Unit testy v SCSS

Jan Kryšpín
Alma Career Czechia, Netvor

krysp.in

Frontendisti – 15. 5. 2024

Jobs.cz, Prace.cz,
Seduo, Teamio, Spirit, ...

Design System Architect

Respekt.cz, Fondee.cz,
Plakatov.cz, ...

Frontend Team Lead

Spirit Design System

maintained with lerna | Code Quality Checks passing | coverage 81%

Spirit is an open-source design system developed by Alma Career (formerly LMC).

Packages

Package name	Description	Version
@lmc-eu/spirit-design-tokens	Design tokens for Spirit Design System	npm v1.1.5
@lmc-eu/spirit-web	CSS and vanilla JS implementation of Spirit Design System	npm v1.13.1
@lmc-eu/spirit-web-react	React implementation of Spirit Design System components	npm v1.16.1
@lmc-eu/spirit-web-twig	Twig implementation of Spirit Design System components	packagist v2.14.1

https://github.com/lmc-eu/spirit-design-system

Tag

Tags can highlight or add emotion to information.

Documentation

© 2024 Alma Career

Button

Buttons allow users to take actions.

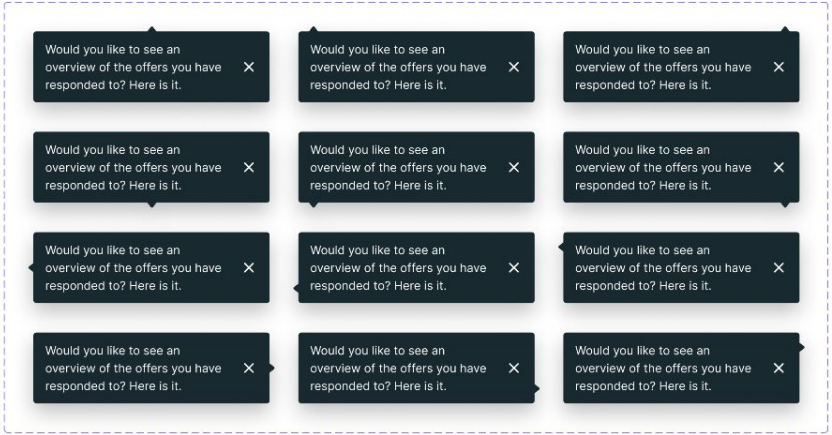
Documentation

© 2024 Alma Career

Tooltip

Tooltip displays additional information without interrupting user flow.

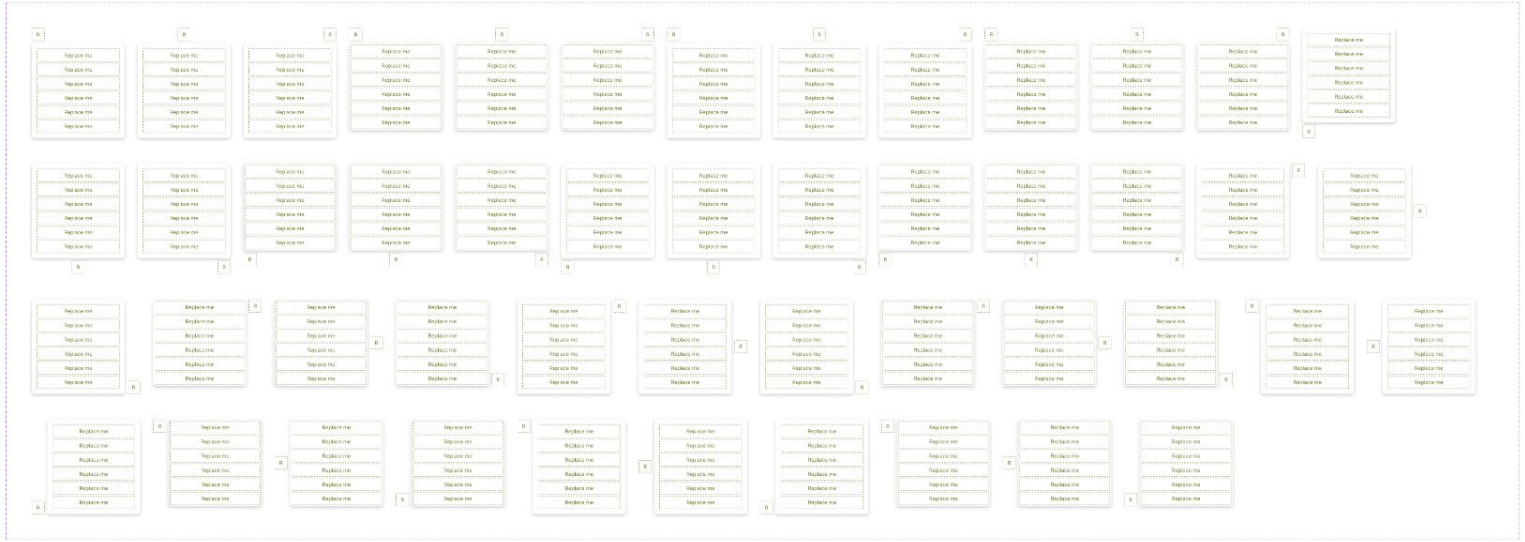
Documentation



Dropdown

A Dropdown is a user interface element commonly used in web forms or menus. Dropdowns are used to conserve screen space and provide a convenient way for users to make selections from a set of predefined options.

