Cross-browser paint worklets and Houdini.how

- Understanding Houdini worklets
- Implementing your own features with the CSS Painting API
- Houdini.how
- Using Houdini
  - 1. Prototyping with a CDN
  - 2. Managing worklets via NPM
- Contribute

https://www.youtube.com/watch?v=5eBar5TI71M

CSS Houdini is an umbrella term that describes a series of low-level browser APIs that give developers much more control and power over the styles they write.

> *Houdini layer*

Houdini enables more semantic CSS with the Typed Object Model. Developers can define advanced CSS custom properties with syntax, default values, and inheritance through the Properties and Values API.

It also introduces paint, layout, and animation worklets, which open up a world of possibilities, by making it easier for authors to hook into the styling and layout process of the browser's rendering engine.

Understanding Houdini worklets

Houdini worklets are browser instructions that run off the main thread and can be called when needed. Worklets enable you to write modular CSS to accomplish specific tasks, and require a single line of JavaScript to import and register. Much like service workers for CSS style, Houdini

worklets are registered to your application, and once registered can be used in your CSS by name.

Write worklet file Register worklet module (`CSS.paintWorklet.addModule(workletURL)`) Use worklet (`background: paint(confetti)`)

Implementing your own features with the CSS Painting API

The CSS Painting API is an example of such a worklet (the Paint worklet), and enables developers to define canvas-like custom painting functions that can be used directly in CSS as backgrounds, borders, masks, and more. There is a whole world of possibilities for how you can use CSS Paint in your own user interfaces.

For example, instead of waiting for a browser to implement an angled borders feature, you can write your own Paint worklet, or use an existing published worklet. Then, rather than using border-radius apply this worklet to borders and clipping.

> The example above uses the same paint worklet with different arguments (see code below) to accomplish this result. Demo on Glitch.

```
.angled {
  --corner-radius: 15 0 0 0;
  --paint-color: #6200ee;
  --stroke-weight: 0;

  /* Mask every angled button with fill mode */
  -webkit-mask: paint(angled-corners, filled);
}

.outline {
  --stroke-weight: 1;

  /* Paint outline */
  border-image: paint(angled-corners, outlined) 0 fill !important;
}
```

The CSS Painting API is currently one of the best-supported Houdini APIs, its spec being a W3C candidate recommendation. It is currently enabled in all Chromium-based browsers, partially supported in Safari, and is under consideration for Firefox.

Caniuse support

The CSS Painting API is currently supported on Chromium-based browsers.

But even without full browser support, you can still get creative with the Houdini Paint API and see your styles work across all modern browsers with the CSS Paint Polyfill. And to showcase a few unique implementations, as well as to provide a resource and worklet library, my team built houdini.how.

Houdini.how

Worklet page screenshot.

Screenshot from the Houdini.how homepage.

Houdini.how is a library and reference for Houdini worklets and resources. It provides everything you need to know about CSS Houdini: browser support, an overview of its various APIs, usage information, additional resources, and live paint worklet samples. Each sample on Houdini.how is backed by the CSS Paint API, meaning they each work on all modern browsers. Give it a whirl!

Using Houdini

Houdini worklets must either be run via a server locally, or on HTTPS in production. In order to work with a Houdini worklet, you will need to either install it locally or use a content delivery network (CDN) like unpkg to serve the files. You will then need to register the worklet locally.

There are a few ways to include the Houdini.how showcase worklets in your own web projects. They can either be used via a CDN in a prototyping capacity, or you can manage the worklets on your own using npm modules. Either way, you'll want to also include the CSS Paint Polyfill to ensure they are cross-browser compatible.

1. Prototyping with a CDN

When registering from unpkg, you can link directly to the worklet.js file without needing to locally install the worklet. Unpkg will resolve to the worklet.js as the main script, or you can specify it yourself. Unpkg will not cause CORS issues, as it is served over HTTPS.

```
CSS.paintWorklet.addModule("https://unpkg.com/<package-name>");
```

Note that this does not register the custom properties for syntax and fallback values. Instead, they each have fallback values embedded into the worklet.

To optionally register the custom properties, include the properties.js file as well.

```
<script src="https://unpkg.com/<package-name>/properties.js"></script>
```

To include the CSS Paint Polyfill with unpkg:

```
<script src="https://unpkg.com/css-paint-polyfill"></script>
```

2. Managing worklets via NPM

Install your worklet from npm:

```
npm install <package-name>
npm install css-paint-polyfill
```

Importing this package does not automatically inject the paint worklet. To install the worklet, you'll need to generate a URL that resolves to the package's worklet.js, and register that. You do so with:

```
CSS.paintWorklet.addModule(..file-path/worklet.js)
```

The npm package name and link can be found on each worklet card.

You will also need to include the CSS Paint Polyfill via script or import it directly, as you would with a framework or bundler.

Here is an example of how to use Houdini with the paint polyfill in modern bundlers:

```
import 'css-paint-polyfill';
import '<package-name>/properties.js'; // optionally register properties
import workletURL from 'url:<package-name>/worklet.js';

CSS.paintWorklet.addModule(workletURL);
```

For more specific instruction per-bundler, check out the usage page on Houdini.how.

Contribute

Now that you've played around with some Houdini samples, it's your turn to contribute your own! Houdini.how does not host any worklets itself, and instead showcases the work of the community.

If you have a worklet or resource you would like to submit, check out the github repo with contribution guidelines. We'd love to see what you come up with!

□□□ #3
□ □□□ □□□ 2026-10-31 21:56:18 UTC
□ cocobook-sync □□□ 2026-11-04 17:37:15 UTC