engineer at Google. From the moment he discovered view-source at the age of 14 *(way back in 1997)*, he fell in love with the web and has been tinkering with it ever since ***(more …)***
View more posts

Leave a comment

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Notify me of followup comments via e-mail. You can also **subscribe** without commenting.

Post Comment

This site uses Akismet to reduce spam. **Learn how your comment data is processed.**