UNIT TESTING IN SASS

Lindsey Wild

SPARKBOX

Slides

https://bit.ly/2Nafemi

We're Hiring! 🎉

https://seesparkbox.com/jobs

WHAT WILL WE LEARN?

Main Points

- Refresher on Sass and Unit Testing
- Why & When
  - Use cases
  - Real life scenarios
  - What to test
- How
  - Using True
  - Best practices for unit testing

General Takeaways

- Unit testing best practices
- Evaluating if a project would be a good candidate for Sass unit testing

Design System

1 Design System, 6 Brands

6 Brands = Multiple websites

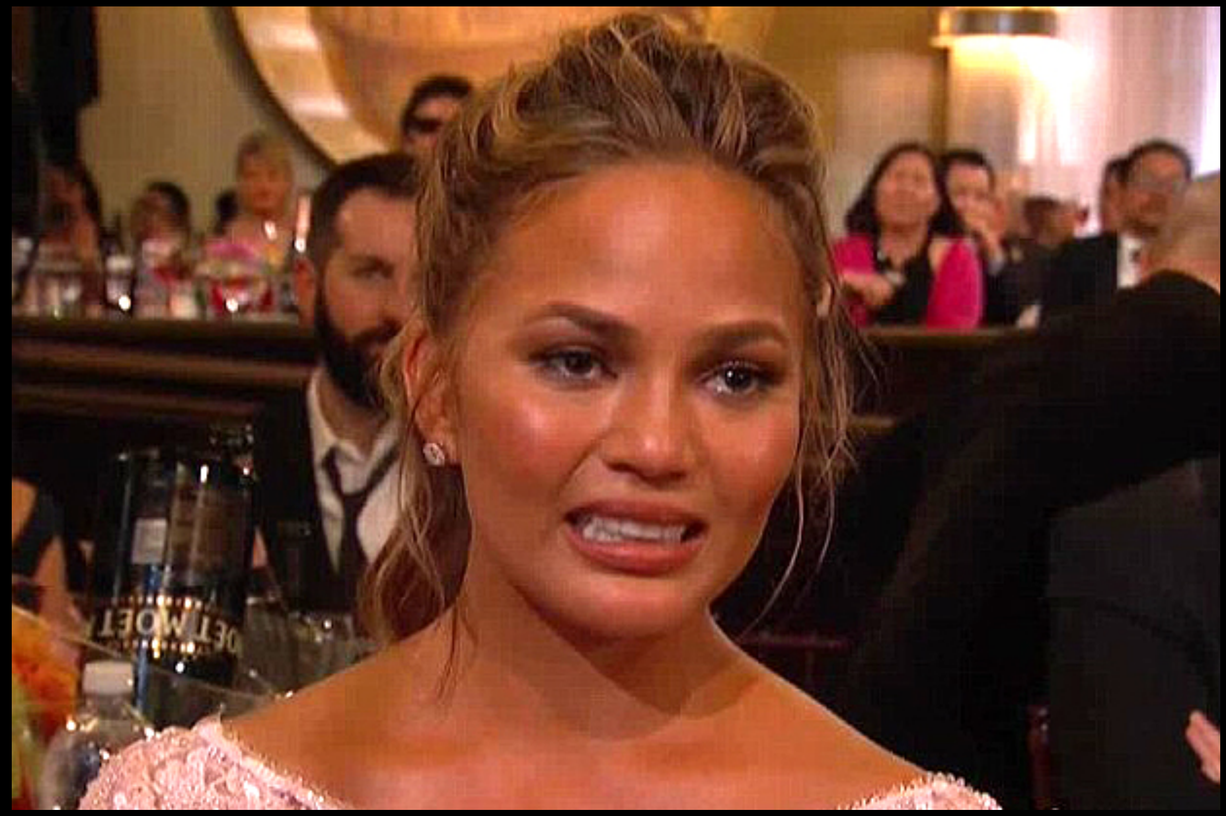
Multiple websites = so many teams!

So many teams = so many subscribers!

We’re perfect though.

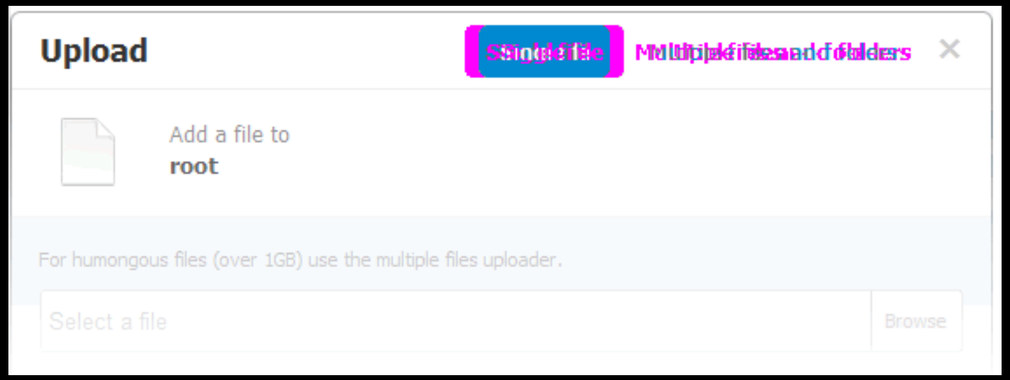
LOL JK

We've made mistakes.



