

CSS Houdini

Painting the Web

About me

I'm a Frontend Engineer with a passion for web animation and UX. The past two years I've been lecturing about web development at the University of Applied Science Salzburg. Now I'm an Developer Experience Engineer at [Storyblok](#).

[lisilinhart.info](#)

[twitter.com/lisi_linhart](#)

[codepen.io/lisilinhart](#)

[www.storyblok.com](#)



what is CSS Houdini?



What is CSS Houdini

CSS Houdini is a W3C effort to define lower-level CSS DOM APIs for authors to understand, recreate, and extend high-level CSS authoring features.

<https://wiki.mozilla.org/CSS/Houdini>

What is CSS Houdini

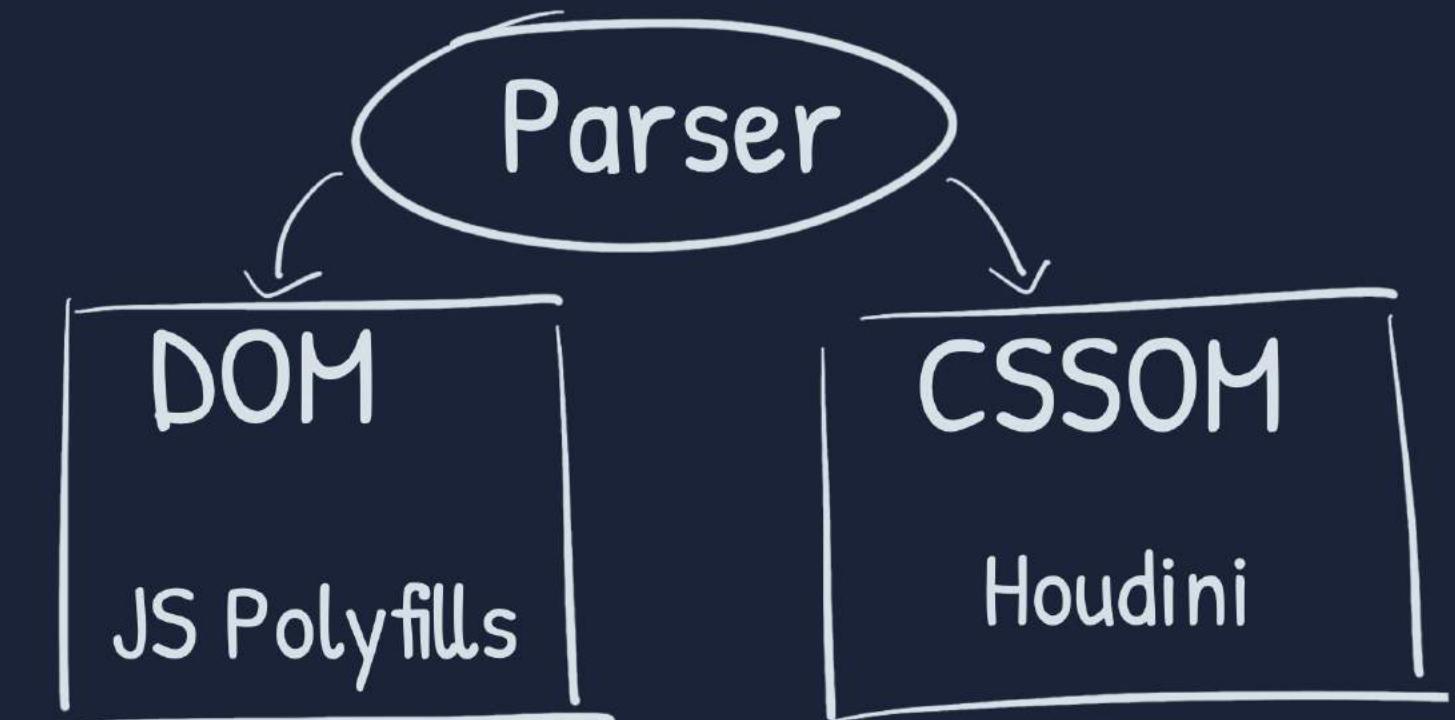
Extending CSS via JS

Houdini introduces low-level JavaScript APIs for the browser's render engine. This is simimilar to how Service Workers are a low-level JavaScript API for the browser's cache.

What is CSS Houdini

Access to the rendering engine

Houdini allows you to extend the CSS with new features using JavaScript, hook into CSS rendering engine and tell the browser how to apply CSS during a render process.



Can we use it?

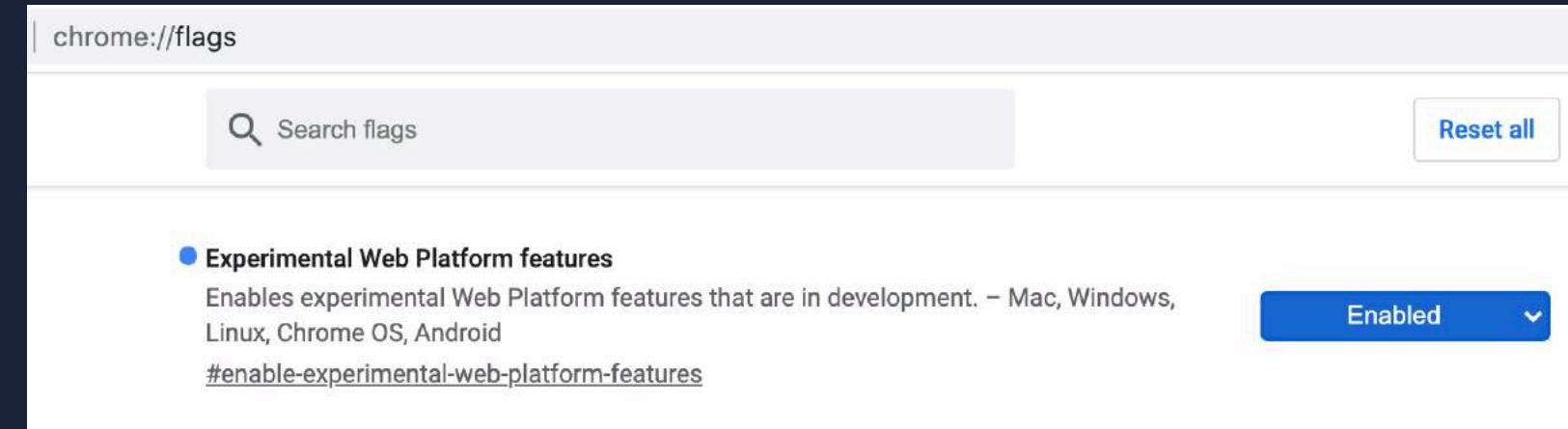
see [ishoudinireadyyet](#)?

Is Houdini ready yet?

