
/**
* Main Speekr Admin styles.
*
* Includes: Metaboxes and Option Pages
*/
:root{
--primary: #9768A7; /* 30aabc */
--primarylight: #D0BAD7;
--primarylighter: #EADAF0;
--primarydark: #5C4B62;
--lightgray: #E6E6E6;
--gray: #A59EA7;
--success: #84A768;
--successdark: #5B7F3D;
--successbg: #F0F3ED;
--error: #D56060;
--errordark: #A63333;
--errorbg: #F8ECEC;
}
.speekr-settings ::-moz-selection,
[value="talks"] ~ #poststuff ::-moz-selection {
background: var(--primary);
color: #FFF;
CSS Custom Properties
Add Variables to Your CSS

Geoffrey Crofte

UX/UI Web Designer at Maltem (Foyer)

@geoffrey_crofte

geoffrey.crofte.fr

creativejuiz.fr

Overview

- What are Custom Properties?
- Quick start with CSS Variables.
- Why CSS C.P. instead of Sass variables?
- Use cases & Demos
- Takeaways

What are Custom Properties?

What are ~~Custom Properties~~?

CSS variables

This not about...

```
$color: #bada55;

@color: #bada55;
```

This is more about...

```
div {
  color: #8f00cb;
  border: 6px double currentColor;
}
div::before {
  border-top: 1px solid currentColor;
  border-bottom: 1px solid currentColor;
}
```

CSS Rocks

http://bit.ly/csscurrentcolor

This is more about...

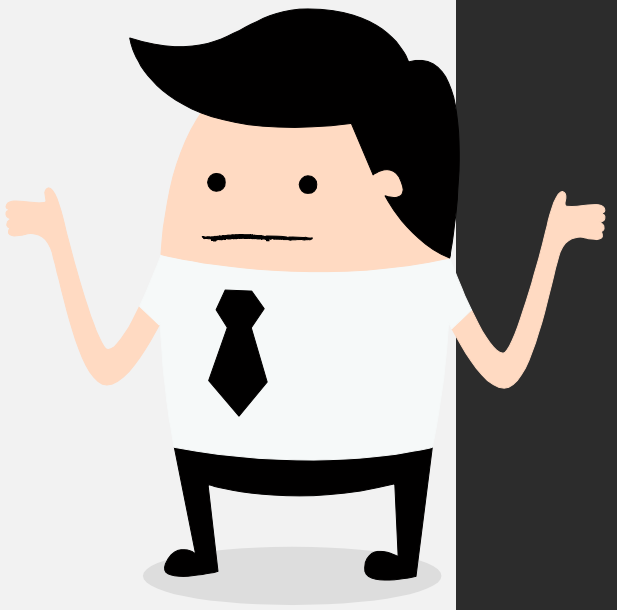
```
div {
  color: #e89200;
  border: 6px double currentColor;
}
div::before {
  border-top: 1px solid currentColor;
  border-bottom: 1px solid currentColor;
}
```

CSS Rocks

http://bit.ly/csscurrentcolor

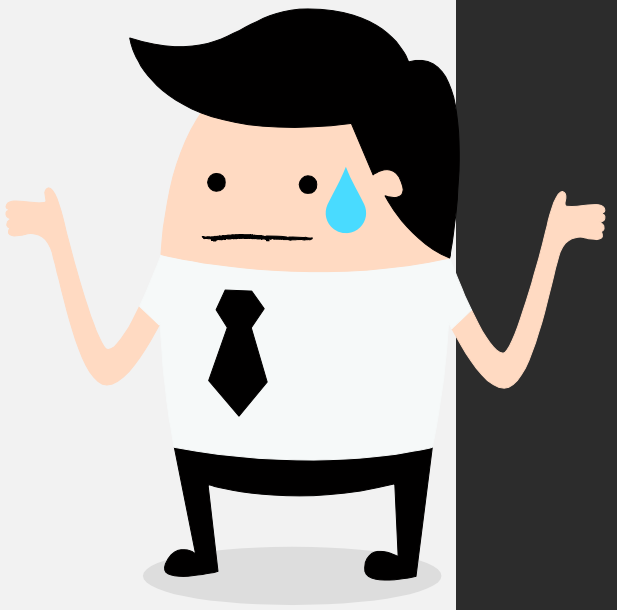
But this is not about...

