# Designing with Sass


with Tjin Au Yeung

Codaisseur

# Table of Contents

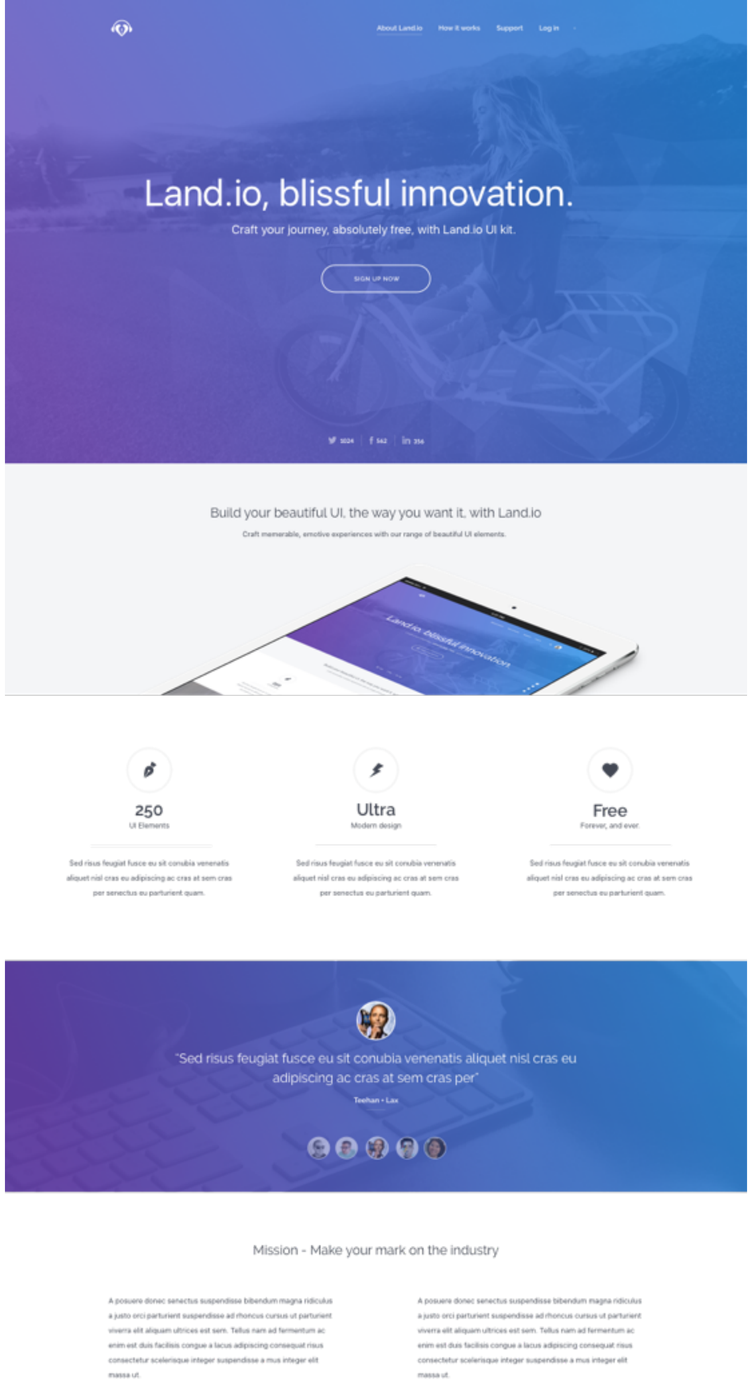
# What are we going to do?

