Developing Inspired Guides with CSS
Custom Properties

Andy Clarke
@malarkey

SydCSS, October 2017
Style guide types

- Static style/visual identity guides
- Voice and tone guides
- Front-end code guidelines
- Component/pattern libraries
Style guide types

- Static style/visual identity guides
- Voice and tone guides
- Front-end code guidelines
- Component/pattern libraries
This text is quoted. A block of quoted text like this is particularly useful when presented as a pull-quote within an article of text.

This is an error feedback message.

This is a feedback message for the user.

Button element  Submit button

Label text
25% off : sydcss25

Valid for an arbitrary 7 days to encourage you buy sooner.
Only 50 codes available. May contain nuts.
Typography

Playfair Display is our primary typeface

Playfair Display is well suited for titling and headlines. It has an extra-large x-height and small descenders. It can be set with no leading if space is tight, for instance in news headlines, or for stylistic effect in titles. Capitals are extra short, and only very slightly lower than the lowercase characters.

Gutenberg grew up knowing the trade of goldsmithing.

The Gutenberg Bible was the first major book to be produced using moveable type. Its impact on literacy and the spread of knowledge was immense. It marked the beginning of modern printing and set the stage for the Reformation and the spread of knowledge throughout Europe.

Gutenberg died in 1468 and was buried in the Franciscan church of Mainz, his contributions largely unknown.

This church and the cemetery were later destroyed, and Gutenberg’s grave is now lost.

This church and the cemetery were later destroyed, and Gutenberg’s grave is now lost.
Typography

Raleway is our primary typeface

Supporting

Base

A versatile, readable typeface designed for large and medium-sized text. It includes all weights from light to bold.

Text

Political

See our website (http://www.example.com) for more information about our typefaces.

Buttons

We use button shapes to distinguish a variety of calls to actions and we've designed these shapes to be easily recognisable and visually appealing.

Button types

Where possible use only one primary action button per content region on one page. Don’t choose a different colour button without considering the reason why that button has been included.

Button sizes

Use standard size button shapes for the majority of calls to action to reserve smaller and larger size for specific purposes. Use large buttons for a single important action and small buttons in confined spaces, for example inside tables. Use the standard button sizes on mobile devices.
**Typography**

**Primary typeface**

**Merriweather**

An elegant sans-serif typeface family intended for headlines and other large size usage, initially designed by Matt Newman as a single thin weight, it was

```
Aa
```

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

**Supporting typeface**

**Lato**

Lato is a sans-serif typeface family started in the summer of 2011 by Warsaw-based designer Łukasz Dąbrowski ("Łat" means "Summer" in Polish). In December 2012 the Lato family was published under the Open Font License by his foundry tp/tpolish, with support from Google.

```
A
```
Noto Sans

Primary typeface

Noto Sans

Noto fonts are intended to be visually harmonious across multiple languages, with compatible heights and stroke thicknesses.

Supporting typeface

Elsie

Blues is inspired by mid-winter energy. This new typeface was created to celebrate the world of women, glamour and fashion. It combines the strength of Bedoni with the softness of italics. Sensitive, attractive, full of personality, innovative and gentle with both classic and new design features.

Branded favicon

Elements of our logo are recognizable even in this, our smallest branded element.
Open Sans is a humanist sans serif typeface, designed with an upright design, round shapes, and a neutral, yet friendly appearance. It was optimized for print, web, and mobile environments, and has excellent legibility characteristics in its letterforms.

Primary
Indicates a strongly interactive element, such as a call to action or a button. It is also used to denote major categories or sections.

Supporting
Indicates an interactive element in focus, for example when someone presses a button or launches an app, or when the user is interacting with a form.

Accent
Used to add emphasis to a design, for example used for highlights, callouts, and for interactive states.