`currentColor`



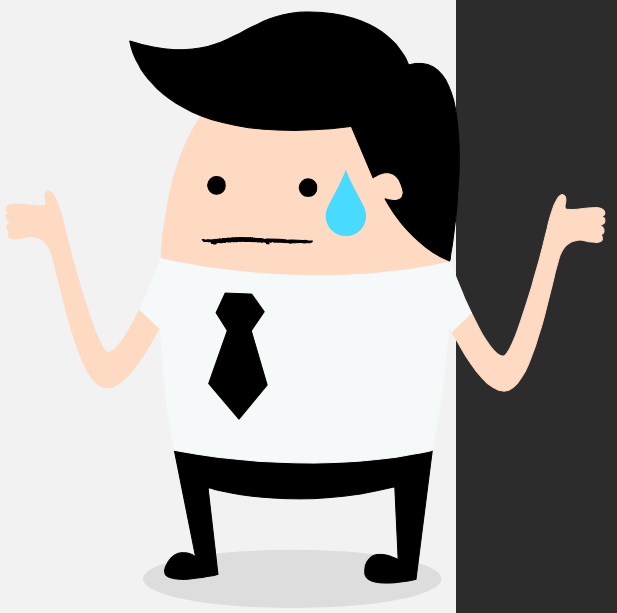
> A *custom property* is any property whose name starts with two dashes [...] like `--foo`

"CSS Custom Properties" — @geoffrey_crofte



> Custom properties are solely for use by authors and users; CSS will never give them a meaning beyond what is presented here.

“CSS Custom Properties” — @geoffrey_crofte



> Custom properties are solely for use by authors and users; CSS will never give them a meaning beyond what is presented here.

“CSS Custom Properties” — @geoffrey_crofte

EXAMPLE 1

Custom properties define variables, referenced with the 'var()' notation, which can be used for many purposes. For example, a page that consistently uses a small set of colors in its design can store the colors in custom properties and use them with variables:

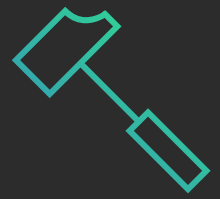
What the Spec says...

- Custom properties define variables
- Variables are referenced with the **var()** notation
- They are Case-Sensitive
- Their value is “extremely permissive”

In other words...

Do the fu%€

you want with them.



How to use CSS Variables?

CSS Declaration

```
:root {
    --variableName: value;
}
```

CSS Declaration

```
.element {
    --variableName: value;
}
```

CSS Use

```
.element {
    property: var(--variableName);
}
```

Quick Example of use

```
:root {
  --link: purple;
}
a {
  color: var(--link);
}
```

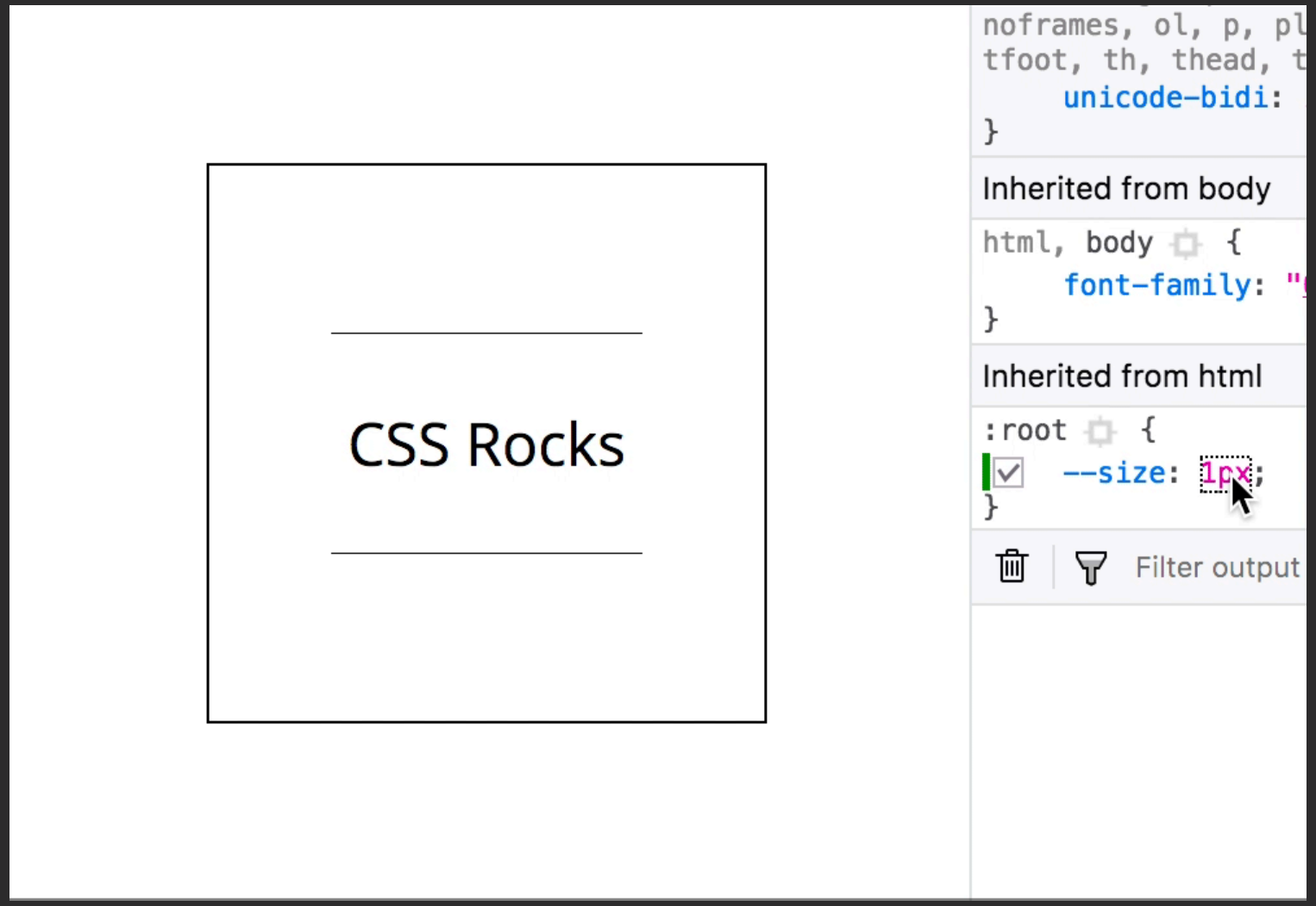
Why CSS instead of Sass Variables?

