CSS Renaissance

From the Middle Age to modern CSS

— Custom Properties
— Shadow DOM
— Houdini (Paint API)

Custom Properties

Runtime CSS *variables*

Or

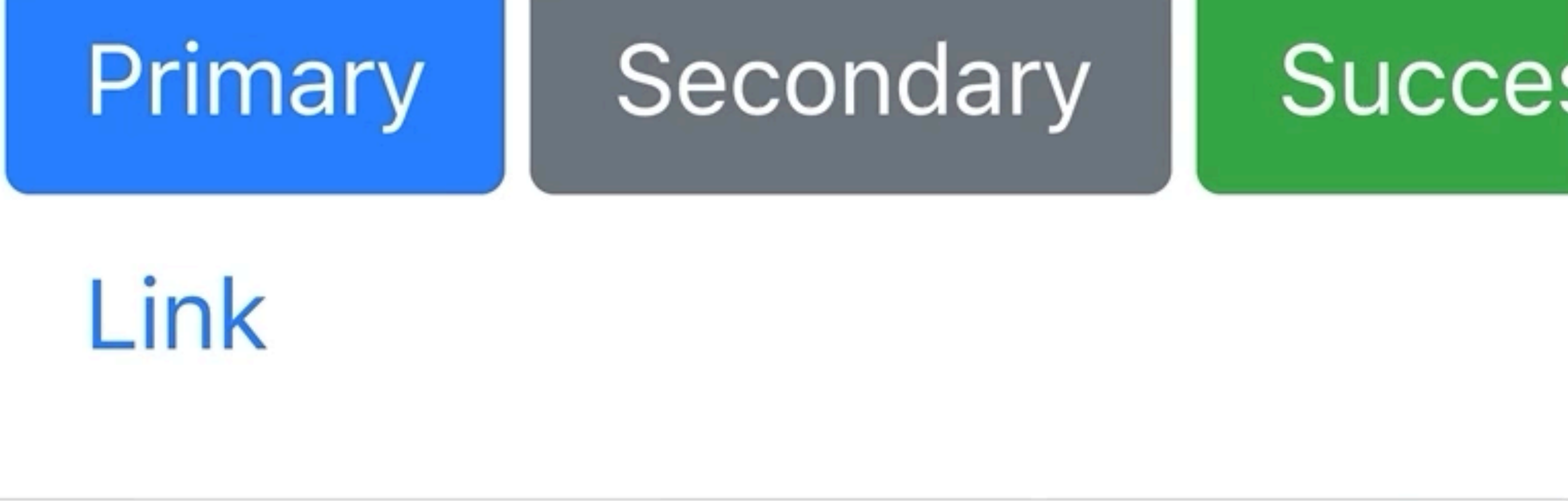
Inherited user-defined *properties*

Runtime CSS Variables

```css
:root {
  --primary: #007bff;
}

.btn-primary {
  color: var(--primary, deepskyblue);
}
```

Runtime CSS Variables

```css
:root { --primary: #007bff; }

.btn-primary { color: var(--primary, deepskyblue); }

.btn-primary:hover {
  --primary: crimson;
}
```



Elements Console Sources Network Performance Memory Application Security Audits Redux 1

```
<!doctype html>
<html lang="en" class="gr__getbootstrap_com">
▶<head>…</head>
▼<body data-gr-c-s-loaded="true">
  ▶<a id="skippy" class="sr-only sr-only-focusable" href="#content">…</a>
  ▶<header class="navbar navbar-expand navbar-dark flex-column flex-md-row bd-navbar">…
  </header>
  ▼<div class="container-fluid">
    ▼<div class="row flex-xl-nowrap">
      ▶<div class="col-12 col-md-3 col-xl-2 bd-sidebar">…</div>
      ▶<div class="d-none d-xl-block col-xl-2 bd-toc">…</div>
      ▼<main class="col-12 col-md-9 col-xl-8 py-md-3 pl-md-5 bd-content" role="main">
        <h1 class="bd-title" id="content">Buttons</h1>
        ▶<p class="bd-lead">…</p>
        <script async src="https://cdn.carbonads.com/carbon.js?
        serve=CKYIKKJL&placement=getbootstrapcom" id="_carbonads_js"></script>
        ▶<h2 id="examples">…</h2>
```

Styles Computed Event Listeners »

Filter :hov .cls

```
--pink: #e83e8c;
--red: #dc3545;
--orange: #fd7e14;
--yellow: #ffc107;
--green: #28a745;
--teal: #20c997;
--cyan: #17a2b8;
--white: #fff;
--gray: #6c757d;
--gray-dark: #343a40;
--primary: #007bff;
--secondary: #6c757d;
--success: #28a745;
--info: #17a2b8;
--warning: #ffc107;
```

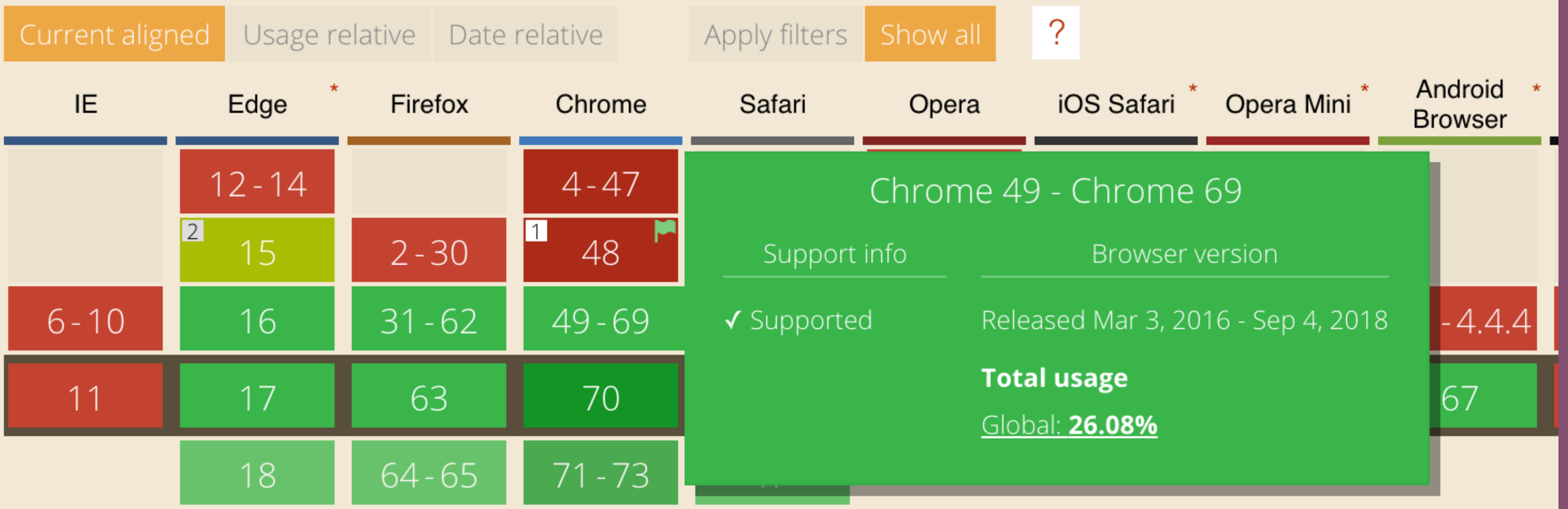
Inherited user-definied properties [ref]

```
/* CSS */
div > p { --primary: crimson; }
p { --primary: aqua; }

.c-block { background-color: var(--primary); }

<!-- HTML -->
<div>
  <p class="c-block"></p>
</div>
```