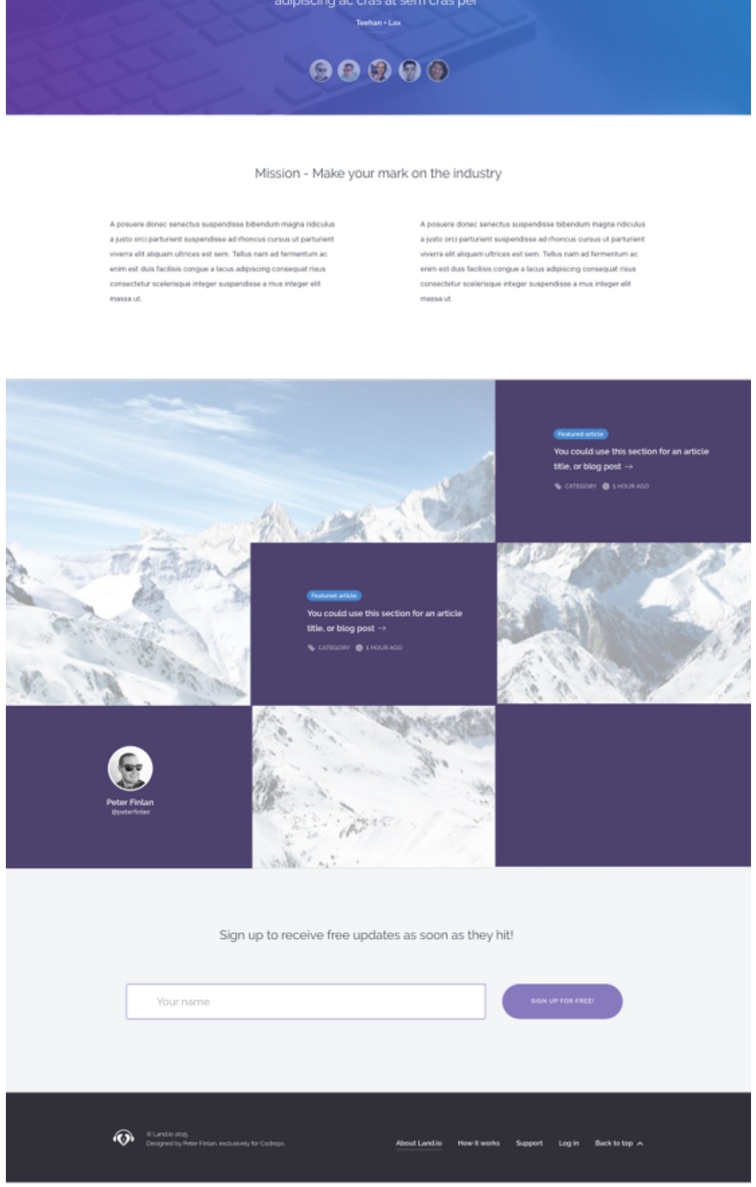
- Goal: style a page step by step using sass

Codaisseur

# Goal

- Style a page step by step using sass
- All assets are found in the git repository



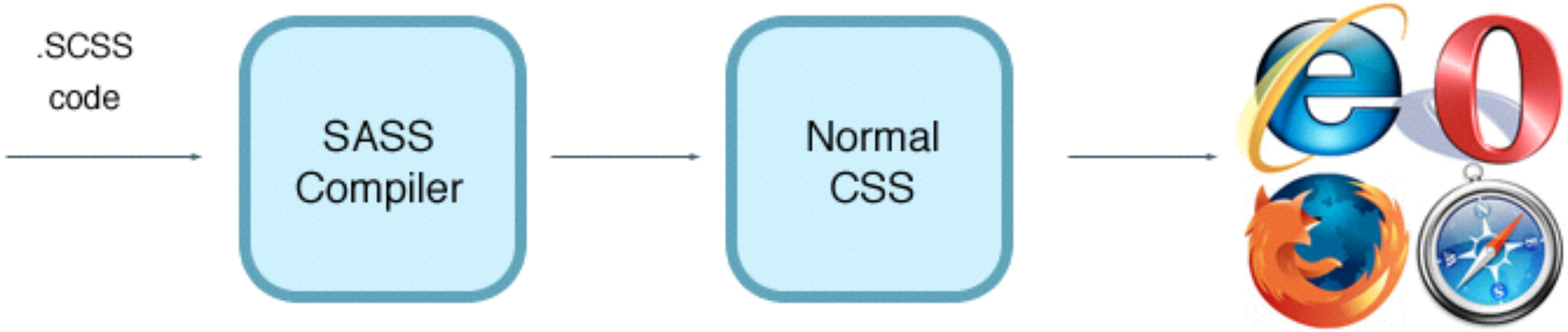




Codaisseur

# Introduction

- What is Sass?
- CSS preprocessor
- Ton of features helping you write semantic maintainable stylesheets

# How does it work?




Codaisseur

# Features

- Variables
- Nesting
- Mixins
- Extends
- Functions
- Lists
- and more!