Computation

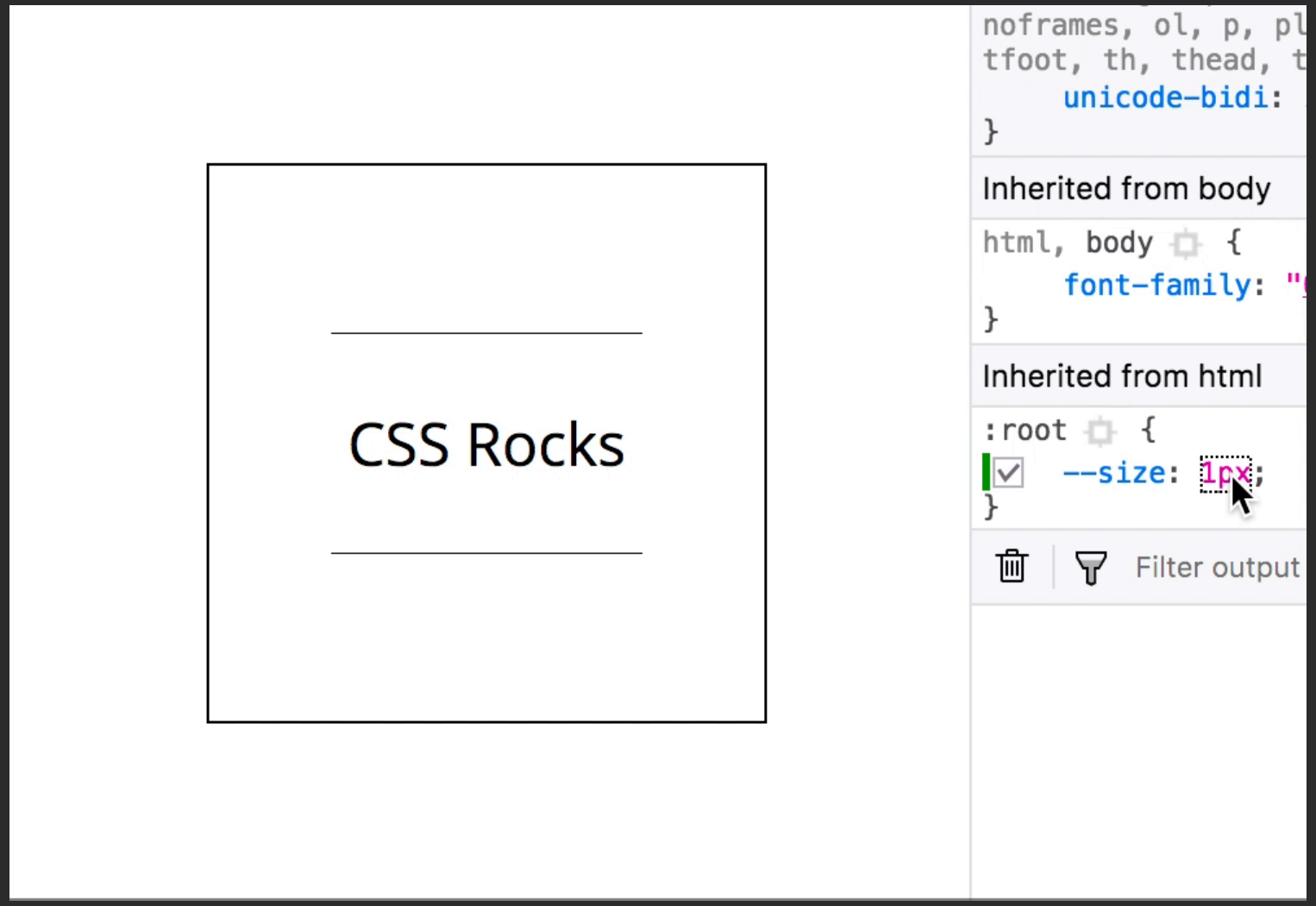
The main issue with Sass (or LESS) variables:

They have to be computed to get their value.

CSS Variables are alive



CSS Variables are alive



JavaScript can access them

```
e.target.style.setProperty('--x', x + 'px');
e.target.style.setProperty('--y', y + 'px');
```

Media Queries ❤️ CSS Variables

```scss
$global-spacing: 15px;

.mod {
    margin: $global-spacing 0;
    padding: $global-spacing;
}
@media (min-width: 640px) {

    $global-spacing: 40px;

    .mod {
        margin: $global-spacing 0;
        padding: $global-spacing;
    }
}
```

Media Queries ❤️ CSS Variables

```scss
$global-spacing: 15px;

.mod {
    margin: $global-spacing 0;
    padding: $global-spacing;
}
@media (min-width: 640px) {

    $global-spacing: 40px;

    .mod {
        margin: $global-spacing 0;
        padding: $global-spacing;
    }
}
```

```css
:root {
    --global-spacing: 15px;
}
.mod {
    margin: var(--global-spacing) 0;
    padding: var(--global-spacing);
}
@media (min-width: 640px) {
    :root {
        --global-spacing: 40px;
    }
}
```

CSS Variables are inherited

```
:root {
  --link: purple;
}
```

Simple Link

Nav Link Second Nav Link

CSS Variables are inherited

```
:root {
  --link: purple;
}

a {
  color: var(--link);
}
```

Simple Link

Nav Link Second Nav Link

CSS Variables are inherited

```
:root {
  --link: purple;
}

a {
  color: var(--link);
}
```

Simple Link

Nav Link Second Nav Link

CSS Variables are inherited

```
:root {
  --link: purple;
}

a {
  color: var(--link);
}

nav a {
  --link: salmon;
}
```

Simple Link

Nav Link Second Nav Link

CSS Variables are inherited

```
:root {
  --link: purple;
}

a {
  color: var(--link);
}

nav a {
  --link: salmon;
}
```

Simple Link

Nav Link Second Nav Link

CSS Variables are inherited

```
:root {
  --link: purple;
}

a {
  color: var(--link);
}

nav {
  --link: salmon;
}
```

Simple Link

Nav Link Second Nav Link

Demos & Use Cases

My last works

They trusted me

Another link for fun

http://bit.ly/cssvargradient

My last works

They trusted me

Another link for fun

http://bit.ly/cssvargradient

Mouse position

```
document.querySelector('.link').addEventListener('mousemove', (e)=>{

  const x = e.pageX - e.target.offsetLeft;
  const y = e.pageY - e.target.offsetTop;

  e.target.style.setProperty( '--x', x + 'px' );
  e.target.style.setProperty( '--y', y + 'px' );
});
```

CSS Transformation

```
.link::before {
    --original-size: 400px;
    --size: 0;

    content: '';
    position: absolute;
    left: calc( -1 * var(--original-size) / 2 );
    top: calc( -1 * var(--original-size) / 2 );
    width: var(--size);
    height: var(--size);
    background: radial-gradient(circle closest-side, rgba(255,255,255,.2), transp
    transform: translate( var(--x), var(--y) );
    transition: width .375s ease, height .375s ease, top .375s ease, left .375s e
}
```

CSS Transformation

```css
.link::before {
    --original-size: 400px;
    --size: 0;

    content: '';
    position: absolute;
    left: calc( -1 * var(--original-size) / 2 );
    top: calc( -1 * var(--original-size) / 2 );
    width: var(--size);
    height: var(--size);
    background: radial-gradient(circle closest-side, rgba(255,255,255,.2), transp
    transform: translate( var(--x), var(--y) );
    transition: width .375s ease, height .375s ease, top .375s ease, left .375s e
}
```