[ref] glazman: CSS Variables, why we drop the $foo notation

CSS Variables (Custom Properties) - CR

Permits the declaration and usage of cascading variables in stylesheets.



postcss-css-variables
postcss-custom-properties

```
:root {
  --color: red;
}

h1 {
  color: var(--color);
}

/* becomes */

:root {
  --color: red;
}

h1 {
  color: red;
  color: var(--color);
}
```

Comparison

```
:root { --backgroundColor: red; }

.header { background-color: var(--backgroundColor, white); }

.header:hover { --backgroundColor: orange; }

.header.is-about-page { --backgroundColor: yellow; }
```

postcss-custom-properties

```
:root { --backgroundColor: red; }

.header {
  background-color: red;
  background-color: var(--backgroundColor, white);
}

.header:hover { --backgroundColor: orange; }

.header.is-about-page { --backgroundColor: yellow; }
```

postcss-css-variables

preserve: 'computed'

```
:root { --backgroundColor: red; }

.header {
  background-color: red;
  background-color: var(--backgroundColor, white);
}

.header:hover { background-color: orange; }

.header:hover { --backgroundColor: orange; }

.header.is-about-page { --backgroundColor: yellow; }
```

```css
:root { --width: 100px; }

@media (max-width: 1000px) {
    :root { --width: 200px; }
}

.box { width: var(--width); }
```

```css
:root {
  --width: 100px;
}

.box {
  width: 100px;
}

@media (max-width: 1000px) {
  .box {
    width: 200px;
  }
}

.box { width: var(--width); }

@media (max-width: 1000px) {
  :root {
    --width: 200px;
  }
}
```

Stripping custom properties

If you can preprocess custom properties and get what you expect, stick with preprocessor variables.

— postcss-simple-vars

Interoperability

— Sass/Less
— React/Angular/Vue
— hyperHTML
— styled-components
— CSS-in-JS: linaria

Values can be any valid CSS value: numbers, strings, lengths, colors, etc.

```
/* CSS */
:root {
  --foo: console.log('Hello world');
}

// JS
const styles = window.getComputedStyle(document.documentElement);
const value = styles.getPropertyValue('--foo');
eval(value);
```

i18n [1]

```
:root,
:root:lang(en) {
  --external-link: "external link";
}

:root:lang(de) {
  --external-link: "externer Link";
}

:root:lang(it) {
  --external-link: "Link esterno";
}

a[href^="http"]::after {content: " (" var(--external-link) ")"}
```

[1] publishing-project.rivendellweb.net

Operations

```
:root {
  --columns: 12;
  --gutter: 16px;
}

.o-col {
  margin: 0 calc(var(--gutter) * 2);
  width: calc(100% / var(--columns));
}
```

Operations

```
:root {
  --columns: 12;
  --gutter: 16px;
  --margin: (var(--gutter) * 2)
}

.o-col {
  margin: 0 calc(var(--margin));
}
```

Operations

```
:root {
  --alpha-hover: 0.04;
  --primary: 98, 0, 238;
}

.c-button:hover {
  background-color: rgba(var(--primary), var(--alpha-hover))
}
```

Operations

```
:root {
  --animation-duration-simple: 0.1s;
  --easing-standard: cubic-bezier(0.4, 0.0, 0.2, 1);
}

.c-box {
  transition:
              all
              var(--animation-duration-medium)
              var(--easing-standard);
}
```

Separate logic from design

Codepen

— All the logic at the top of the document
— See changing property

Custom properties in JS

```
<button style={{ '--primary': colors.primary }}></button>
```

Vanilla JS

```
const getVariable = (el, propertyName) => {
  const styles = window.getComputedStyle(el);

  return String(styles.getPropertyValue(propertyName)).trim();
};

const setDocumentVariable = (propertyName, value) => {
  document.documentElement.style.setProperty(propertyName, value);
};
```

Speed factor: 1

Codepen

Speed factor: 1

Codepen

Complaints

1. Syntax is ugly and verbose
2. Sass/Less already have variables

Preprocessor vs CSS Variables [2]

— Sass variables are static and lexically scoped
— CSS variables are live and scoped to the DOM

[2] philipwalton: Why I'm Excited About Native CSS Variables

What preprocessor cannot do

1. Interact with Javascript or 3rd party stylesheets
2. Aware of the DOM or CSSOM
3. Be changed dynamically
4. Cascade
5. Inherit

Responsive properties with media queries

```
$gutterSm: 1em;
$gutterMd: 2em;
$gutterLg: 3em;

.o-container {
  padding: $gutterSm;
}

@media (min-width: 30em) {
  .o-container {
    padding: $gutterMd;
  }
}

@media (min-width: 48em) {
  .o-container {
    padding: $gutterLg;
  }
}
```

Responsive properties with media queries

```
:root { --gutter: 1.5em; }

.o-container {
  padding: var(--gutter);
}

@media (min-width: 30em) {
  :root { --gutter: 2em; }
}
@media (min-width: 48em) {
  :root { --gutter: 3em; }
}
```

Responsive modular scale

```
:root {
  --base-font-size: 1em;
  --modular-scale: 1.2;
}
p { font-size: var(--base-font-size); }
h1 { font-size: calc(var(--modular-scale) * 3 * var(--base-font-size)); }
h2 { font-size: calc(var(--modular-scale) * 2 * var(--base-font-size)); }

@media (min-width: 30em) {
  :root { --modular-scale: 1.333 }
}
@media (min-width: 48em) {
  :root { --modular-scale: 1.414 }
}
```

Reusable and extensible components

```html
<header class="header">
  <button class="c-button c-header-button"></button>
</header>
```

```css
.c-button {
  background-color: #eee;
  border: 2px solid crimson;
  color: crimson;
  font-size: 18px;
}

.c-header-button {
  background-color: #333;
  border: 2px solid aqua;
  color: aqua;
  font-size: 24px;
}

/* Or worse ... */
.header .c-button {}
```

Reusable and extensible components

```
.c-button {
  --btn-bg-color: #eee;
  --btn-primary-color: crimson;
  --btn-font-size: 18px;

  background-color: var(--btn-bg-color);
  border: 2px solid var(--btn-primary-color);
  color: var(--btn-primary-color);
  font-size: var(--btn-font-size);
}

.c-header-button {
  --btn-bg-color: #333;
  --btn-primary-color: aqua;
  --btn-font-size: 24px;
}
```

Component styling API [3]

API: Application programming interface [4]

“By abstracting the underlying implementation and only exposing objects or actions the developer needs, an API simplifies programming.”