38 SCSS Files
Each brand: ~8,000 lines of CSS

Visual Regression Testing Tools



Screenshot from PhantomCSS: https://github.com/HuddleEng/PhantomCSS

Can you Unit Test your Sass?

WHAT IS SASS?

Sass is a CSS Preprocessor

- More efficient, can use variables, nesting
- Break files into smaller chunks (partials) for better organization, more component driven
- Write functions and mixins to output conditional CSS

WHAT IS UNIT TESTING?

Unit Testing: Testing “Units”

- Test small pieces of code, aka “units”
- Ensure functions and other logic is working as expected
- Isolate issues within smaller pieces for faster debugging

WHY & WHEN TO UNIT TEST SASS

Why?

- Alleviate some manual UI regression testing
- Ensure output from mixins/functions is as expected
  - Especially third-party libraries, or Sass itself
- Double check important updates, such as changing the main color
- CSS could still compile without errors
- Ship less errors!

When

- Large project with numerous Sass files/partials
- Components exist in many places/contexts
- Third-party library with Sass functions/mixins
- You've written complex Sass functions/mixins
- Styles are very important and if they change, it could break the UI dramatically
- Project supports multiple users or "subscribers"

When *not* to

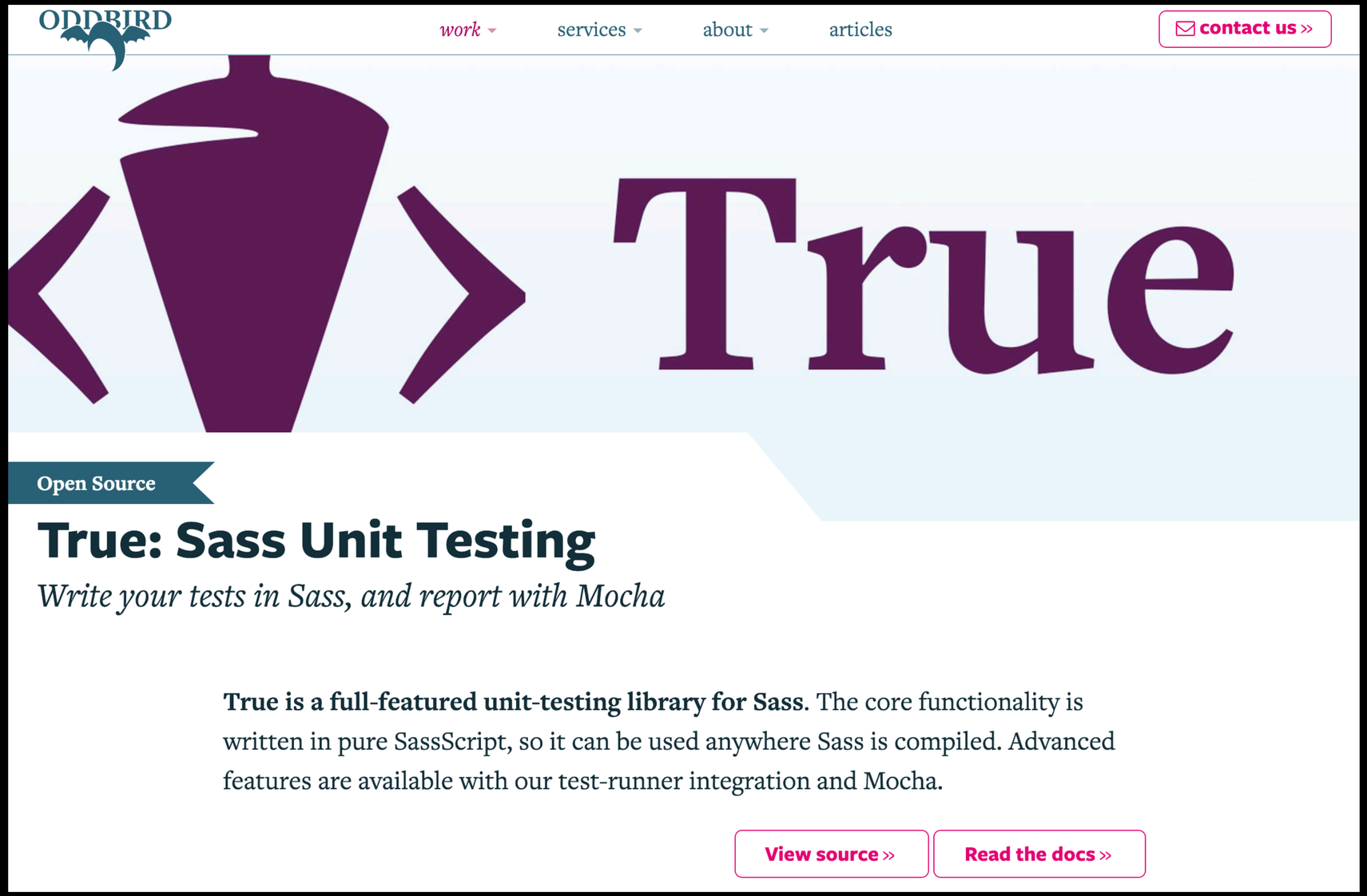
- Small project with little CSS, such as a landing page or small marketing site
- Static content websites
- Project doesn’t heavily rely on a consistent UI
- Project doesn’t use Sass mixins or functions

Real Life Example!