CSS Transformation

```css
.link::before {
    --original-size: 400px;
    --size: 0;

    content: '';
    position: absolute;
    left: calc( -1 * var(--original-size) / 2 );
    top: calc( -1 * var(--original-size) / 2 );
    width: var(--size);
    height: var(--size);
    background: radial-gradient(circle closest-side, rgba(255,255,255,.2), transp
    transform: translate( var(--x), var(--y) );
    transition: width .375s ease, height .375s ease, top .375s ease, left .375s e
}
```

CSS Transformation

```
.link::before {
    --original-size: 400px;
    --size: 0;

    content: '';
    position: absolute;
    left: calc( -1 * var(--original-size) / 2 );
    top: calc( -1 * var(--original-size) / 2 );
    width: var(--size);
    height: var(--size);
    background: radial-gradient(circle closest-side, rgba(255,255,255,.2), transp
    transform: translate( var(--x), var(--y) );
    transition: width .375s ease, height .375s ease, top .375s ease, left .375s e
}
```

CSS Transformation

```
.link:hover::before,
.link:focus::before {
  width: 400px;
  height: 400px;
}
```

CSS Transformation

```
.link:hover::before,
.link:focus::before {
  width: 400px;
  height: 400px;
}
```

```
.link:hover::before,
.link:focus::before {
    --size: var(--original-size);
}
```

CSS Transformation

```
.link:hover::before,
.link:focus::before {
  width: 400px;
  height: 400px;
}
```

```
.link:hover::before,
.link:focus::before {
    --size: var(--original-size);
}

width: var(--size);
height: var(--size);
```

Now you’re wondering...

Why not pushing transformations directly in JavaScript?

Now you’re wondering...

Why not pushing transformations directly in JavaScript?

```
el.style.setProperty('transform:',
                     'translate(' + x + 'px,' + y + 'px)');
```

Now you're wondering...

Why not pushing transformations directly in JavaScript?

```
el.style.setProperty('transform:',
                     'translate(' + x + 'px,' + y + 'px)');
```

- **Maintainability / Portability**
  style="transform..." is dirty. Period.

Now you're wondering...

Why not pushing transformations directly in JavaScript?

```
el.style.setProperty('transform:',
                     'translate(' + x + 'px,' + y + 'px)');
```

- **Maintainability / Portability**
  style="transform..." is dirty. Period.

- **Inheritance**
  style="transform..." do not make --x and --y inherited.

Now you're wondering...

Why not pushing transformations directly in JavaScript?

```
el.style.setProperty('transform:',
                     'translate(' + x + 'px,' + y + 'px)');
```

- **Maintainability / Portability**
  style="transform..." is dirty. Period.
- **Inheritance**
  style="transform..." do not make --x and --y inherited.
- **It's a philosophy**
  CSS is for styling. JS is not.

Now you're wondering...

Why not pushing transformations directly in JavaScript?

```
el.style.setProperty('transform:',
                     'translate(' + x + 'px,' + y + 'px)');
```

- **Maintainability / Portability**
  style="transform..." is dirty. Period.
- **Inheritance**
  style="transform..." do not make --x and --y inherited.
- **It's a philosophy**
  CSS is for styling. JS is not.
- **Futur proof**
  Let the CSS engine handle that part.

Be less curious about people and more curious about ideas.

http://bit.ly/cssvargradient2

Be less curious about people and more curious about ideas.