[3] mrmrs: Component styling API

[4] Wikipedia - API

Theming

The act of laying a veneer over the top of an already styled website: an optional extra which alters or customises the UI

UK closer to delivering Brexit, says May

But the opposition says the proposed deal would leave the UK in an "indefinite half way house".

1h | UK Politics | 3557

- Kuenssberg: What next?
- A brief guide to where we are
- Adler: EU gives May space

LIVE
Brexit negotiations latest

3m DUP's Foster in London over Brexit plans

9m 'Moment of calm' urged over Brexit deal

9m Welsh government 'not briefed on Brexit plan'

Israel defence minister resigns over Gaza

Avigdor Lieberman denounces the cabinet's decision to accept a ceasefire as "surrendering to terror".

1h | Middle East

Caravan migrants reach US border

More than three hundred people are the first to arrive in Tijuana, Mexico, hoping to start new lives in the US.

2h | Latin America & Caribb…

Outcry over teen's underwear in rape trial

An Irish MP displays a thong in parliament after a defence lawyer criticises a teenager's underwear.

3h | Europe

Grieving Lion Air bride takes photos alone

The young woman says her partner had joked she should take the photos alone should he be delayed.

2h | Asia

Calgary votes against Winter Olympics bid

6h | US & Canada

'Sultan of Coins' executed in Iran

3h | Middle East

▶ Video manipulation: 'I never said that'

15h | Entertainment & Arts

Melania calls for White House aide's firing

17h | US & Canada

Hope for mountain gorilla and fin whale

2h | Science & Environment

UN to lift nine-year Eritrea sanctions

1h | Africa

Paraplegic man 'made to urinate in bottle'

23m | Australia

Call of Duty death prankster pleads guilty

2h | Technology

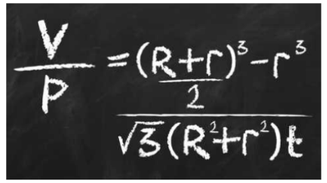
LIVE ATP Finals - Djokovic v Zverev

Novak Djokovic faces Alexander Zverev in the round-robin stage of the ATP Finals in London - watch BBC Two coverage and follow live text commentary.

Tennis

Fulham sack Jokanovic and appoint ex-Leicester boss Ranieri

3h | Football | 532



Is your brain hurting with ATP finals permutations?

15h | Tennis

Schumacher's record has never been a target - Hamilton

59m | Formula 1

Solari gets Real Madrid job on permanent basis

19h | European Football | 158

Watch: Snowboarder Gasser makes history by landing first cab triple 1260

15h | Winter Sports

Buttler & Curran lead England fightback in Sri Lanka

2h | Cricket | 139

Football Scores

Int Friendlies | More

WEDNESDAY 14TH NOVEMBER 2018

Switzerland 18:00 Qatar

View all International Friendlies scores

Insight

How damaging are the Man City Football Leaks allegations?

Football

'The Moliwood blockbuster' - it's Molinari v Fleetwood in the Race to Dubai

Golf | 32

Who's the greatest F1 driver of the 21st century?

Formula 1 | 367

'Underhill must change to be a

Variant Theming

```
<nav class="c-navigation c-news-navigation"></nav>
```

```
/* Navigation.css */
:host {
  --navigation-bg: var(--primary)
}

.c-navigation {
  background-color: var(--navigation-bg);
}
```

```
/* NewsNavigation.css */
.c-news-navigation {
  --navigation-bg: darkred;
}
```

Static Theming

```css
/* settings.css */
:root {
  --main-color: #1b70de;
  --bg-color: #FFF;
  --text-color: #000;
  --button-color: rgba(0, 0, 0, 0.8);
  --header-color: #424242;
}

:root.dark {
  --main-color: darkblue;
  --bg-color: #333;
  --text-color: white;
  --button-color: black;
  --header-color: #333;
}
```

Cambia la foto dell'intestazione

Cambia la foto del profilo

Tweet 1.376 | Following 533 | Follower 508 | Mi piace 5.203 | Altri

Annulla | Salva modifiche

Jiayi Hu

@jiayi_ghu

22 y/o. Addicted Book Reader & Tardis traveller. Working smart as web developer/consultant.

Padova, Italia

https://github.com/jiayihu/

Mostra quando sono IN DIRETTA

Colore del tema

01579b

Tweet | Tweet e risposte | Contenuti

Tweet fissato

Jiayi Hu @jiayi_ghu · 10 dic 2017

🔥 Finally had the time to finish react-tiny-dom, a minimal implementation of react-dom using react-reconciler APIs: github.com/jiayihu/react-…
I think it could very useful to see how to implement a custom renderer in React 16 @reactjs @dan_abramov

Traduci il Tweet

User theme

```
:root {
  --user-color: #01579B;
}

.u-textUserColor,
.u-borderUserColor {
  color: var(--user-color) !important;
}
```

Theming with JS

```
class App extends Component {
  constructor(props) {
    super(props);
    const theme = {
      '--main-color': '#1b70de',
      '--bg-color': '#FFF',
      '--text-color': '#000',
      '--button-color': 'rgba(0, 0, 0, 0.8)',
      '--header-color': '#424242',
    };
    this.state = { theme };
  }

  render() {
    return (
      <div>
        <RootCSSVariables variables={this.state.theme} />
        {...}
      </div>
    );
  }
}
```

Theming with JS

```
class RootCSSVariables extends Component {
  componentDidMount() {
    this.updateCSSVariables(this.props.variables);
  }

  componentDidUpdate(prevProps) {
    if (this.props.variables !== prevProps.variables) {
      this.updateCSSVariables(this.props.variables);
    }
  }

  updateCSSVariables(variables) {
    Object.keys(variables).forEach((key) => {
      const value = variables[key]
      document.documentElement.style.setProperty(key, value));
    });
  }

  render() { return this.props.children }
}
```