Engine	Google Chrome	Microsoft Edge	Opera	Samsung Internet	Mozilla Firefox	Apple Safari	W3C Spec
Layout API (Explainer Demos)	Partial support (Canary) Details	Partial support (Canary) Details	Partial support (Developer) Details	No signal	Under consideration Details	No signal	First Public Working Draft Spec
Paint API (Explainer Demos Article)	Shipped (Chrome 65) Details	Shipped (Edge 79) Details	Shipped (Opera 52) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Candidate Recommendation Spec
Parser API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec
Properties & Values API (Demos Article)	Shipped (Chrome 78) Details	Shipped (Edge 79) Details	Shipped (Opera 65) Details	Shipped (Internet 12.0) Details	Under consideration Details	Partial support (Safari TP 67) Details	Working Draft Spec
AnimationWorklet (Explainer Demos Article)	Partial support (Chrome 71) Details	Partial support (Edge 79)	Partial support (Opera 58)	Partial support (Internet 10.2 Beta)	No signal	No signal	First Public Working Draft Spec
Typed OM (Explainer Article)	Shipped (Chrome 66) Details	Shipped (Edge 79) Details	Shipped (Opera 53) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Working Draft Spec
Font Metrics API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec

Testing it out

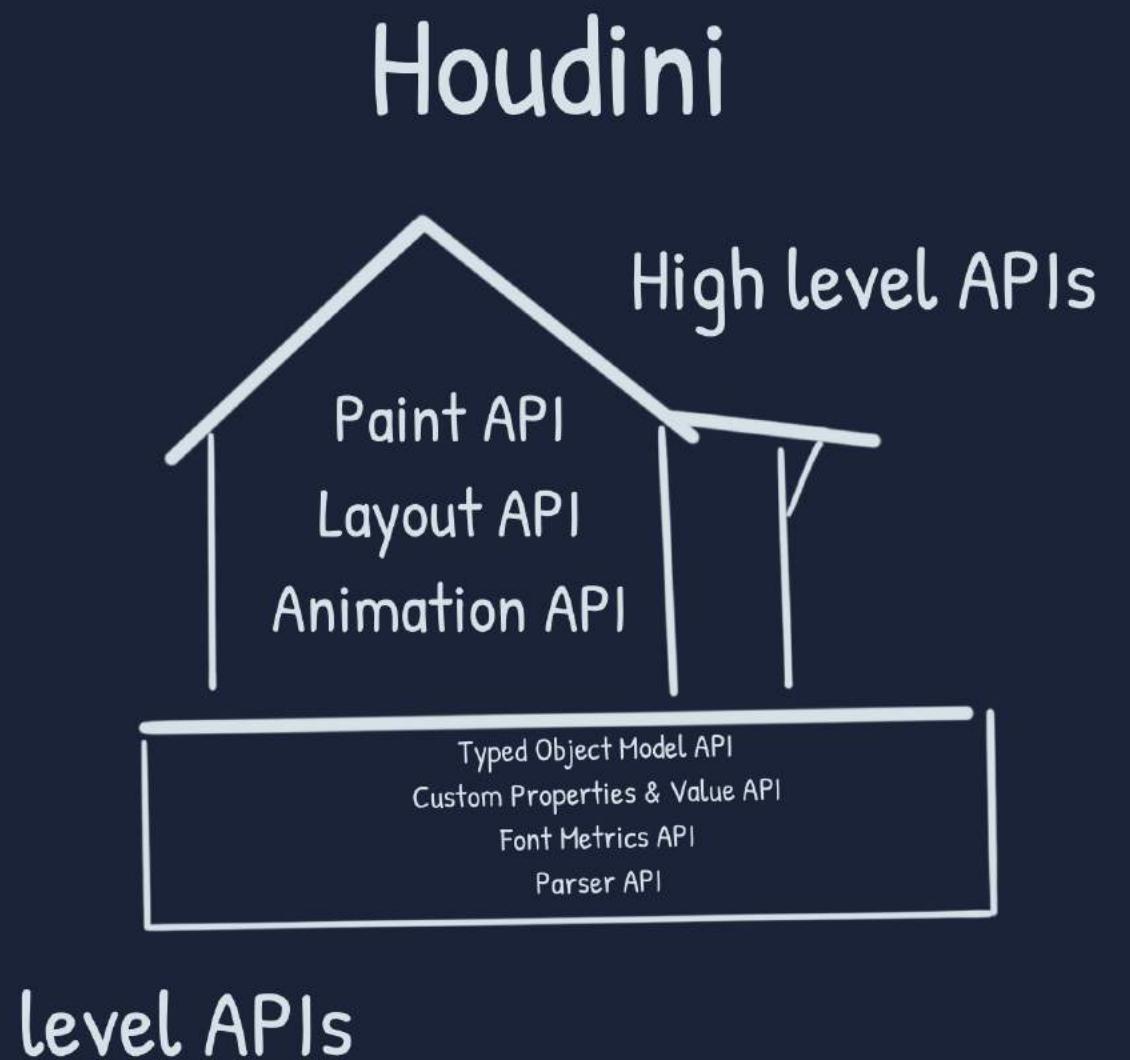
Activate experimental web platform features on
<chrome://flags/>



Houdini APIs

Houdini APIs

The Houdini APIs allow you to work with the CSS parser, CSSOM, cascade, the layout, paint and composite rendering stages.



Houdini APIs - High Level

The high level apis are closely related to the browser's rendering process (style → layout → paint → composite)

→ Paint API

→ Layout API

→ Animation API

Engine	Google Chrome	Microsoft Edge	Opera	Samsung Internet	Mozilla Firefox	Apple Safari	W3C Spec
	Partial support (Canary) Details	Partial support (Canary) Details	Partial support (Developer) Details	No signal	Under consideration Details	No signal	
Layout API (Explainer Demos)	Paint API (Explainer Demos Article)	AnimationWorklet (Explainer Demos Article)	Blink	Gecko	WebKit	-	
	Shipped (Chrome 65) Details	Shipped (Edge 79) Details	Shipped (Opera 52) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Candidate Recommendation Spec
	Partial support (Chrome 71) Details	Partial support (Edge 79) Details	Partial support (Opera 58) Details	Partial support (Internet 10.2 Beta) Details	No signal	No signal	First Public Working Draft Spec

Paint API (High-level)

An extension point for the browser's paint rendering step where visual properties (color, background, border, etc.) are determined.

- similar to a canvas drawing context
- can "paint" anywhere, images are supported
- has already quite good browser support (apart from Firefox)

Paint API (High-level)

Demo:
<https://css-houdini.rocks/tooltip>

Read more:
<https://houdini.glitch.me/paint>



The screenshot shows a user interface for configuring CSS properties related to the Paint API. At the top, there are three input fields: 'CSS' containing 'S00000', 'color' (hex code), and 'HUGE!?' (text). Below these are several slider controls:

- highlighter-pen: A dropdown menu set to "Highlighter marker".
- highlighter-size: A slider with a blue handle.
- highlighter-opacity: A slider with a blue handle.
- highlighter-smooth: A slider with a blue handle.
- transition-timing-function: A dropdown menu set to "ease-in-quart".
- transition-duration: A slider with a blue handle.

At the bottom, there is a code snippet showing a CSS rule for an element (el) with various properties defined:

```
.el {  
  --highlighter-pen: url('/posts/highlighter-marker-annotations/highlighter.png');  
  --highlighter-color: cyan;  
  --highlighter-path: path("M46.83,230.9c24.41,-51.06 455.28,-6.72 465.09,-16.82");  
  --highlighter-size: 21;  
  --highlighter-opacity: 0.45;  
}
```

There are also small buttons for "Replay animation" and "LIVE EDIT".