Codaisseur

# Set up the compiler

- To compile the scss we can use different tools:
- We will be using Gulp.js

Codaisseur

# What is Gulp?

- Automated task runner

Codaisseur

# What is Gulp?

- Automated task runner
- Can be used to build and process your files automatically

Codaisseur

# What is Gulp?

- Automated task runner
- Can be used to build and process your files automatically
- It does this using gulp plugins

# What is Gulp?

- Automated task runner
- Can be used to build and process your files automatically
- It does this using gulp plugins
- Each task has its own plugin

Codaisseur

# What is Gulp?

- Automated task runner
- Can be used to build and process your files automatically
- It does this using gulp plugins
- Each task has its own plugin
- Gulp Sass

Codaisseur

# Setting up

To set up our work environment clone the following repository:

```
$ git clone git@github.com:tjinauyeung/sass_lesson.git
$ cd sass_lesson/sass_project
$ npm install
```

*

# Folder structure so far

- my_sass_project
  - sass_slides.key
  - sass_project
    - node_modules
    - gulpfile.js
    - package.son
    - src

Codaisseur

# Setting up - testing the configuration

Test out the setup and run ‘`gulp`’ in your command line. Note: you have to be in the same directory as your gulpfile.js

```
[18:54:07] Finished 'copyFiles' after 170 ms
[18:54:07] Starting 'scss'...
[BS] 1 file changed (style.css)
[18:54:07] Finished 'scss' after 15 ms
[18:54:07] Starting 'watch'...
[18:54:07] Finished 'watch' after 25 ms
```

*

# Exercise - Setting Up

Basic

Clone the repository with the basic configuration and set up the gulpfile.js and test it out.

Challenge

Add a gulp plugin for cleaning the dist directory and plugins for minifying the html and css

# Structuring your styles

- @import directive from Sass
- Stylesheet can be seperated into smaller chunks of code -> Partials
- Difference with CSS import?

Codaisseur

# Structuring your styles

- @import directive from Sass
- Stylesheet can be seperated into smaller chunks of code -> Partials
- Difference with CSS import?
  - No HTTP requests needed

Codaisseur

# Structuring Your Files

- Partials start with _
- So _mixins.scss, _variables.scss
- Partials will not be compiled to the css file unless imported

Codaisseur

# Structuring your files - Using Partials

_layout.scss

```
body {
  background: red;
}
```

style.scss

```
@import ‘layout’;
```

style.css

```
body {
  background: red;
}
```

# structuring your files - using partials

- scss
  - global
    - _variables.scss
    - _mixins.scss
    - _colors.scss
    - _typography.scss
    - _utilities.scss
  - components
    - _header.scss
    - _navigation.scss
    - _landing.scss
    - _cta.scss
    - _footer.scss
  - style.scss

style.scss

```
// Global vars & mixins
@import ‘global/variables’;
@import ‘global/mixins’;
@import ‘global/colors’;
@import ‘global/typography’;
@import ‘global/utilities’;

// Components
@import ‘components/header’;
@import ‘components/navigation’;
@import ‘components/landing’;
@import ‘components/cta’;
@import ‘components/footer’;
```

style.scss acts like a manifest file where all the styles are imported in

*

# Exercise - Structure

Basic

Set up a basic folder structure in your src directory

Challenge

Include a css reset as a partial in your folder and import it in the style.css manifest.

# Ask yourself

- Ever changed your mind about a color and had to update this in multiple location?
- Ever encountered a HEX value and had to check it in photoshop to see what color it actually is?