Theming with Sass

```
/* Navigation.css */
:host {
  /* --navigation-bg: var(--primary); */
  --navigation-bg: $primary;
}

.c-navigation {
  background-color: var(--navigation-bg, $primary);
}
```

Theming with Sass

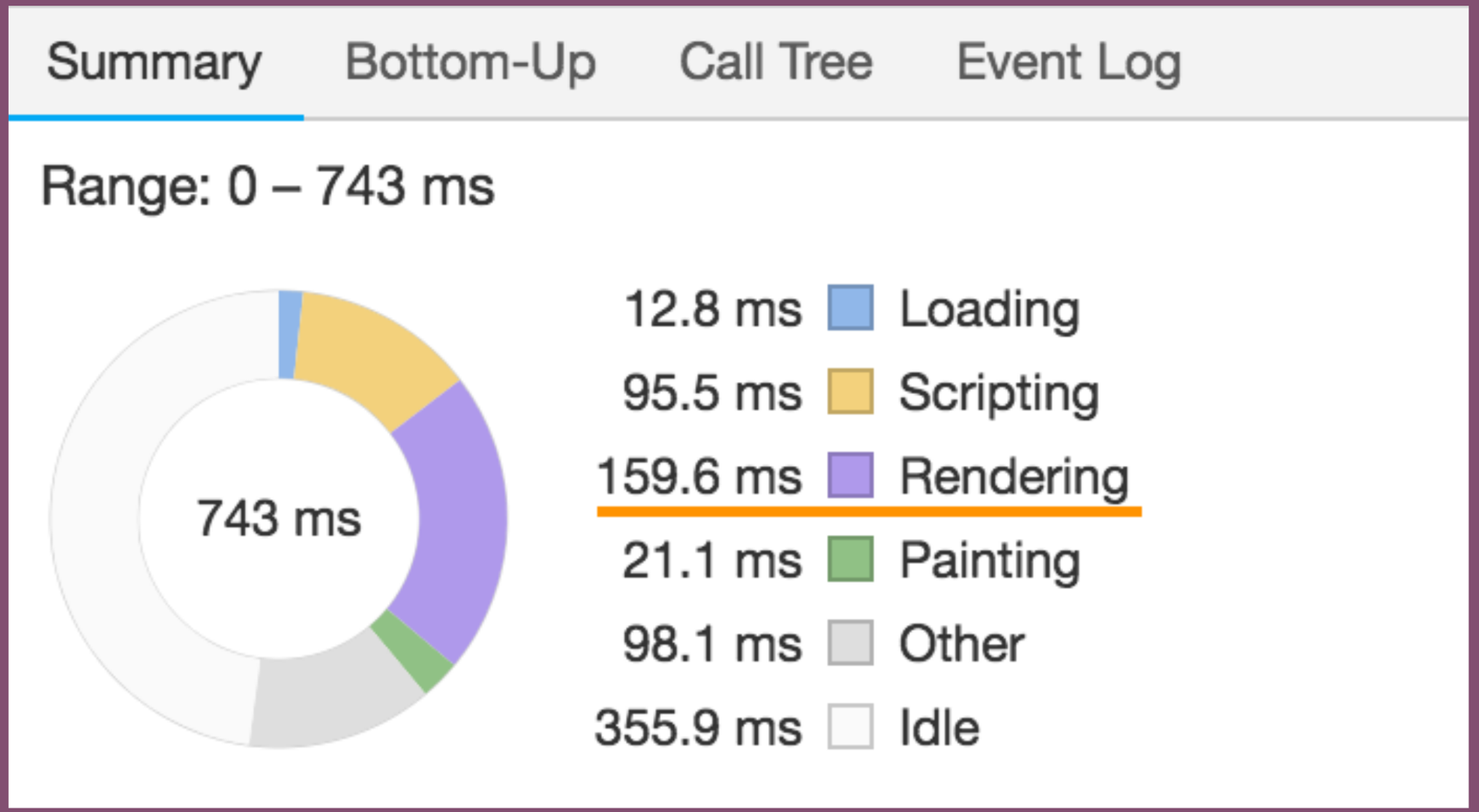
```
/* Navigation.css */
:host {
  --navigation-bg: crimson;
}

.c-navigation {
  background-color: crimson;
  background-color: var(--navigation-bg, crimson);
}
```

— Encapsulation and theming - Maxart

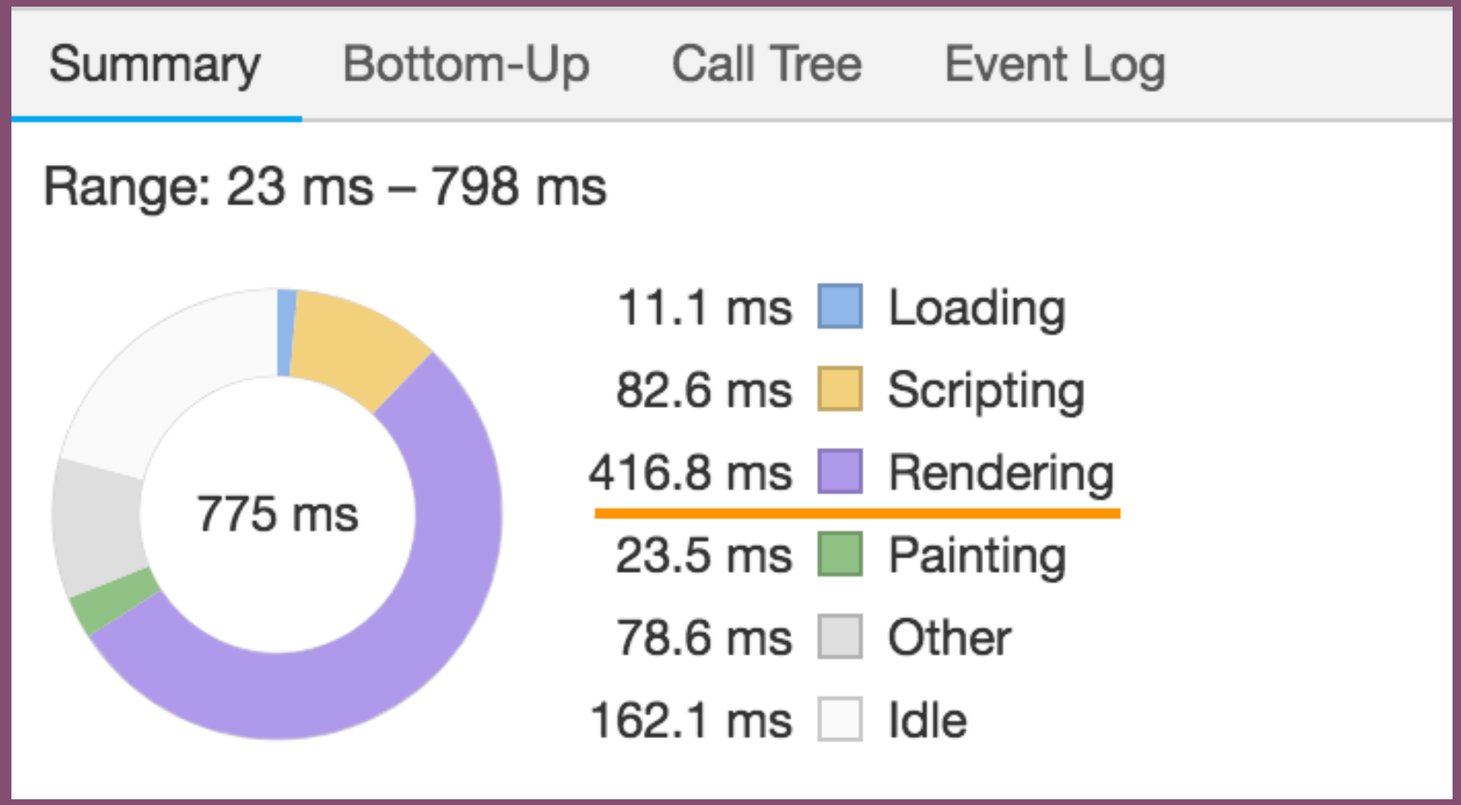
Runtime performance (25k nodes)