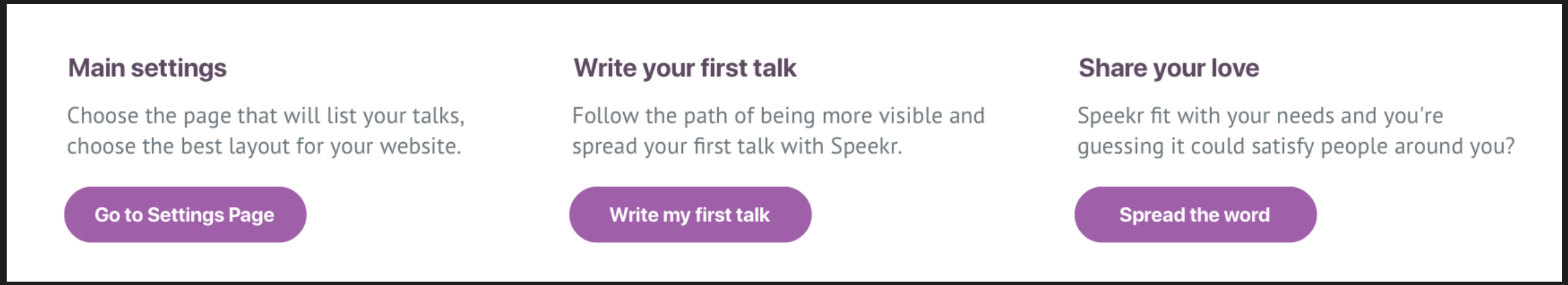
http://bit.ly/cssvargradient2

CSS Transformation

```
  background-image: linear-gradient(45deg,#fff070 0%,#fff070
calc(var(--x) * .1 * 1%),#00c9d3 calc(var(--x) * .4 * 1%),#00c9d3
calc(var(--x) * .7 * 1%) );
  -webkit-text-fill-color: transparent;
  -webkit-background-clip: text;
```

Themable interfaces

Themable interfaces



Themable interfaces

Main settings

Choose the page that will list your talks, choose the best layout for your website.

Go to Settings Page

Write your first talk

Follow the path of being more visible and spread your first talk with Speekr.

Write my first talk

Share your love

Speekr fit with your needs and you're guessing it could satisfy people around you?

Spread the word

Main settings

Choose the page that will list your talks, choose the best layout for your website.

Go to Settings Page

Write your first talk

Follow the path of being more visible and spread your first talk with Speekr.

Write my first talk

Share your love

Speekr fit with your needs and you're guessing it could satisfy people around you?

Spread the word

Themable interfaces

```
h2, .h2-like {
    color: purple;
}
p, li {
    color: #444;
}
button, .button {
    background: purple;
    color: #FFF;
}
/* Theme */
.container-dark h2,
.container-dark .h2-like {
    color: #F6ECFA;
}
.container-dark p,
.container-dark li {
    color: #F3F3F3;
}
.container-dark button,
.container-dark .button {
    background: #F6ECFA;
    color: #666;
}
```

Themable interfaces

```css
h2, .h2-like {
    color: purple;
}
p, li {
    color: #444;
}
button, .button {
    background: purple;
    color: #FFF;
}
/* Theme */
.container-dark h2,
.container-dark .h2-like {
    color: #F6ECFA;
}
.container-dark p,
.container-dark li {
    color: #F3F3F3;
}
.container-dark button,
.container-dark .button {
    background: #F6ECFA;
    color: #666;
}
```

```css
:root {
    --text: #444;
    --primary: purple;
    --btn-color: #FFF;
}
h2, .h2-like {
    color: var(--primary);
}
p, li {
    color: var(--text);
}
button, .button {
    background: var(--primary);
    color: var(--btn-color);
}

.container-dark {
    --text: #F3F3F3;
    --primary: #F6ECFA;
    --btn-color: #666;
}
```

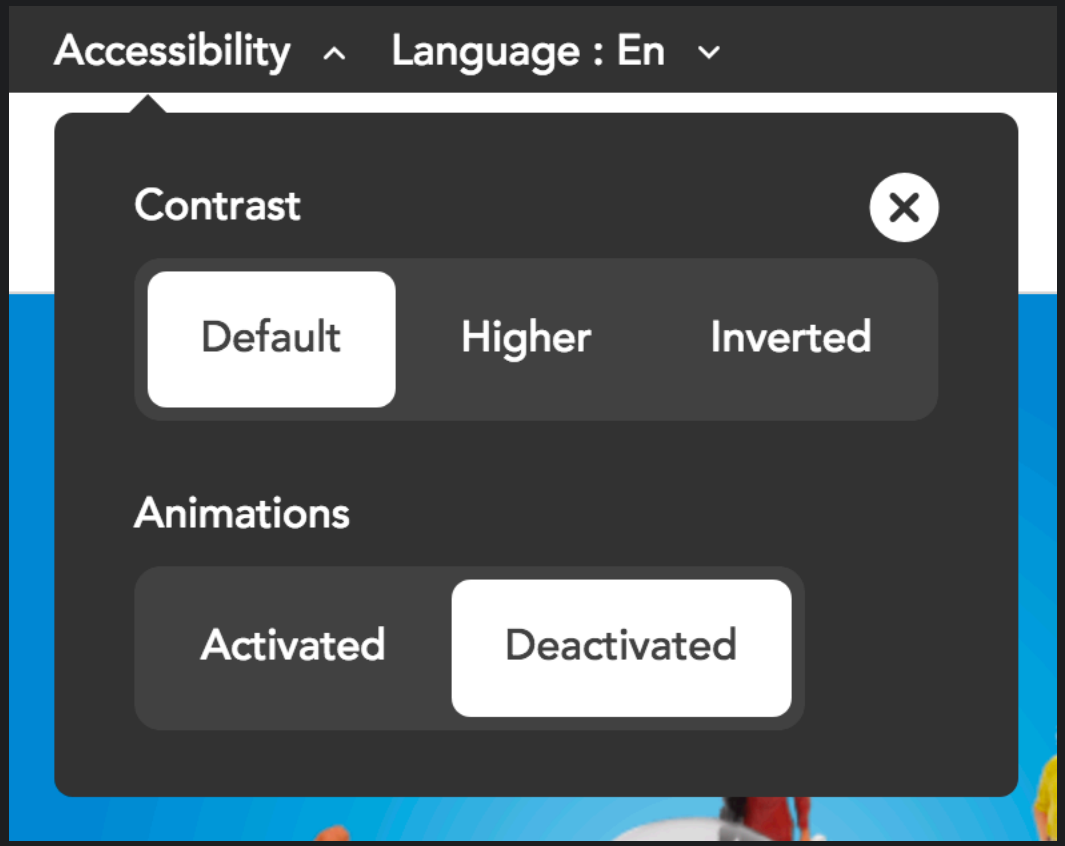
Themable interfaces

```css
h2, .h2-like {
    color: purple;
}
p, li {
    color: #444;
}
button, .button {
    background: purple;
    color: #FFF;
}
/* Theme */
.container-dark h2,
.container-dark .h2-like {
    color: #F6ECFA;
}
.container-dark p,
.container-dark li {
    color: #F3F3F3;
}
.container-dark button,
.container-dark .button {
    background: #F6ECFA;
    color: #666;
}
```

```css
:root {
    --text: #444;
    --primary: purple;
    --btn-color: #FFF;
}
h2, .h2-like {
    color: var(--primary);
}
p, li {
    color: var(--text);
}
button, .button {
    background: var(--primary);
    color: var(--btn-color);
}

.container-dark {
    --text: #F3F3F3;
    --primary: #F6ECFA;
    --btn-color: #666;
}
```