Codaisseur

# Variables to the rescue

- With variables colors, font sizes, margins, paddings and more can be named semantically and stored in a single location

Codaisseur

# Variables

_colors.scss

```
$primary-color: #0000FF;
$text-color: #FFFFFF;
```

_header.scss

```
header {
   background: $primary-color;
}

header__text {
   color: $text-color;
}
```

style.css

```
header {
  background: #0000FF;
}

header__text {
  color: #FFFFFF;
}
```

*

# Exercise - Variables (5min)

Basic
Setup a partial called variables and set up two or three colors, test it out on page

Challenge
Think of more useful variables that can be setup and include them in your variables partial

# Nesting

- Nesting is great in Sass
- It allows for more indentation in your styles and improves readability

Codaisseur

# Nesting

- Nesting is handy for these

| | | | |
|---|---|---|---|
| *element element* | div p | Selects all <p> elements inside  elements | 1 |
| *element>element* | div > p | Selects all <p> elements where the parent is a  element | 2 |
| *element+element* | div + p | Selects all <p> elements that are placed immediately after  elements | 2 |
| *element1~element2* | p ~ ul | Selects every <ul> element that are preceded by a <p> element | 3 |

http://www.w3schools.com/cssref/css_selectors.asp

Codaisseur

# Nesting

_header.scss

```
header {
  background: red;
  height: 200px;
  width: 100%;

  nav {
    list-style-type: none;
    padding: 0;
  }

  li {
    display: inline-block;
    color: $text-color;
  }
}
```

style.scss

```
header {
  background: red;
  height: 200px;
  width: 100%;
}

header nav {
  list-style-type: none;
  padding: 0;
}

header li {
  display: inline-block;
  color: $text-color;
}
```

*

# Nesting

- The use of &
- & will be replaced with the parent selectors its in
- Very useful for adding pseudo elements

Codaisseur

# Nesting

Pseudo elements syntax

| | | | |
|---|---|---|---|
| :active | a:active | Selects the active link | 1 |
| ::after | p::after | Insert something after the content of each <p> element | 2 |
| ::before | p::before | Insert something before the content of each <p> element | 2 |
| :checked | input:checked | Selects every checked <input> element | 3 |
| :disabled | input:disabled | Selects every disabled <input> element | 3 |
| :empty | p:empty | Selects every <p> element that has no children (including text nodes) | 3 |
| :enabled | input:enabled | Selects every enabled <input> element | 3 |
| :first-child | p:first-child | Selects every <p> element that is the first child of its parent | 2 |
| ::first-letter | p::first-letter | Selects the first letter of every <p> element | 1 |
| ::first-line | p::first-line | Selects the first line of every <p> element | 1 |
| :first-of-type | p:first-of-type | Selects every <p> element that is the first <p> element of its parent | 3 |
| :focus | input:focus | Selects the input element which has focus | 2 |
| :hover | a:hover | Selects links on mouse over | 1 |

Codaisseur

# Nesting

_header.scss

```
.header {
  background: red;
  height: 200px;
  width: 100%;

  &__nav {
    list-style-type: none;
    padding: 0;
  }

  li {
    display: inline-block;
    color: $text-color;

    &:hover {
      text-decoration: underline
    }
  }
}
```

style.scss

```
.header {
  background: red;
  height: 200px;
  width: 100%;
}

.header__nav {
  list-style-type: none;
  padding: 0;
}

header li {
  display: inline-block;
  color: $text-color;
}

header li:hover {
  text-decoration: underline
}
```

*

# Some advice

Don’t nest too deep because this affects the readability of styles and the specificity of the style rules.

Go 3 levels deep max!

Codaisseur

# Exercise - Nesting (15min)

Basic
Make use of nesting to finish up the splash page of the landing page

Challenge
Use the ampersand when possible

# Mixins

- Blocks of code that can be included
- Helps to keep your code DRY
- Won’t be rendered unless imported

# Nesting

_header.scss

```
@mixin flex-center {
  display: flex;
  align-items: center;
  justify-content: center;
}

.header {
  @include flex-center;

  height: 200px;
  width: 100%;
  background: $primary-gradient;
}
```

style.css

```
.header {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 200px;
  width: 100%;
  background:
    linear-gradient(#4C426D,#0D73D0);
}
```

*

# Some advice

Whenever you’re repeating styles three or more times, consider writing a mixin

Codaisseur

# Exercise - Mixins

Basic
Write a simple mixin for the buttons displayed on the page and include this in the file.