1. Start-up performance: *3x slower*
2. Style-recalculation: avoid frequent :root changes
3. Setting with JS
   — `el.setProperty('--color', 'green')` *4x slower* than
   — `el.setProperty('color', 'green')`
4. *1.3x faster* than inline styles

Start-up performance



Start-up performance [5]



[5] jiayihu: CSS Custom Properties performance in 2018

Recommendation

Use preprocessor for global static variables,
CSS custom properties for component styling API and theming.

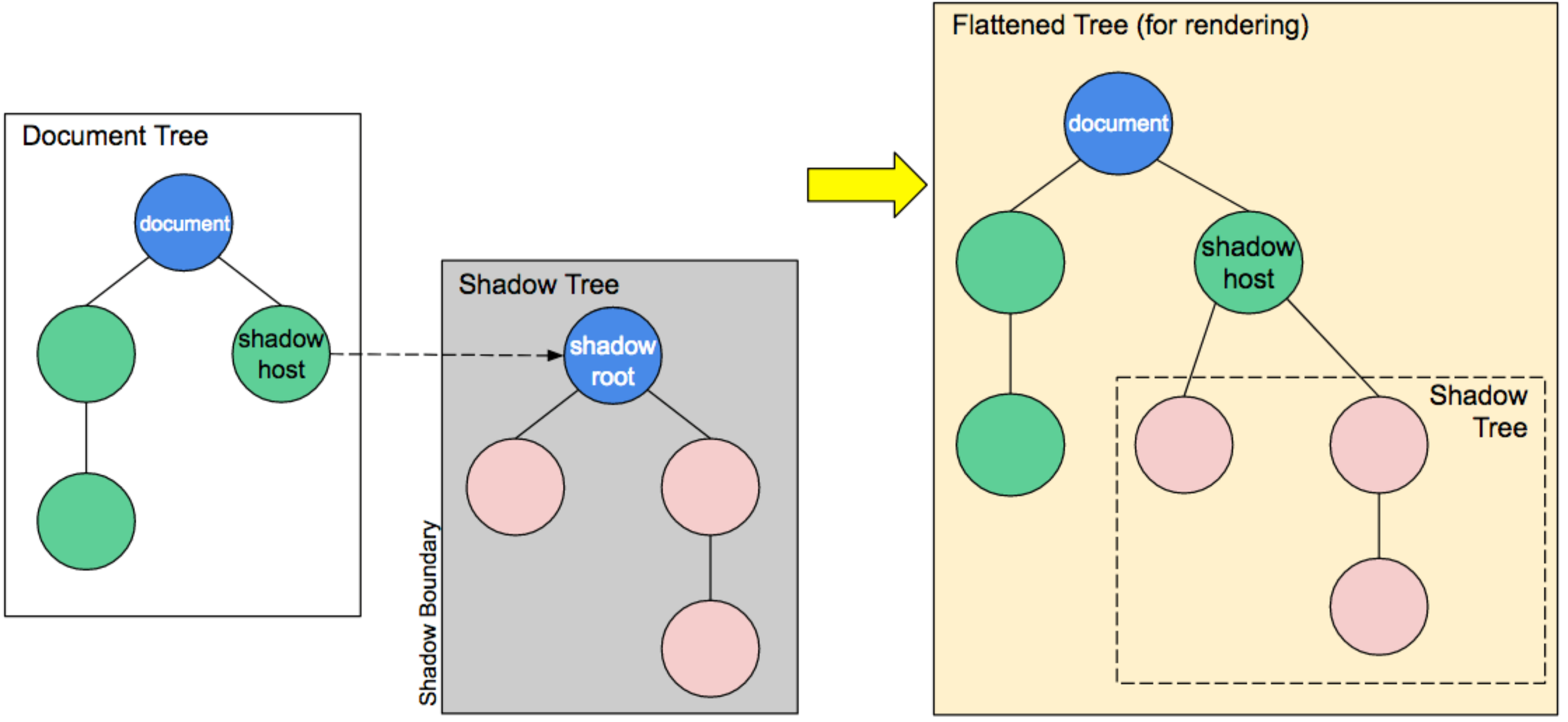
About Custom Properties

It's like when OOP was first invented.

Shadow DOM

Shadow DOM

— Part of Web Components
— Used by native DOM elements
— Similar to `iframe`


Document Tree
document
shadow host
Shadow Tree
shadow root
Shadow Boundary
Flattened Tree (for rendering)
document
shadow host
Shadow Tree

— A boundary between the developer and the browser implementation

```
<input id="foo" type="range">
```

— <video>, <select> etc.

```
const hostEl = document.querySelector('.host');

const shadowRoot = hostEl.attachShadow({ mode: 'open' });
shadowRoot.innerHTML = `
  <style>
    p {
      color: red;
    }
  </style>

  <p>Element with Shadow DOM</p>
`;
```

Codepen

```
<c-button class="c-header-button"></c-button>

.c-header-button {
  --btn-bg-color: #333;
  --btn-primary-color: aqua;
  --btn-font-size: 24px;
}
```

```
<c-navigation class="c-news-navigation"></c-navigation>

/* NewsNavigation.css */
.c-news-navigation {
  --navigation-bg: darkred;
}
```

```
const styles = `:host { background-color: ${props.theme} }`;

return (
    <ShadowDOM>
      <style>{styles}</style>
      <div>
        <h1>Calendar for {props.date}</h1>
      </div>
    </ShadowDOM>
);
```

ReactShadow

Encapsulation

Encapsulation is used to hide the values or state of a structured data object inside a class, preventing unauthorized parties direct access to them.
Publicly accessible methods are generally provided in the class.