Documentation



© 2024 Alina Cansel

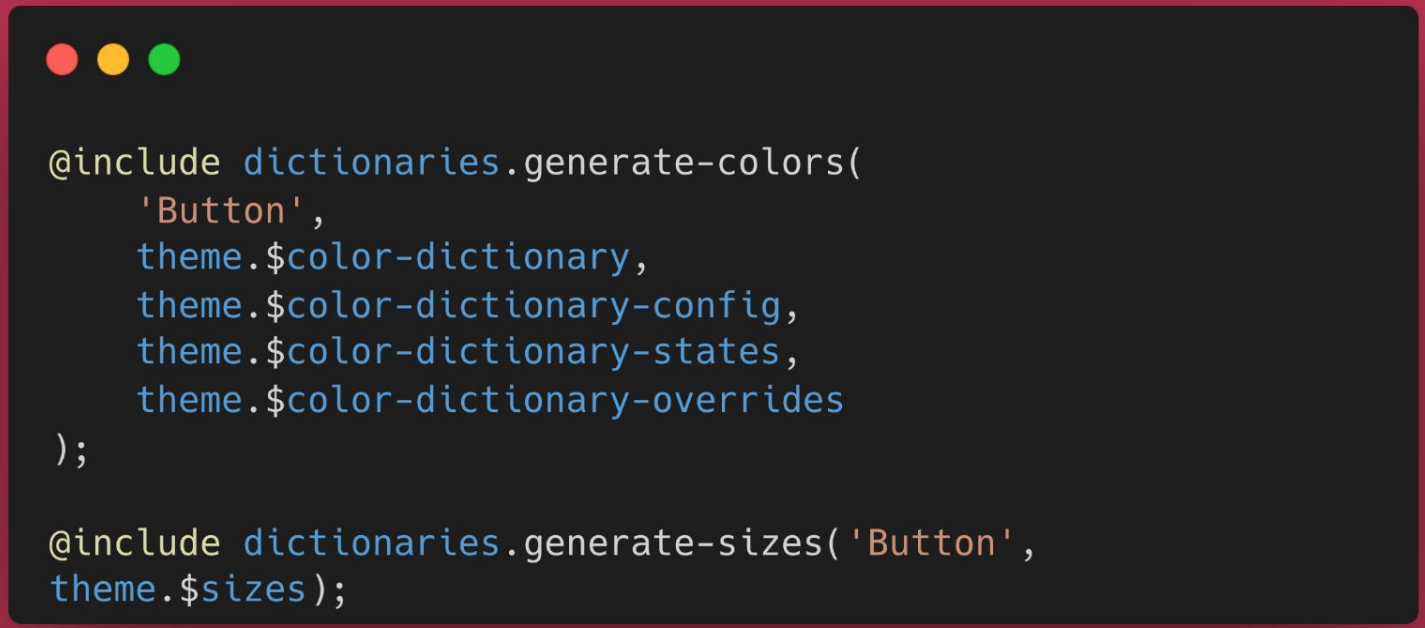
Slovníky

```scss
@use 'sass:list';
@use '@tokens' as tokens;

$_placement-sides: top, bottom, left, right;
$_placement-side-corners: top-start, top-end, bottom-start, bottom-end, left-start, left-end, right-start, right-end;

$_size-extended: xsmall, xlarge;

$action-colors: primary, secondary, tertiary, inverted;
$action-link-colors: primary, secondary, inverted;
$emotion-colors: success, informative, warning, danger;
$placement: list.join($_placement-sides, $_placement-side-corners);
$size: small, medium, large;
$size-extended: list.join($size, $_size-extended);
$text-colors: primary, secondary, primary-inverted, secondary-inverted;
```



```
@include dictionaries.generate-colors(
    'Button',
    theme.$color-dictionary,
    theme.$color-dictionary-config,
    theme.$color-dictionary-states,
    theme.$color-dictionary-overrides
);

@include dictionaries.generate-sizes('Button',
theme.$sizes);
```

```scss
@mixin generate-colors(
    $class-name,
    $dictionary-values,
    $config,
    $states: null,
    $overrides: null,
    $custom-variable-name: null,
    $child-selector: null
) {
    @each $dictionary-value in $dictionary-values {
        .#{$class-name}--#{$dictionary-value}#{$child-selector} {
            @each $property-name, $property-value in $config {
                $value: -get-property-value($dictionary-value, $property-name, $property-value, $overrides);
                $prefix: if(
                    $custom-variable-name,
                    $custom-variable-name,
                    spirit-string.convert-pascal-case-to-kebab-case($class-name)
                );

                --#{$prefix}-#{$property-name}: #{$value};

                #{$property-name}: var(--#{$prefix}-#{$property-name});
            }

            @if $states {
                @each $state-name, $state-properties in $states {
                    #{$state-name} {
                        @each $property-name, $property-value in $state-properties {
                            #{$property-name}: -get-property-value(
                                $dictionary-value,
                                $property-name,
                                $property-value,
                                $overrides,
                                $state-name
                            );
}}}}}}}
```

```
@include dictionaries.generate-colors(
  'Test',
  ('primary', 'secondary'),
  (
    color: 'default',
  )
);

// Output
.Test--primary {
  --test-color: #29616f;

  color: var(--test-color);
}

.Test--secondary {
  --test-color: #a0a0a0;

  color: var(--test-color);
}
```

```
.Button--primary {
    --button-color: #fff;
    color: var(--button-color);
    --button-border-color: #29616f;
    border-color: var(--button-border-color);
    --button-background-color: #29616f;
    background-color: var(--button-background-color)
}

.Button--primary:hover,.Button--primary:focus {
    color: #fff;