Layout API (High-level)

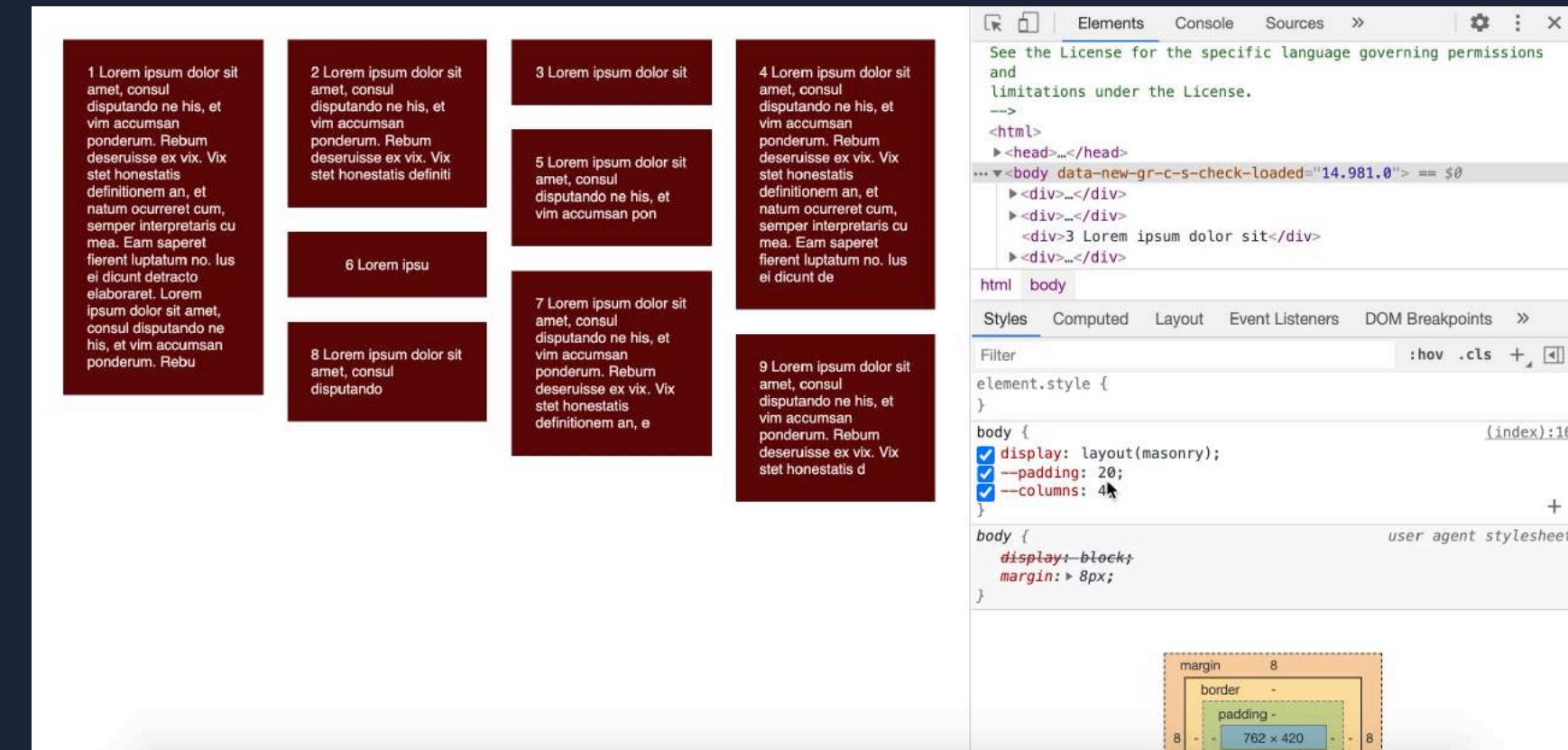
An extension point for the browser's layout rendering step where element dimensions, position, and alignment are determined.

- make your own display properties. Masonry, here we come!
- Able to polyfill layout specs like Container Queries

Layout API (High-level)

Demo:
<https://googlechromelabs.github.io/houdini-samples/layout-worklet/masonry/>

Read more:
<https://houdini.glitch.me/layout>



Animation API (High-level)

An extension point for browser's composite rendering step where layers are drawn to the screen and animated. The Animation API allows animation effects to be tied to user input like scrolling in a performant way

- Allows animations based on user input
- Allows performant parallax

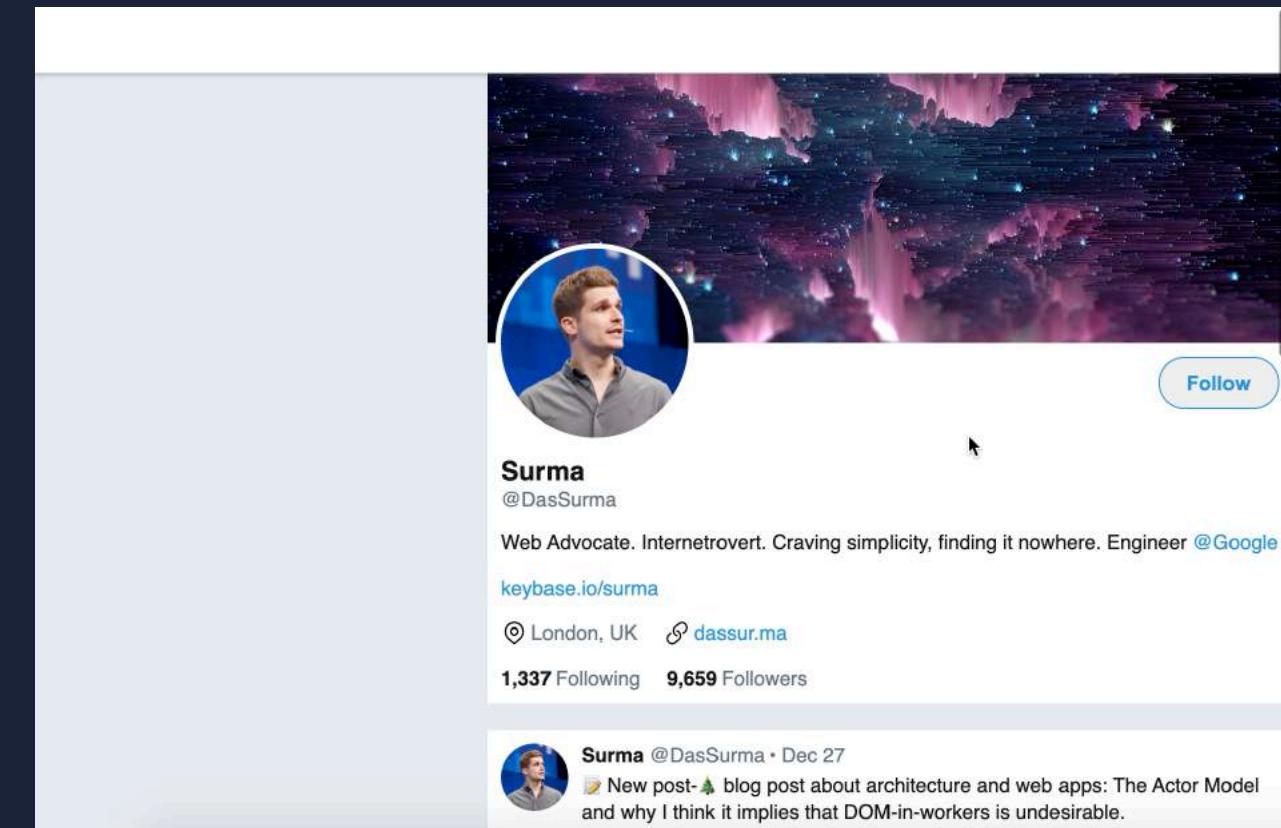
Animation API (High-level)