Encapsulation

— Isolated DOM
— Re-targeted events
— Scoped CSS
— Simplify CSS selectors

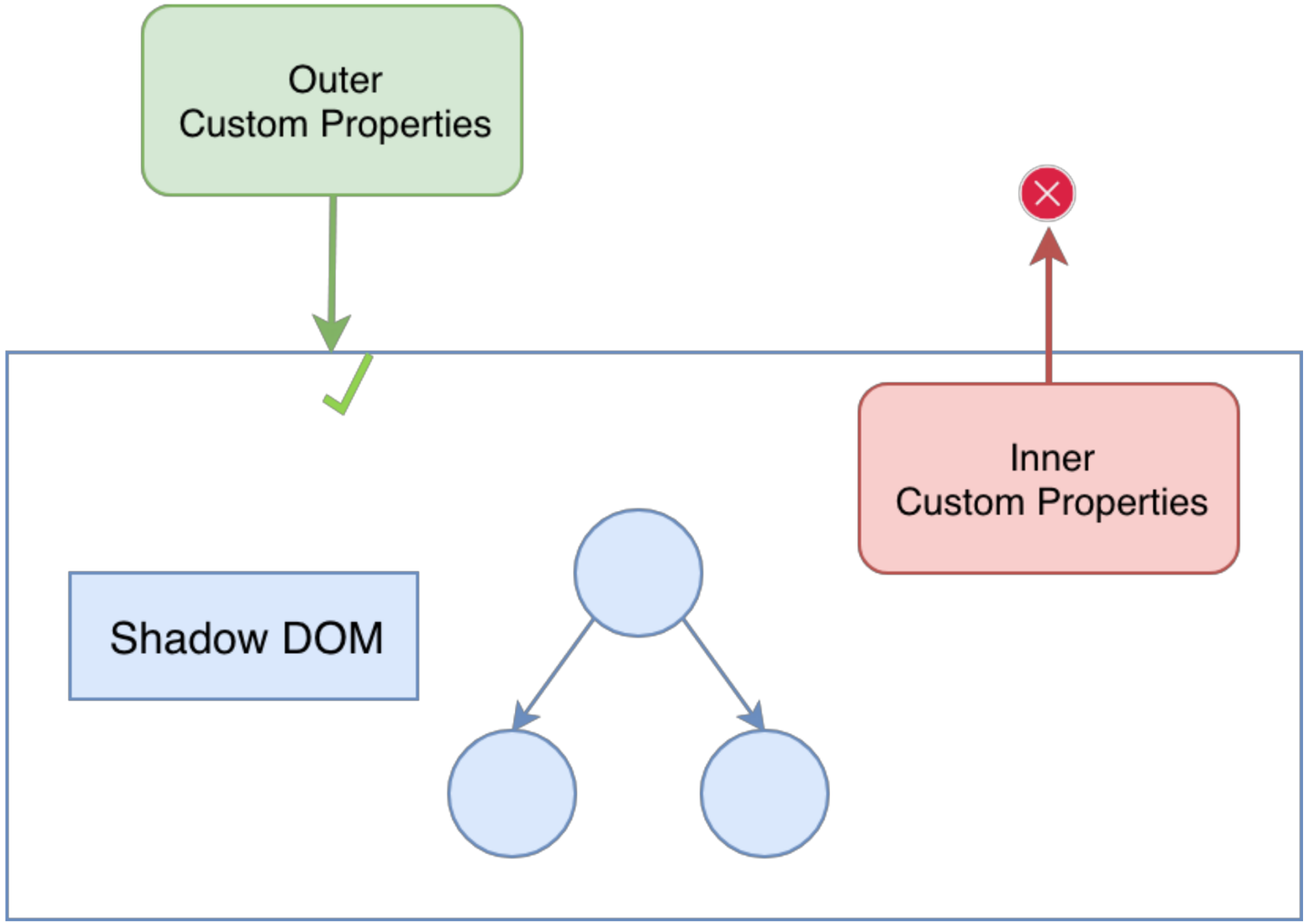
New selectors

```
:host {
  --navigation-bg: var(--primary);

  all: initial;
}

:host([disabled]) {
  pointer-events: none;
  opacity: 0.4;
}

:host-context(.dark-theme) {
  background-color: black;
}
```

CSS Containment

```
:host {
  contain: none | strict | content | [ size || layout || style || paint ];
}

:host {
  contain: content;
}
```

With Custom Properties?



vjeux CSS-in-JS

1. Global namespace
2. Dependencies
3. Dead code
4. Minification
5. Sharing constants
6. Non-deterministic resolution
7. Breaking isolation

vjeux CSS-in-JS

1. Global namespace => **Shadow DOM**
2. Dependencies => **Shadow DOM**
3. Dead code => **Shadow DOM**
4. Minification => **Shadow DOM (?)**
5. Sharing constants => **Custom Properties**
6. Non-deterministic resolution => **Shadow DOM (?)**
7. Breaking isolation => **Shadow DOM**

Dependencies

With postcss-import

```
@import "normalize";
@import "local/button.css";

.c-promo-button {
  background: rebeccapurple;
}
```

Dead-code elimination

```
.c-promo-button {
  background: rebeccapurple;
}
```

Minification

CSS-Blocks => OptiCSS

```
.c-promo-button {
  background: rebeccapurple;
}

/* into */

.f {
  background: rebeccapurple;
}
```