Challenge
Scan the page for more reusable ‘components’ and try writing mixins for them

# Functional mixins

- Mixins can also take up arguments and reuse them throughout the code - a bit like functions
- i.e. @mixin button($button-width, $button-color);

Codaisseur

# Mixins - leveled up

## Example of a gradient helper

_mixins.scss

```
@mixin background-gradient($start-color, $end-color) {
    background: $start-color;
    background: -webkit-linear-gradient(top, $start-color, $end-color);
    background: linear-gradient(to bottom, $start-color, $end-color);
}

.header {
  @include background-gradient($primary-color, $secondary-color);
}
```

# Mixins - leveled up

We can even add some extra functionality

_mixins.scss

```
@mixin background-gradient($start-color, $end-color, $orientation) {
    background: $start-color;

    @if $orientation == 'vertical' {
      background: -webkit-linear-gradient(top, $start-color, $end-color);
      background: linear-gradient(to bottom, $start-color, $end-color);
    } @else if $orientation == 'horizontal' {
      background: -webkit-linear-gradient(left, $start-color, $end-color);
      background: linear-gradient(to right, $start-color, $end-color);
    } @else {
      background: -webkit-radial-gradient(center, ellipse cover, $start-color, $end-color);
      background: radial-gradient(ellipse at center, $start-color, $end-color);
    }
}
```

*

# Exercise - Mixins (20min)

Basic
Search for a grid mixin and import it into your projectfolder and setup the grid section on the landing page.

Challenge
Try writing a grid mixin with the following arguments grid columns, gutter-width

# Functional mixins

- @content directive
- used to pass in a content block into a mixin
- example **media queries**

# Media Queries

- Media Query gist
- https://gist.github.com/tjinauyeung/5e91aa3d957060be734d9e1ae8cff7e3

Codaisseur

*

# Exercise - Mixins (20min)

Basic
Download the mediaquery mixin in your project and start make what you have so far responsive.

Challenge
Do the basic challenge but also add a responsive grid

# Extends

- Extends are a way to extend the styles of another class
- Instead of mixing a block of code in (a.k.a mixin), it works a little bit differently

Codaisseur

# slide.title

filename.extension

```
.text-input {
  height: 50px;
  border: 1px solid #DEDEDE;
  width: 450px;
  border-radius: 3px;
}

input {
  extend .text-input;

  font-family: sans-serif;
}
```

filename.extension

```
input,
.text-input {
  height: 50px;
  border: 1px solid #DEDEDE;
  width: 450px;
  border-radius: 3px;
}

input {
  font-family: sans-serif;
}
```

# Extends

- Extends save a bit of code because it’s extending existing rules.

Codaisseur

# Extends

- Another feature is the placeholder class using % syntax
- i.e. %flexbox-center

Codaisseur

# Some advice

The output of extend is less obvious than mixins, especially when they're being used in multiple locations - keep that in mind when using it

Codaisseur

# slide.title

filename.extension

```
%flexbox-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  extend %flexbox-center;
}

.signup {
  extend %flexbox-center;
}
```

filename.extension

```
.container,
.signup {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

# Exercise - Extends ()

Basic
Make a partial in the global folder called _extends.scss and add a placeholder class i.e. %clearfix

# Lists

- Sass provides possibility to store lists inside a variable

```
$list: (
 ‘key’: ‘value’,
 ‘key2: ‘value2’
 );
```

Codaisseur

# Lists

- How lists can be utilized for setting font faces

  https://gist.github.com/tjinauyeung/46e90462a5ee26c3e0ed8e40a43421b4

Codaisseur

# Final Challenge

- Finish the landing page
- Keep your file structure organized
- Keep your code DRY
- Make it responsive

**Good luck!**

# Questions?

:{)

# Further research

**Mixin libraries**

- bourbon.io
- bootstrap framework
- foundation framework

**Digital playground**

- sassmeister.com
- codepen.io

**Good reads**

- www.thesassway.com
- www.sitepoint.com
- https://css-tricks.com/sass-style-guide/
- www.drupalnorth.org/sites/default/files/inline-files/2016-Drupal-North-Mainspring_v10.pdf

**Lesson repo**

https://github.com/tjinauyeung/sass_lesson

Codaisseur