Accessibility Example



Accessibility Example

```
html.a11y {
    --text-size: 1.1em;
    --text: #444;
    --background: #F2F2F2;
    --btn-bg: #552F62;
    --section-dark: #444;
    --section-dark-text: #F2F2F2;
    /* … */
}
```

Responsive

```
:root {
    --global-spacing: 15px;
}
.mod {
    margin: var(--global-spacing) 0;
    padding: var(--global-spacing);
}
@media (min-width: 640px) {
    :root {
        --global-spacing: 40px;
    }
}
```

Responsive

```
:root {
    --global-spacing: 15px;
}
.mod {
    margin: var(--global-spacing) 0;
    padding: var(--global-spacing);
}
@media (min-width: 640px) {
    :root {
        --global-spacing: 40px;
    }
}
```

Further Takeaways

CSS Variables are CSS properties

```
:root {
    --variableName: value !important;
}
```

(kind of) Silent error

```
:root {
    --variableName: lolilol;
}
```

You can’t “build up” values.

```
:root {
    --spacing: 20;
}
:root {
    margin: var(--spacing)px;
}
```

You can’t “build up” values.

```
:root {
    --spacing: 20;
}

:root {
    margin: var(--spacing)px;
}
```

Doesn’t work

Strange behaviour

```
:root {
    --spacing:;
}
```

Invalid

```
:root {
    --spacing: ;
}
```

Valid

Yeah, that's a space caracter.

Fallback value

```
var(--variableName, default);
```

Fallback value

```
button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Fallback value

```
button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback value

```
button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback is there

Fallback value

```
:root {
    --bgColor: #30E28E;
}

button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback is there

Fallback value

```
:root {
    --bgColor: #30E28E;
}

button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback is there

--bgColor Value

Fallback value

```
:root {
    --bgColor: 5deg;
}

button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback is there

--bgColor Value

Fallback value

```
:root {
    --bgColor: 5deg;
}

button {
    background: purple;
    background: var(--bgColor, salmon);
}
```

Doesn’t support var()

Fallback is there

--bgColor Value

Transparent

Variable as Fallback

```
var(--var1, var(--var2, default));
```

Variable as Fallback

```
var(--var1, var(--var2, var(--var3,
var(--var4, var(--var5,
default)))));
```

You can’t cycle with variables

```
.element {
    --size: 200px;
    height: var(--size);
}
.element.modifier {
    --size: calc(var(--size) + 50px);
}
```