Demo:

<https://googlechromelabs.github.io/houdini-samples/animation-worklet/twitter-header/>

Read more:

<https://houdini.glitch.me/animation>



Houdini APIs - Low Level

The low level APIs form a foundation for high-level APIs. They are closely related to the parser extensions -> CSSOM, cascade

- Typed Object Model API
- Custom Properties & Values API
- Font Metrics API
- Parser API

	Google Chrome	Microsoft Edge	Opera	Samsung Internet	Mozilla Firefox	Apple Safari	Spec
Engine	Blink				Gecko	WebKit	-
Parser API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec
Properties & Values API (Demos Article)	Shipped (Chrome 78) Details	Shipped (Edge 79) Details	Shipped (Opera 65) Details	Shipped (Internet 12.0) Details	Under consideration Details	Partial support (Safari TP 67) Details	Working Draft Spec
Typed OM (Explainer Article)	Shipped (Chrome 66) Details	Shipped (Edge 79) Details	Shipped (Opera 53) Details	Shipped (Internet 9.2) Details	Under consideration Details	In Development Details	Working Draft Spec
Font Metrics API (Explainer)	No signal	No signal	No signal	No signal	No signal	No signal	Proposal Spec

Typed Object Model API (Low-level)

Before Houdini the only way for JavaScript to interact with CSS was by parsing CSS represented as string values and modifying them. The Typed OM gives shape and structure to CSS values that previously were simple strings. It's an extension to the existing CSS Object Model (CSSOM) that exposes CSS values as typed JavaScript objects, instead of a simple strings.

```
selectedElement.computedStyleMap().get("font-size");

{
  value: 20,
  unit: "px"
}

selectedElement.attributeStyleMap.set("font-size", CSS.em(2));
```

Read more: <https://houdini.glitch.me/typed-om>

Custom Properties & Values API (Low-level)

Custom Properties give shape, structure, and constraints to CSS Variables. The CSS Properties And Values API allows developers to extend CSS variables by adding a type, initial value and define inheritance.

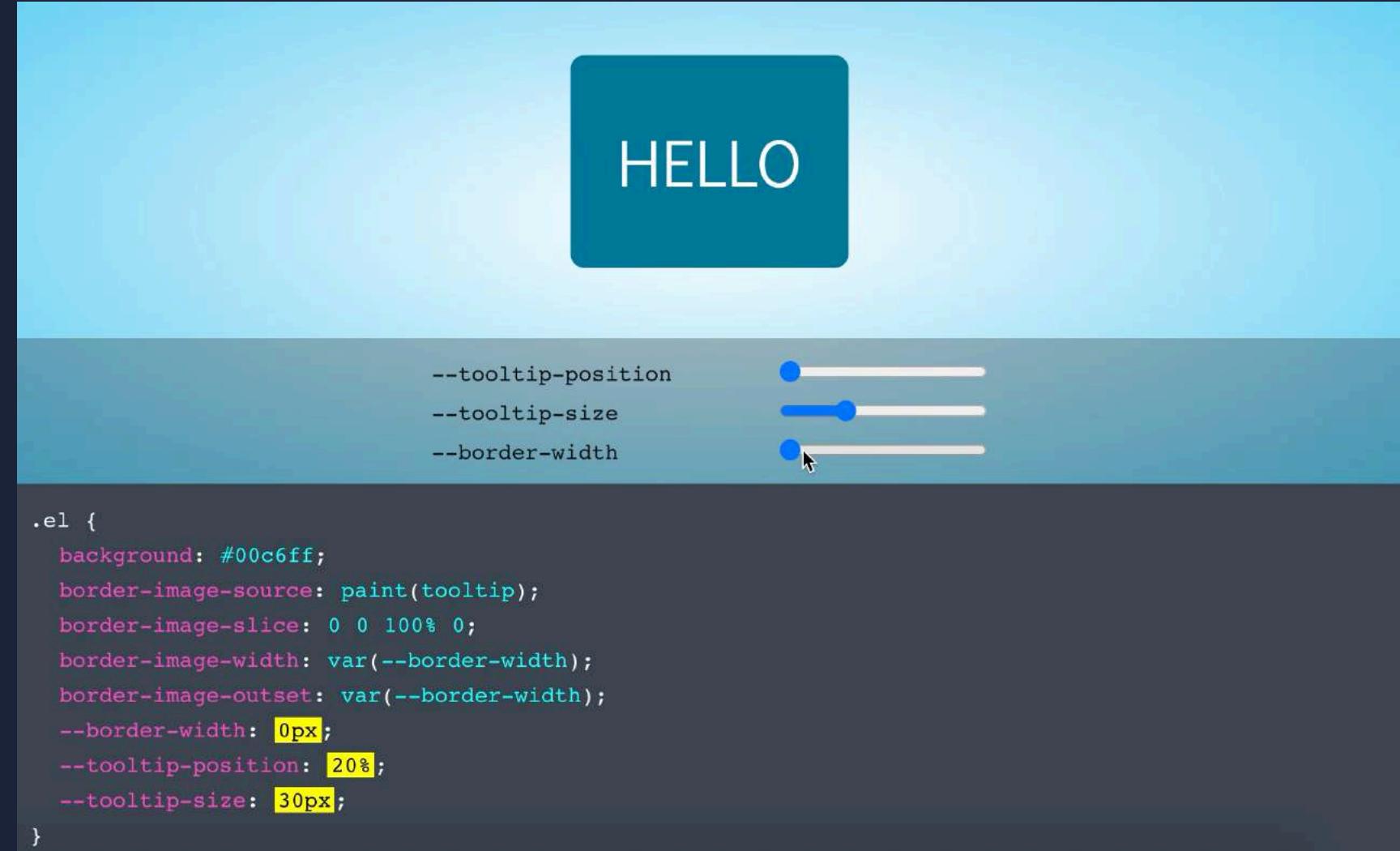
- Types are preserved between CSS & JS
- Good browser support already

```
CSS.registerProperty({  
  name: "--colorPrimary",  
  syntax: "<color>",  
  inherits: false,  
  initialValue: "blue",  
});
```

Custom Properties & Values API (low-level)

Demo:
<https://css-houdini.rocks/tooltip>

Read more:
<https://houdini.glitch.me/custom-properties>



Font Metrics API (Low-level)

(Very Early Stage) The Font Metrics API will provide methods for measuring dimensions of text elements that are being rendered on screen in order to allow developers to affect how text elements are being rendered on screen. Multi-line dynamic text truncation is an example of one of those features.