Base
A neutral color used for text, labels, and non-interactive states.
Preprocessor variables

- Are removed when compiled
- Cannot be updated at runtime
- Cannot be manipulated by JavaScript
Native CSS variables

• No need for a pre/post processor
• Use the cascade
• Media query or other state changes
• Can be manipulated by JavaScript
LESS syntax

Variable
@color-text-default : black;

Style
body {
  color : @color-text-default; }

Sass syntax

Variable
$color-text-default: black;

Style
body {
  color: $color-text-default;
}
Custom properties

```css
--color-text-default: black;
```

or:

```css
--color_text_default: black;
```

Two dashes identified as variables. Hyphens or underscores, but not spaces, allowed in property names.
Case sensitive

- --color-text-default: black;

is different to:

- --Color_Text_Default: dimgray;

Custom property names are case-sensitive. --foo and --FOO are distinct properties.
Multiple value strings

Variable
--text-shadow : 1px 1px 2px black;

Style
body {
  text-shadow : var(--text-shadow);
}

Multiple value strings allowed.
Retrieving variables

Custom property
--color-text-default : black;

Style
body {
  color : var(--color-text-default);
}

var() tells a browser to retrieve the value of a property.
JavaScript

document.documentElement.style.setProperty("--color-text-default", "black");

CSS Custom Properties can be accessed and manipulated by JavaScript. I have no idea how to do that.
Nesting variables

1st custom property
--color-base-default-darker: dimgray;

2nd custom property
--gradient-gray: linear-gradient(to top,
  var(--color-base-default-darker), black);
Specificity

:root {
  --color-text-default: black;
}

body {
  --color-text-default: dimgray;
}

h1 {
  colour: var(--color-text-default);
}
## Colour palette

<table>
<thead>
<tr>
<th></th>
<th>Primary</th>
<th>Supporting</th>
<th>Accent</th>
</tr>
</thead>
<tbody>
<tr>
<td>--color-primary</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-primary-dark</td>
<td></td>
<td></td>
<td>Tint</td>
</tr>
<tr>
<td>--color-primary-darker</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-primary-light</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-primary-lighter</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-supporting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--color-supporting-dark</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-supporting-darker</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-supporting-light</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-supporting-lighter</td>
<td></td>
<td>Tint</td>
<td></td>
</tr>
<tr>
<td>--color-accent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--color-accent-dark</td>
<td></td>
<td></td>
<td>Tint</td>
</tr>
<tr>
<td>--color-accent-darker</td>
<td></td>
<td></td>
<td>Tint</td>
</tr>
<tr>
<td>--color-accent-light</td>
<td></td>
<td></td>
<td>Tint</td>
</tr>
</tbody>
</table>
Design tokens

- Colour palette
- Background colours
- Border colours
- Border radii
- Border widths
- Font colours
- Font families
- Font sizes
- Line heights
- Spacing
- Time
Design tokens

--color-primary
--color-primary-dark
--color-primary-darker
--color-primary-light
--color-primary-lighter

--color-supporting
--color-supporting-dark
--color-supporting-darker
--color-supporting-light
--color-supporting-lighter

Semantic, human readable properties.
Scope

:root {
--font-primary-typeface : "Playfair Display";
--font-primary-stack : "Playfair Display", serif;
--font-primary-weight : 400; }

:root pseudo-class matches everything in the <html> element but with a higher specificity.
Scope

Root element
:root {
--color-text-default : black; }

Header
header {
--color-text-default : dimgray; }

Collects properties for easier maintenance.
Scope elements

ARIA roles
[role="main"] {--color-text-default : black; }

Classes
.secondary { --color-text-default : dimgray; }

IDs
#main { --color-text-default : black; }
Scope elements

<p><strong>black</strong>: Inherited from :root<html></p>
<header><p><strong>dimgray</strong>: Set by <header> element</header></p></header>
<div role="main"><strong>black</strong>: Set by main role</div>
<p class="secondary"><strong>dimgray</strong>: Set by .secondary class</p>
<div id="main"><strong>black</strong>: Set by #main id</div>
Scope

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

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

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

Set the appropriate ::after pseudo-element language according to the lang attribute value.
Scope theming

```css
body, playfair {
  --font-typeface-primary: "Playfair Display";
}

body, merriweather {
  --font-typeface-primary: Merriweather;
}

body {
  font-family: var(--font-typeface-primary);
}
```
Calc