Static analysis

No runtime cost for
- dead-code elimination
- minification

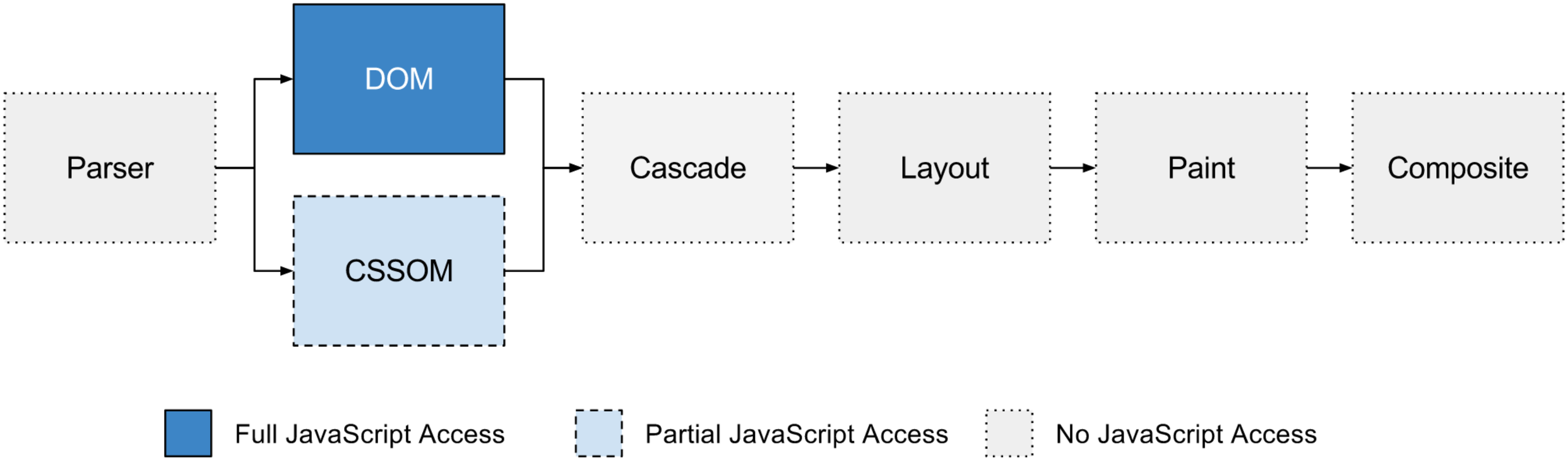
Sharing constants

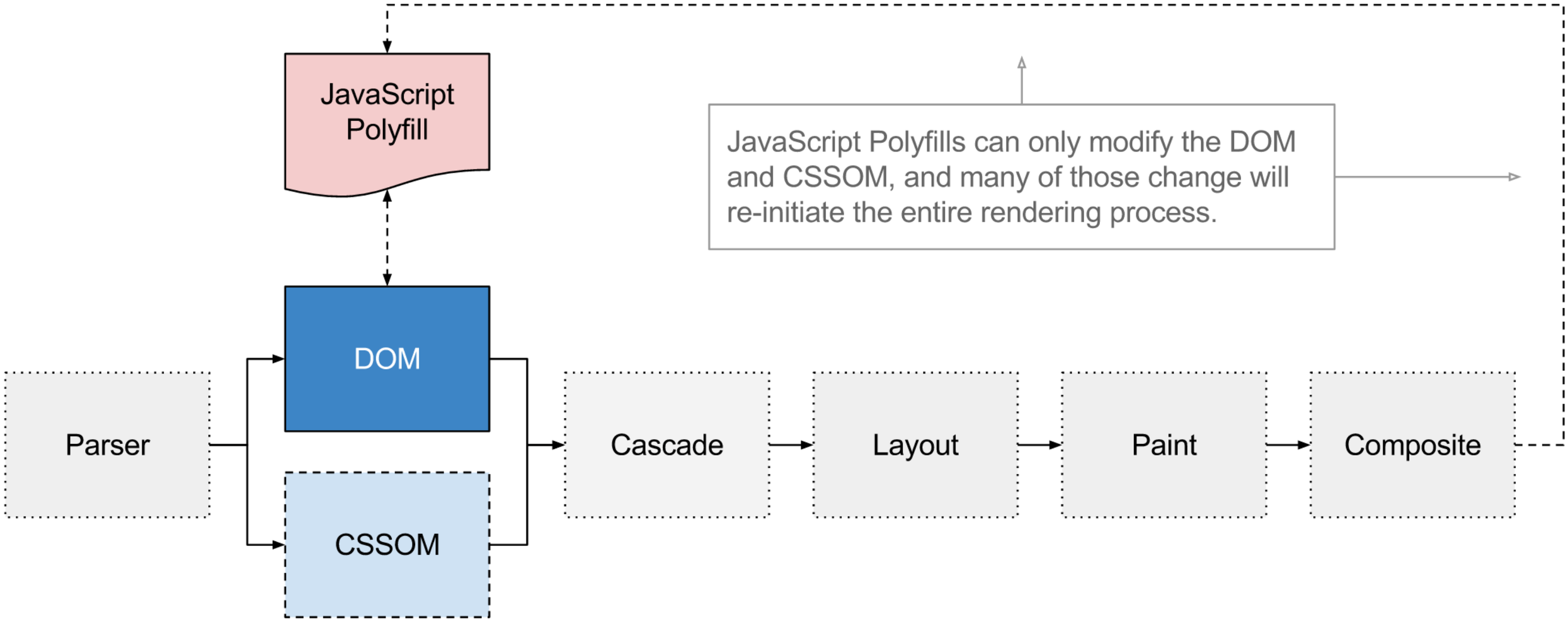
```
.c-button {
  padding: var(--button-padding);
}

render() {
  const buttonPadding = 10
  return (
    <button
      className="c-button"
      style={{ '--button-padding': buttonPadding }}
    ></button>
  )
}
```

CSS Houdini

CSS Houdini

— API to extend CSS itself
— Hook into CSS rendering engine
— In JS things move too fast, in CSS too slow


DOM
Parser
CSSOM
Cascade
Layout
Paint
Composite
Full JavaScript Access
Partial JavaScript Access
No JavaScript Access


JavaScript Polyfill
JavaScript Polyfills can only modify the DOM and CSSOM, and many of those change will re-initiate the entire rendering process.
DOM
Parser
CSSOM
Cascade
Layout
Paint
Composite