Worklets

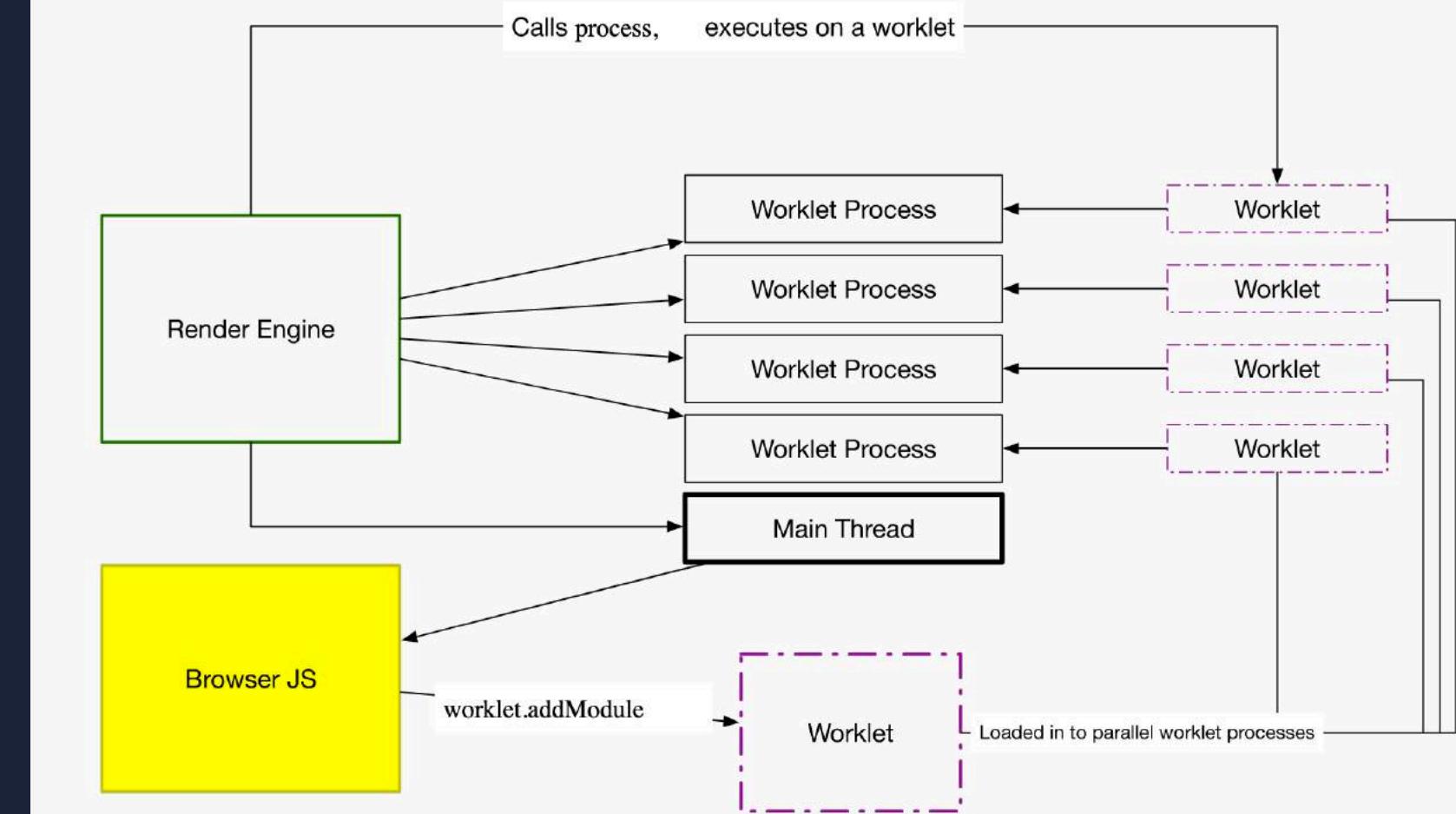
Worklets are extension points for rendering engines. Worklets are scripts that run during render and are independent of the main JavaScript environment. Conceptually, they're similar to Web Workers. They have very restricted access to the global scope.

Houdini introduces following Worklets to extend the browser render engine:

- Paint Worklet - Paint API
- Animation Worklet - Animation API
- Layout Worklet - Layout API

<https://houdini.glitch.me/worklets>

Worklet Lifecycle



The Paint API

The Paint API

What does it do?

The Paint API works very similar to the canvas drawing context. We can use JS to create custom drawing function that draw an image in CSS. We can then use this function for any CSS property that expects an image. You could use it for the background-image, the border-image or the list-style-image.

The Paint API

Why do wee need it?

- example: [extra.css](#)
- instead of polyfills like the [conic gradient polyfill](#)
- Reduce DOM nodes number,
e.g. when drawing lots of
particles
- create fancy painting features

A CSS [Houdini](#) library for
making your site a little
more [#extra](#).

How to use in CSS

```
.slanted-bg {  
  background-image: paint(slanted-bg);  
}
```

How to use in CSS - Test for support

```
.slanted-bg {  
    background: papayawhip;  
}  
  
@supports (background: paint(slanted-bg)) {  
    .slanted-bg {  
        background: paint(slanted-bg);  
    }  
}
```

How to register in JS

1. Declare a custom paint class
2. Register paint
3. Load worklet

How to register in JS

Declare class

```
class SlantedBackground {  
  paint(ctx, geom, props, args) {  
    // paint implementation  
  }  
}
```