What to Unit Test

- Functions
  - Especially ones that have calculations or take parameters
- Mixins
- Any important output
- Variables? Probably overkill

HOW TO SET IT UP IN YOUR PROJECT



https://oddbird.net/true

Requirements

- A computer
- Yarn or npm
- Node-sass or another way to compile Sass to CSS
- Optional: A JS testing library such as Jasmine or Mocha

Install the `sass-true` Module

```
// In your Terminal
 yarn add sass-true --dev
 OR
 npm install sass-true —save-dev
```

Set Up Your File Structure

```
my-cool-project/                          # Root project directory
|-- src/                                  # Project files
|   |-- sass/                             # Sass files
|   |-- js/
|   |-- index.html
|-- spec/                                  # Jasmine tests & setup
|   |-- true-sass-tests/                   # Sass unit tests
|   |   |-- _typography-tests.scss         # Sass unit test partial
|   |   |-- _font-mixin-tests.scss
|   |   |-- _font-function-tests.scss
|   |   |-- sass_tests.scss                # Import Sass test partials
|   true-sass-spec.js                      # True setup
|   ...                                    # Other Jasmine tests/setup
```

Import the Library & Your Sass

sass_tests.scss

```
// Path to module
@import "node_modules/sass-true/sass/true";
// Path to Sass file(s) to test
@import "../css/styles";
```

Set Up The Sass Tests

true-sass-spec.js

```
// Require the module
const sassTrue = require('sass-true');
// Path to test file
const sassFile = '../true-sass-tests/sass-tests.scss');
// Tell True your file and testing pattern
sassTrue.runSass({ file: sassFile }, describe, it);
```

Testing a Function

```
// Sass
@function multiply($value1, $value2) {
  @return $value1 * $value2;
}

// CSS where function is called
.header {
  width: multiply(12, 2); //24
}

// Test
@include describe('The multiply() function') {
  @include it('Returns the result of multiplication') {
    @include assert-equal(
      // ...
    );
  }
}
```

Testing a Function

- Compare internal Sass values (variables and functions) by asserting is-equal, is-unequal, is-true, or is-false:

```
@include describe('The multiply() function') {
  @include it('Returns the result of multiplication') {
    @include assert-equal(
      multiply(12, 2),
      24
    );
  }
}
```

Terminal Output

`npm run test`

- Passing Test

```
The multiply() function
  ✓ Returns the result of multiplication
```

- Failing Test

```
The multiply() function
  ✗ Returns the result of multiplication
    - AssertionError [ERR_ASSERTION]: Returns the result of multiplication ("[number] 24" assert-equal "[number] 26")
```

Testing a Mixin

- Test CSS output of mixins with assert, output, and expect (or contains)

```
@include describe('The large font mixin') {
  @include it('Returns correct font properties') {
    @include assert {
      @include output {
        // @include mixin to test
      }

      @include expect {
        // font properties
      }
    }
  }
}
```

Testing a Mixin

```sass
// Sass
@mixin font-size($size) {
  @if $size == 'large' {
    font-size: 2rem;
    line-height: 3rem;
  }
}
```

```sass
// CSS using mixin
.header {
  @include font-size('large');
}
```

```sass
// Test
@include describe('The large font mixin') {
  @include it('Returns correct font properties') {
    @include assert {
      @include output {
        @include font-size('large');
      }

      @include expect {
        font-size: 2rem;
        line-height: 3rem;
      }
    }
  }
}
```

Mixin Terminal Output

`npm run test`

- Passing Test

```
The large font mixin
  ✓ Returns correct font properties
```

- Failing Test

```
    The large font mixin
      ✗ Returns correct font properties
        - AssertionError [ERR_ASSERTION]:  (".test-output {
  font-size: 2rem;
  line-height: 3rem;
}" equal ".test-output {
  font-size: 2rem;
  line-height: 2rem;
}")
```

TESTING BEST PRACTICES

UNIT TESTING IN SASS

Testing Best Practices

- Only test your code, not a dependency
- Write tests like a readable sentence
- Test small chunks at a time (“unit”)
- Don’t test everything

Recap

- Refresher on Sass and Unit Testing
- Why & When
  - Use cases
  - Real life scenarios
  - What to test
- How
  - Using True
  - Best practices for unit testing

RESOURCES

Links and Resources

- https://oddbird.net/true/docs/
- https://github.com/oddbird/true
- https://seesparkbox.com/foundry/how_and_why_we_unit_test_our_sass
- https://jasmine.github.io/
- https://twitter.com/stananick
- https://bit.ly/2Nafemi

THANKS!

@stananick

wild@heysparkbox.com

SPARKBOX