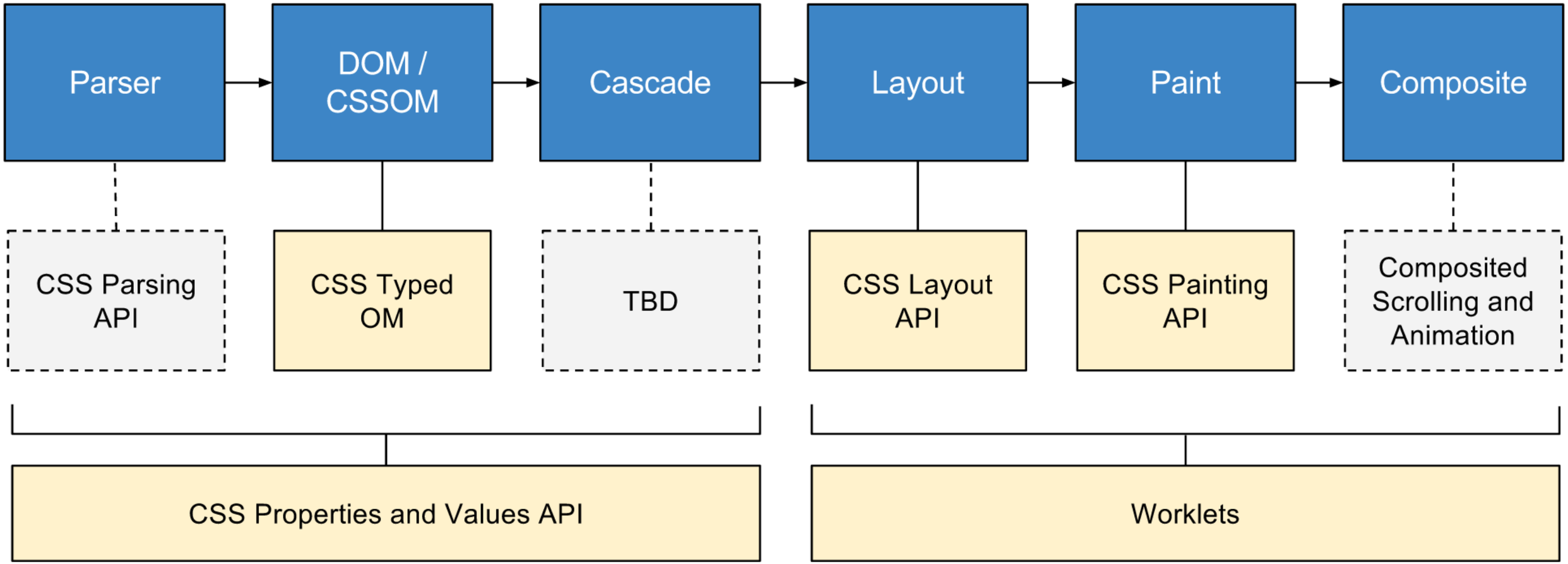
Houdini

— Performant CSS polyfills
— Normalized CSS

CSS Polyfills [6]

— Polyfilling CSS is incredibly hard
— CSSOM discards any CSS rule it doesn’t understand

[6] smashingmagazine


Parser
DOM / CSSOM
Cascade
Layout
Paint
Composite
CSS Parsing API
CSS Typed OM
TBD
CSS Layout API
CSS Painting API
Composited Scrolling and Animation
CSS Properties and Values API
Worklets

CSS Properties and Values API

```
CSS.registerProperty({
    name: "--stop-color",
    syntax: "<color>",
    inherits: false,
    initialValue: "rgba(0,0,0,0)"
});
```

```
.button {
  --stop-color: red;
  background: linear-gradient(var(--stop-color), black);
  transition: --stop-color 1s;
}
```

```
.button:hover {
  --stop-color: green;
}
```

CSS Object Model (CSSOM)

— Like DOM, but for CSS

```
<style>
  body { background-color: red; }
</style>

const stylesheet = document.styleSheets[0];
stylesheet.cssRules[0].style.backgroundColor = "blue";
```

CSS Object Model (CSSOM)

```
const getVariable = (el, propertyName) => {
  const styles = window.getComputedStyle(el);

  return String(styles.getPropertyValue(propertyName)).trim();
};

const setDocumentVariable = (propertyName, value) => {
  document.documentElement.style.setProperty(propertyName, value);
};
```

CSS Object Model (CSSOM)

```
const el = $('#someDiv').style.height;
el.style.height += 10;

el.style.height = `${Number(el.style.height) + 10}px`;
```

CSS Object Model (CSSOM)

```
const el = $('#someDiv').style.height += 10;
el.style.height += 10;

el.style.height = `${Number(el.style.height) + 10}px`;
```

CSS Typed OM

```
el.attributeStyleMap.set('height', CSS.px(42));
const height = el.attributeStyleMap.get('height');
console.log(height.value, height.unit); // 42, 'px'

el.attributeStyleMap.has('opacity') // false
el.attributeStyleMap.delete('height')
el.attributeStyleMap.clear();
```

Why CSS Typed OM

— Fewer bugs
— Math operations and conversions
— Better performance (`requestAnimationFrame`)

CSS Layout API

— Custom layout logic

```
body {
  display: layout('masonry');
}
```

CSS Layout API

```
registerLayout('masonry', class {
  static get inputProperties() {
    return ['width', 'height']
  }

  static get childrenInputProperties() {
    return ['x', 'y', 'position']
  }

  *layout(children, edges, constraints, styleMap) {
    // Layout logic goes here.
  }
}
```

Worklets

— Like Workers but for rendering
— Indipendent from main thread
— No access to DOM, Network etc.
— Lifetime is not defined

```
if ('layoutWorklet' in CSS) {
  CSS.layoutWorklet.addModule('masonry.js');
}
```

CSS Paint API

— Custom behaviour anywhere a CSS image is expected

— `background-image`, `border-image`, `linear-gradient`

— anything that can accept `url()`

```
.bubble {
  --circle-color: blue;
  background-image: url("circle.png");
  background-image: paint('circle');
}
```

CSS Paint API [^7]

```
registerPaint('circle', class {
  static get inputProperties() { return ['--circle-color']; }

  paint(ctx, size, properties) {
    const color = properties.get('--circle-color');
    ctx.fillStyle = color;

    const x = size.width / 2;
    const y = size.height / 2;
    const radius = Math.min(x, y);

    ctx.beginPath();
    ctx.arc(x, y, radius, 0, 2 * Math.PI, false);
    ctx.fill();
  }
});
```

PaintRenderingContext2D

— A subset of the CanvasRenderingContext2D API

— No `CanvasImageData`, `CanvasText`

— spec

Use cases

— Lighter and more performant implementation (ripple)
— Dynamic background
— Polyfill for CSS features like *conic gradients*

Demos

— Circle in textarea
— Ripple
— QRCode
— css-houdini.rocks

Paint API vs pure Canvas

- OffscreenCanvas
- Paint API is
  - reactive
  - lazy
  - auto-sized

— Is Houdini ready yet?

— houdini.glitch.me

Always bet on standards

One last slide...

Jiayi Hu

[dʒʌɪ]

Front-end developer

— jiayi.ghu@gmail.com
— Twitter: @jiayi_ghu
— GitHub: jiayihu/talks
— italiajs.slack.com


CSS