How to register in JS

Register paint

```
registerPaint('slanted-bg', SlantedBackground);
```

How to register in JS

Load Worklet

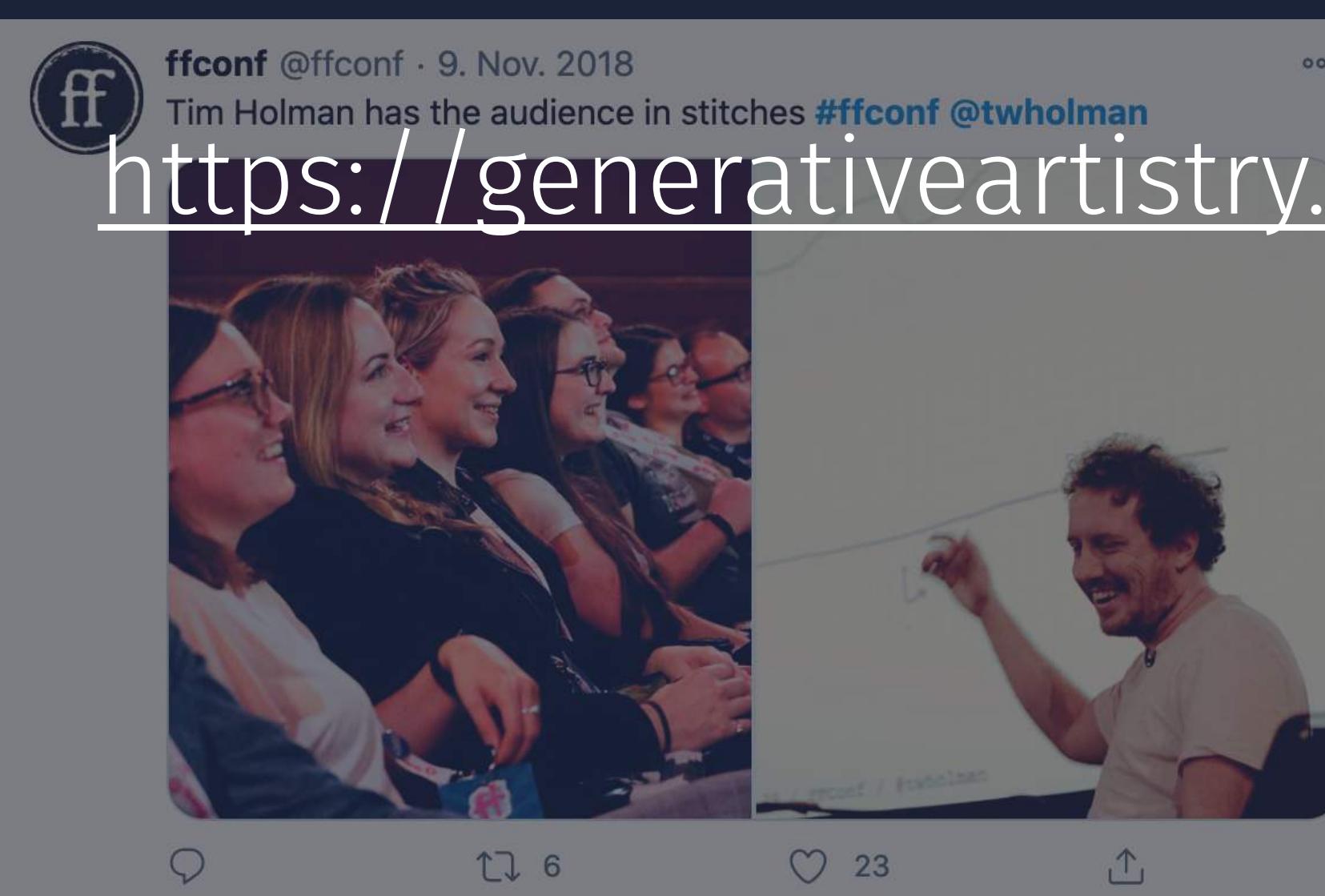
```
if ('paintWorklet' in CSS) {  
    CSS.paintWorklet.addModule('slanted-bg-module.js');  
}
```

Paint API Example



How my Houdini Journey started

FFConf - 19. Nov 18



ffconf @ffconf · 9. Nov. 2018

Tim Holman has the audience in stitches #ffconf @twholman

<https://generativeartistry.com>

generative artistry

tutorials · podcast · rss

Tutorials

A range of interactive tutorials, exploring ideas and techniques used in generative art. Progressively learn the steps used to create generative art.

[Get Started](#)

Podcast

Join Tim and Ruth as we delve into different aspects of generative art and the amazing community around it. Available everywhere you listen.

[Listen now](#)

By [ruth & tim](#) · [source](#)

6

23

↑

May 31, CSSconf EU



SAMUEL RICHARD

Design System Magic
with CSS Houdini

What is Houdini?

Can I use it?

CSSConf - 31. May 19
Extending CSS
with JS

Custom Properties

↳ `window.css.registerProperty`

- + make CSS variables smarter
- + make Real properties

LAYOUT API

★ LayoutFragment

★ `registerLayout('centered', {})`

★ build a Layout that puts
elements in the center

PAINT API

1. get property
2. draw a circle with API
3. register worklet
4. call worklet in CSS

★ animated circular
navigation menu

Lisi Linhart - @lisi_linhart - @storyblok



Samuel Richard

Design System Magic with CSS Houdini

How my Houdini Journey started

 Canvas & Houdini seem really similar

How my Houdini Journey started

<https://generativeartistry.com/tutorials/tiled-lines>

```
var canvas = document.querySelector('canvas');
var context = canvas.getContext('2d');

for(var x = 0; x < size; x += step) {
  for(var y = 0; y < size; y+= step) {
    var leftToRight = Math.random() >= 0.5;

    if(leftToRight) {
      context.moveTo(x, y);
      context.lineTo(x + width, y + height);
    } else {
      context.moveTo(x + width, y);
      context.lineTo(x, y + height);
    }

    context.stroke();
  }
}
```

How my Houdini Journey started

Let's try it in Houdini

```
paint(ctx, geom) {  
    for (let y = 0; y <= geom.height; y += size) {  
        for (let x = 0; x <= geom.width; x += size) {  
            let leftToRight = Math.random() >= 0.5;  
  
            if (leftToRight) {  
                ctx.moveTo(x, y + size);  
                ctx.lineTo(x + size, y);  
            } else {  
                ctx.moveTo(x, y);  
                ctx.lineTo(x + size, y + size);  
            }  
  
            ctx.stroke();  
        }  
    }  
}
```

How my Houdini Journey started

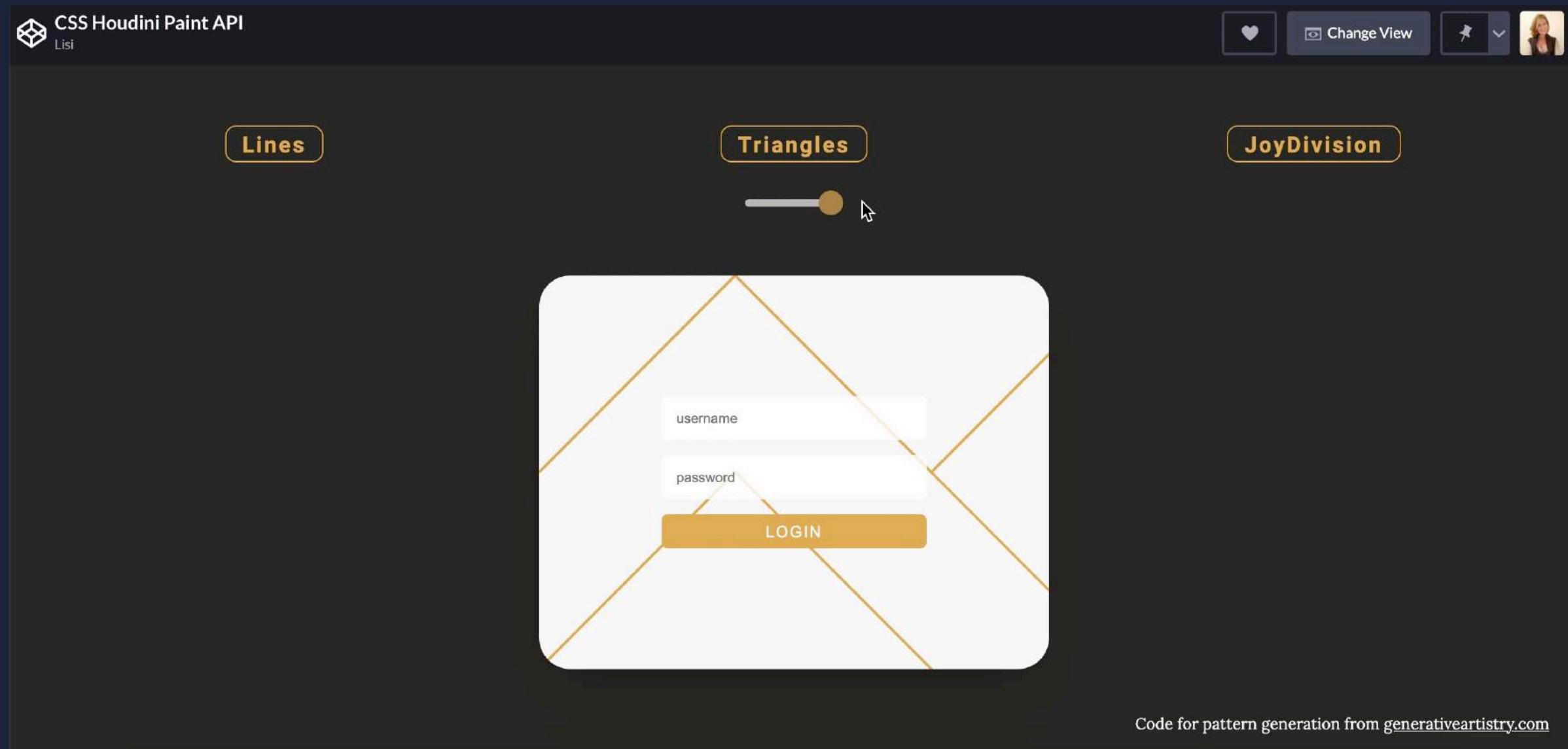
Lisi Linhart @lisi_linhart · 16. Jan.