1st custom property
--spacing : 1.5rem;

More custom properties
--spacing-small : calc(2/var(--spacing));
--spacing-medium : calc(1.5*var(--spacing));
--spacing-large : calc(2*var(--spacing));
NB: It’s not possible to use CSS Custom Properties in pseudo elements in Edge 15.
Internet Explorer 11

CSS Custom Properties
not implemented
Fails silently

Style
body {
  color: var(--color-text-default);
}

Properties whose names don’t exist are ignored.
Dropdowns

Substitutions are similar to font stacks.

body {
  color: black;
  color: var(--color-text-default); }

Substitutions are similar to font stacks.
Substitutions

body {
  color: var(--color-text-default), black;
}

Substitutions are similar to font stacks.
Supports

Style
@sUPPORTS((--foo : bar)) {

body {
  color : var(--color-text-default);
}
}

Renders CSS Custom Properties only if the dummy property is supported.
Supports (not)

Style
@supports(not (--foo : bar)) {

body {
  color : $color-text-default; }
}

Renders CSS style when the dummy property isn’t supported.
Polyfill

PostCSS has a plugin to transform CSS Custom Properties if you understand how to

$ npm install postcss-custom-properties.

I don’t.

http://postcss.org
https://github.com/postcss/postcss-custom-properties
Playfair Display is our primary typeface

Playfair Display is well suited for titling and headlines. It has an extra-large x-height and short descenders. It can be set with no leading if space is tight, for instance in news headlines, or for stylistic effect in titles. Capitals are extra short, and only very slightly larger than the lowercase characters.

Gutenberg grew up knowing the trade of goldsmithing.

This church and the cemetery were later destroyed, and Gutenberg’s grave is now lost.

Gutenb erg died in 1468 and was buried in the Franciscan church of Mainz, his contributions largely unknown.

The old-fash ioned block of print with quotation marks is set in Playfair Display, the font used for titling and headlines. The text is set in paragraph mode, with justified alignment and a line height of 1.15. The quotation marks are set in the same font and are aligned to the right margin.
HTML
Properties to display

CSS
Properties for rendering
Configure variables in config.json.

```json
"color": {
  "text": {
    "default": "#574e4e",
    "alt": "#a19c9c",
    "inverse": "#fff",
    "inverse-alt": "#a19c9c"
  }
}
```
Mustache

<dl>
  <dt>--color-text-default</dt>
  <dd>{{ color.text.default }}</dd>
</dl>

Render Mustache variables into HTML at runtime.
Render partial HTML pages into style guide index page.
CSS

body {
  color: var(--color-text-default);
}
Dedicated stylesheets for each element type and components.
CSS imports

<link rel="stylesheet" type="text/css" href="css/inspired.css"/>
inspired.css

@import url('_global.css');
@import url('_a.css');
@import url('_alerts.css');
@import url('_breadcrumbs.css');
@import url('_cards.css');
@import url('_components.css');
@import url('_forms.css');
@import url('_form-buttons.css');
...

Style guides, pattern and component library templates that are beautiful as well as functional

Easily create beautiful, functional style guides for yourself, your clients or company. Six distinctive designs to choose from that each contain pages for design principles and tokens, branding colours and logos, typography, icons and media, form elements and buttons, tables, plus interactive and navigation elements.

HTML and CSS. There’s nothing to install and they can be hosted anywhere. All come with free updates and bug fixes forever. Inspired Guides were designed by Andy Clarke, who was talking about design systems before it was “the hip thing to do.” Andy’s helped clients including Greenpeace, Fairfax Media, SunLife, and
25% off : sydcss25

Valid for an arbitrary 7 days to encourage you buy sooner.
Only 50 codes available. May contain nuts.
speakerdeck.com/malarkey

Andy Clarke
@malarkey

SydCSS, October 2017
Thanks

Andy Clarke
@malarkey