You can’t cycle with variables

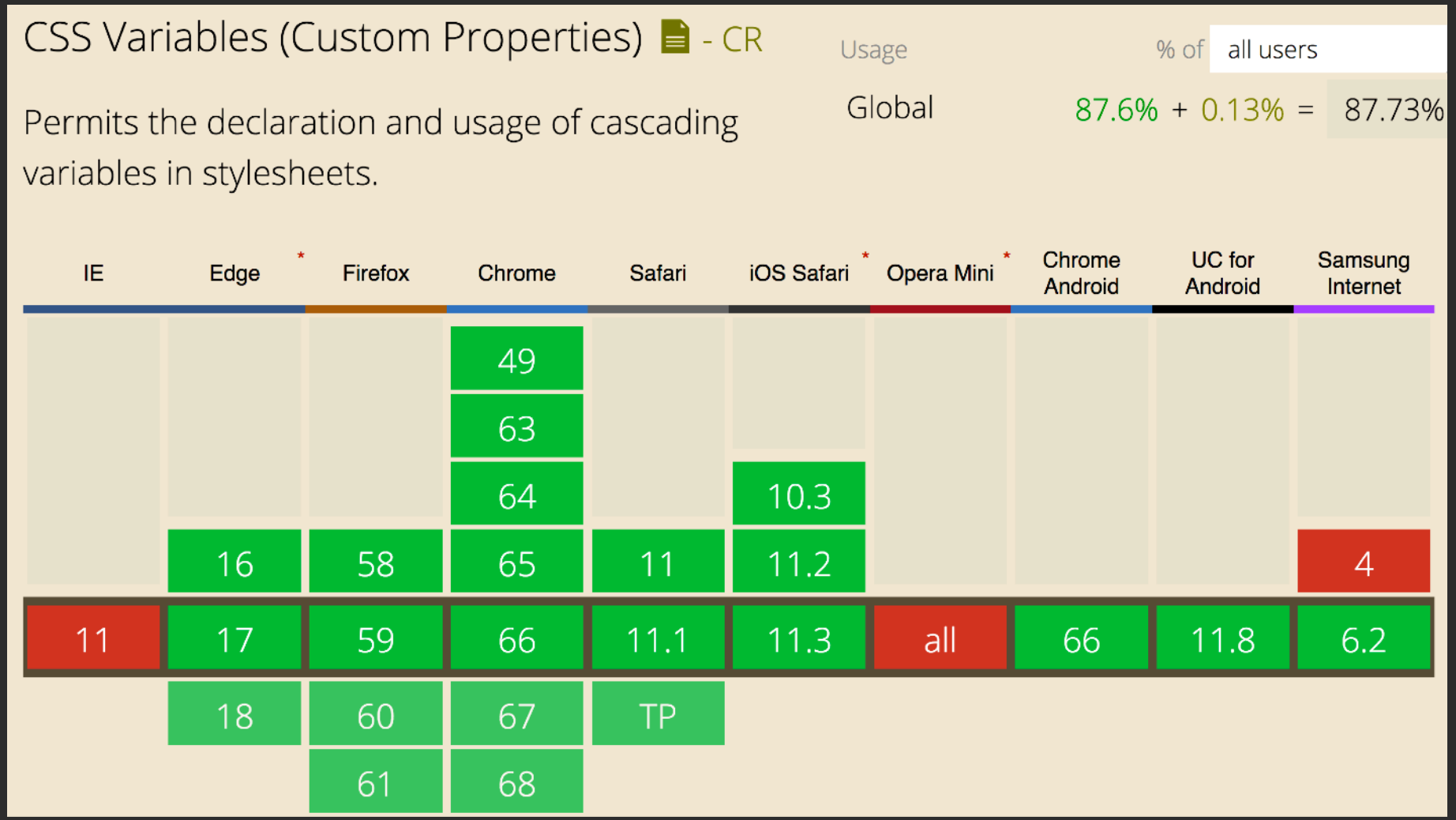
```
.element {
    --size: 200px;
    height: var(--size);
}
.element.modifier {
    --size: calc(var(--size) + 50px);
}
```

```
.element {
    --size: 200px;
    height: var(--size);
}
.element.modifier {
    height: calc(var(--size) + 50px);
}
```

Support Testing

```
@supports (color: var(--)) {
    ...
}
```

Compatibility



Polyfills

https://github.com/**aaronbarker/css-variables-polyfill**

Some resources

- **CSS Variable Secrets** (Lea Verou Smashing Conf. 2017)
- **CSS Custom Properties** (CCSWG W3C documentation)
- **CSS Mouse Tracking Gradient** (Gradient on Text demonstration)
- **CSS Variable Tutorials** (A series of 4 short videos about CSS Variables)

Thanks!

Geoffrey Crofte

UX/UI Web Designer at Maltem (Foyer)

@geoffrey_crofte

geoffrey.crofte.fr

creativejuiz.fr