Gotten around to play with CSS **Houdini**. I've taken some of the code snippets out of generativeartistry.com and created adaptive background image with the Paint API ⚡️ Pen at codepen.io/lisilinhart/pe... (Chrome only, enable flags via chrome://flags - Exp. Web Plat. Feat.) **#codepen**



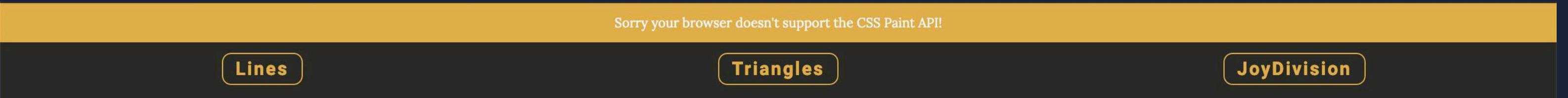
The screenshot shows a CodePen preview window. At the top, there are three tabs: "Lines" (highlighted in yellow), "Triangles", and "JoyDivision". Below the tabs is a slider. The main area displays a login form with a background featuring a complex, adaptive geometric pattern of triangles and lines in light blue and white. The form includes fields for "Username" and "Password", and a "LOGIN" button. At the bottom left, a timer indicates "0:21" and "5.794 Mal angezeigt". At the bottom right, a note says "Code for pattern generation from generativeartistry.com". Below the preview are social sharing icons: a speech bubble with "2", a retweet icon with "36", a heart icon with "155", an upvote arrow, and a share icon.

Houdini & Generative Art



<https://codepen.io/lisilinhart/pen/LYEJWQQ>

1. Register & Support



```
if (CSS.paintWorklet) {  
    CSS.paintWorklet.addModule('/lisilinhart/pen/LYEJWQQ.js');  
} else {  
    document.body.classList.add('warning');  
}
```

2. Use in CSS

```
.card {  
  background: var(--c-light);  
  background-image: paint(generative);  
  --type: var(--drawType, lines); /* Types: lines, joyDivision, triangles */  
  --size: 120;  
  --line-color: #E0AF53;  
  --line-width: 2;  
}  
  
input[value="lines"]:checked ~ .card {  
  --drawType: lines;  
}
```

3. JS - registerPaint - input

```
if (typeof registerPaint !== "undefined") {  
  registerPaint("generative", class {  
  
    paint(ctx, geom, properties) {  
  
      ...  
    }  
  });  
}
```

The JS - paint()

```
registerPaint("generative", class {
  static get inputProperties() {
    return ["--type", "--size", "--line-color", "--line-width"];
  }

  paint(ctx, geom, properties) {
    let drawType = String(properties.get("--type")).replace(' ', '');
    let size = Number(properties.get("--size"));
    let lineWidth = Number(properties.get("--line-width"));
    let lineColor = String(properties.get("--line-color"));
    ctx.lineWidth = lineWidth;
    ctx.strokeStyle = lineColor;

    const drawFunctions = {
      "lines": () => this.drawLines(ctx, geom.width, geom.height, size),
      "joyDivision": () => this.drawJoyDivision(ctx, geom.width, geom.height, size),
      "triangles": () => this.drawTriangles(ctx, geom.width, geom.height, size),
    };
    drawFunctions[drawType]();
  }
});
```

The JS - drawLines()

```
registerPaint("generative", class {
  static get inputProperties() { ... }

  paint(ctx, geom, properties) { ... }

  drawLines(ctx, width, height, size) {
    for (let y = 0; y <= height; y += size) {
      for (let x = 0; x <= width; x += size) {
        let leftToRight = Math.random() >= 0.5;

        if (leftToRight) {
          ctx.moveTo(x, y + size);
          ctx.lineTo(x + size, y);
        } else {
          ctx.moveTo(x, y);
          ctx.lineTo(x + size, y + size);
        }

        ctx.stroke();
      }
    }
  }
});
```

How does it perform compared to current CSS features?

I wrote [an article](#) about it:

Performance depends on various factors:

- Painting performance between current features and Houdini Paint Examples are quite similar
- Painting performance depends on your Houdini module complexity
- CSS Houdini adds the overhead of loading the module
- Consider adding a CSS fallback until it's loaded (progressive enhancement!)



Conclusion

What is the CSS Houdini all about?

CSS Houdini allows us to hook into the browser rendering process, so we can develop various CSS features that can be easily shared, implemented and, potentially, added to CSS specification itself.

What are the advantages

- close access to the CSSOM and parser
- better performance than JS polyfills
- great for features that depend on requestAnimationFrame or scroll interactions
- build any CSS feature you want

What is the Paint API all about?

- Extend the browser render engine in various ways
- Not all features have wide browser support:
ishoudinreadyyet?
- Allows you to easily include and adapt features that previously were not possible in CSS
- great for reducing DOM nodes
- no need for JS polyfills, we can just implement our own CSS features!

What to keep in mind

- Houdini is an experimental technology and is not production ready yet
- always check for browser support before implementing any styles using this technique
- be careful with animated effects and not do overdo it
- great for progressive enhancement

How do I get started?

How do I get started?



<https://houdini.glitch.me>

How do I get started?

The screenshot shows a GitHub repository page for 'CSSHoudini/awesome-css-houdini'. The page has a dark theme. At the top, there's a navigation bar with icons for red, yellow, and green circles, followed by the repository name 'CSSHoudini/awesome-css-houdini' and a '+' button. Below the navigation is a header with back, forward, and refresh buttons, and the URL 'github.com/CSSHoudini/awesome-css-houdini'. The main content area is divided into two sections: 'Podcasts' and 'Samples'.

Podcasts

Podcast	Speaker
Houdini Reloaded	Una & Chris Dhanaraj
Modern Web Podcast	Surma, Aimee Knight, Vitalii Bobrov

Samples

Samples	APIs
CSS Houdini 😎	Worklets Typed OM Custom Properties & Values API Paint API Animation Worklet Layout API
CSS Houdini Experiments 😎	Custom Properties & Values API Paint API Layout API
Houdini Samples (Google Chrome Labs)	Custom Properties & Values API Paint API Animation Worklet Layout API
Houdini's CSS Paint Polyfill	Paint API
Using Houdini CSS Paint API with Rough.js	Paint API

<https://github.com/nucliweb/awesome-css-houdini>

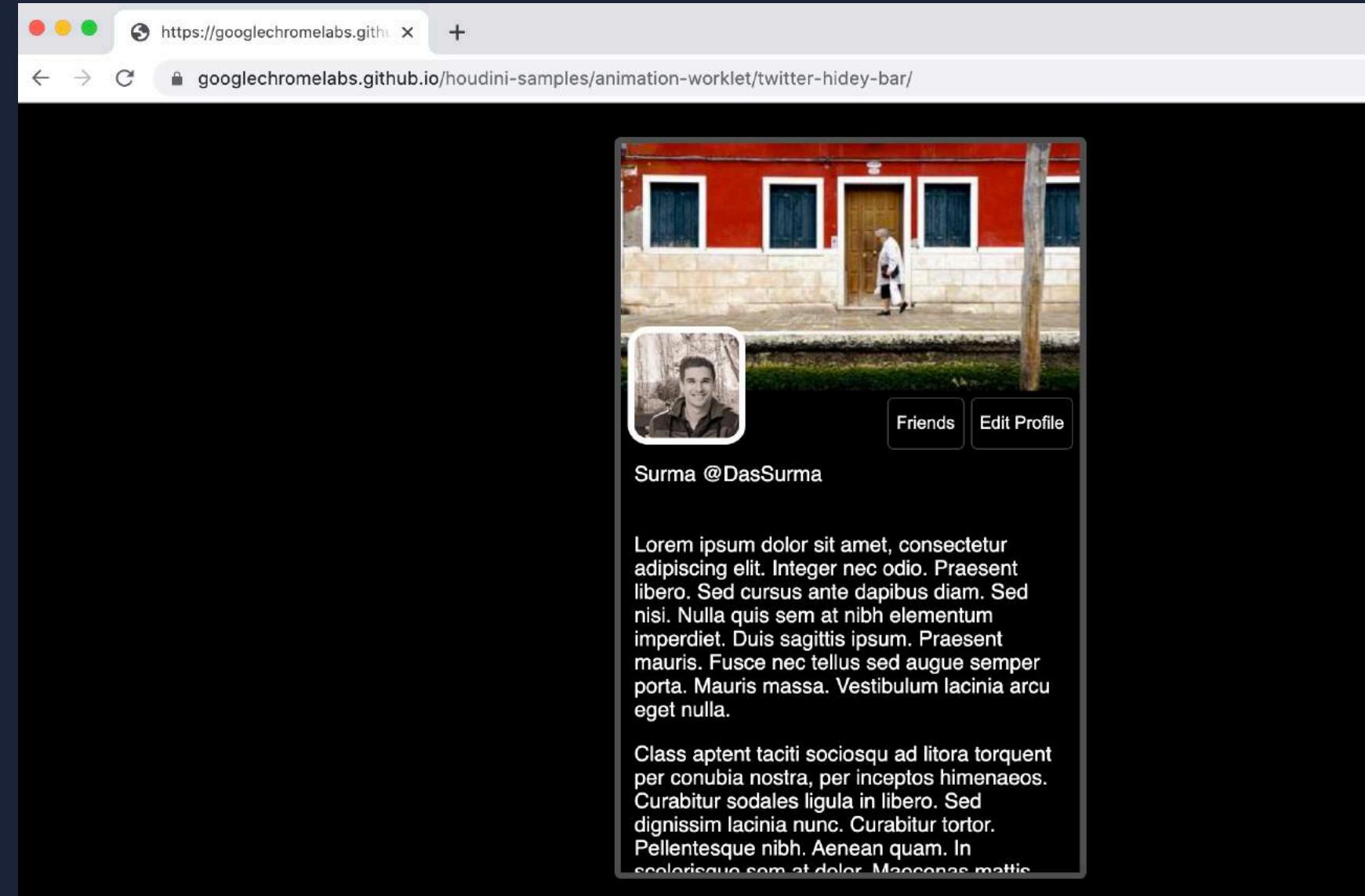
How do I get started?

The screenshot shows a web browser window with the title bar "CSS Houdini Experiments - @iamvdo" and the address bar "css-houdini.rocks". The main content area is titled "CSS Houdini Experiments" and contains the following text: "Here is my experiments with the latest CSS Houdini spec. Some demos require Chrome with Web Platform flag enabled. See [support](#)". Below this, there are links to "Source code on GitHub" and "Learn Houdini". A section titled "Other resources" lists several links: "Houdini on Smashing Mag", "Houdini and the paint API", "Working with the new CSS Typed Object Model", and "Awesome CSS Houdini". Another section titled "Specification links" lists: "CSS Properties & Values API", "CSS Typed OM", "CSS Painting API", and "CSS Layout API". On the left side of the page, there is a sidebar with categories and links:

- BACKGROUNDS**
 - Corners gradient •
 - Slanted backgrounds •
 - Checkboxes ••
 - Background properties ••
 - JS-in-CSS ••
- BORDERS**
 - Simple tooltip ••
 - LAST Inner borders •
 - Rough boxes ••
 - Non-rectangular separators ••
- MASKS**
 - Corner shapes ••
 - Avatar polygons ••
 - Smooth corners ••
 - Dynamic hover masks ••
 - Random bubbles mask ••
- ANIMATIONS**
 - Animating a gradient •
 - Ripple animation •
 - Highlighter marker annotations ••

<http://css-houdini.rocks>

How do I get started?



<https://googlechromelabs.github.io/houdini-samples/>

Articles

→ [CSS Tricks - Paint API](#)

→ [Logrocket - Paint API](#)

→ [Google Developers - Houdini](#)

→ [Google Developers - Paint API](#)

→ [Smashing Magazine - CSS Houdini](#)

The End

Thank you for joining!

→ My blog

→ My twitter

→ My Codepen

→ Storyblok



storyblok

Storyblok is a headless CMS, that helps your team to tell your story and manage content for every use-case: corporate websites, e-commerce, helpdesks, mobile apps, and screen displays.

storyblok.com

The screenshot shows the Storyblok CMS interface. On the left, a preview window displays a blog post titled "We are Going to Mars" with a subtext about Mars. On the right, the content editor interface is visible, showing a sidebar with navigation tabs (Content, Status, Config, Tools) and a main panel for managing content blocks. A sidebar on the right lists content items like "Principal Blog Post" and "Blog Post Section". A red circle labeled "1" is at the top right of the preview window, "2" is on the left edge of the preview window, and "3" is on the right edge of the sidebar.