    border-color: #1b5260;
    background-color: #1b5260
}

.Button--primary:active {
    color: #fff;
    border-color: #0b3a46;
    background-color: #0b3a46
}

.Button--primary:disabled:not(.Button--loading),.Button--primary.is-disabled:not(.Button--
loading),.Button--primary.Button--disabled:not(.Button--loading) {
    color: #737373;
    border-color: #c4c4c4;
```

```scss
// Function to transform placement
// Example: transform('top-start') will return 'top-start' (no transformation)
// Example: transform('top-start', $main-axis-inverse: true) will return 'bottom-start'
// Example: transform('top-start', $main-axis-inverse: true, $cross-axis-inverse: true) will return 'bottom-end'
// Example: transform('top-start', $main-axis-inverse: true, $cross-axis-inverse: true, $cross-axis-physical: true) will return 'bottom-right'
// Example: transform('top-start', $join-with: ' ') will return 'top start'
@function transform(
    $placement,
    $main-axis-inverse: false,
    $cross-axis-inverse: false,
    $cross-axis-physical: false,
    $join-with: $_hyphen
) {
    $placement-chunks: string.split($placement, $_hyphen);
    $main-axis: list.nth($placement-chunks, 1);
    $main-axis-transformed: transform-axis(
        $axis: $main-axis,
        $inverse: $main-axis-inverse,
    );

    @if list.length($placement-chunks) > 1 {
        $cross-axis: list.nth($placement-chunks, 2);
        $cross-axis-direction: get-cross-axis-direction($main-axis);
        $cross-axis-transformed: transform-axis(
            $axis: $cross-axis,
            $inverse: $cross-axis-inverse,
            $physical-direction: if($cross-axis-physical, $cross-axis-direction, null),
        );

        @return spirit-list.to-string(($main-axis-transformed, $cross-axis-transformed), $join-with);
    }

    @return $main-axis-transformed;
}
```

sass-true

8.0.0 • Public • Published 3 months ago

Readme | Code Beta | 3 Dependencies | 26 Dependents | 26 Versions

True

License BSD 3-Clause

1. To make true; shape, adjust, place, etc., exactly or accurately:

   True the wheels of a bicycle after striking a pothole.

2. To make even, symmetrical, level, etc. (often followed by *up*):

   True up the sides of a door.

3. To test your Sass code; debug, perfect, etc. (often using *True*):

   True your sweet plugin before you deploy.

True is a unit-testing tool for Sass code. All of the tests are written in plain Sass, and can be

Install

```
> npm i sass-true
```

Repository

github.com/oddbird/true

Homepage

www.oddbird.net/true/

Weekly Downloads

14,950

Version

8.0.0

License

BSD-3-Clause

View Project View Source

True

License BSD 3-Clause

1. To make true; shape, adjust, place, etc., exactly or accurately:

   True the wheels of a bicycle after striking a pothole.

2. To make even, symmetrical, level, etc. (often followed by *up*):

   True up the sides of a door.

3. To test your Sass code; debug, perfect, etc. (often using *True*):

   True your sweet plugin before you deploy.

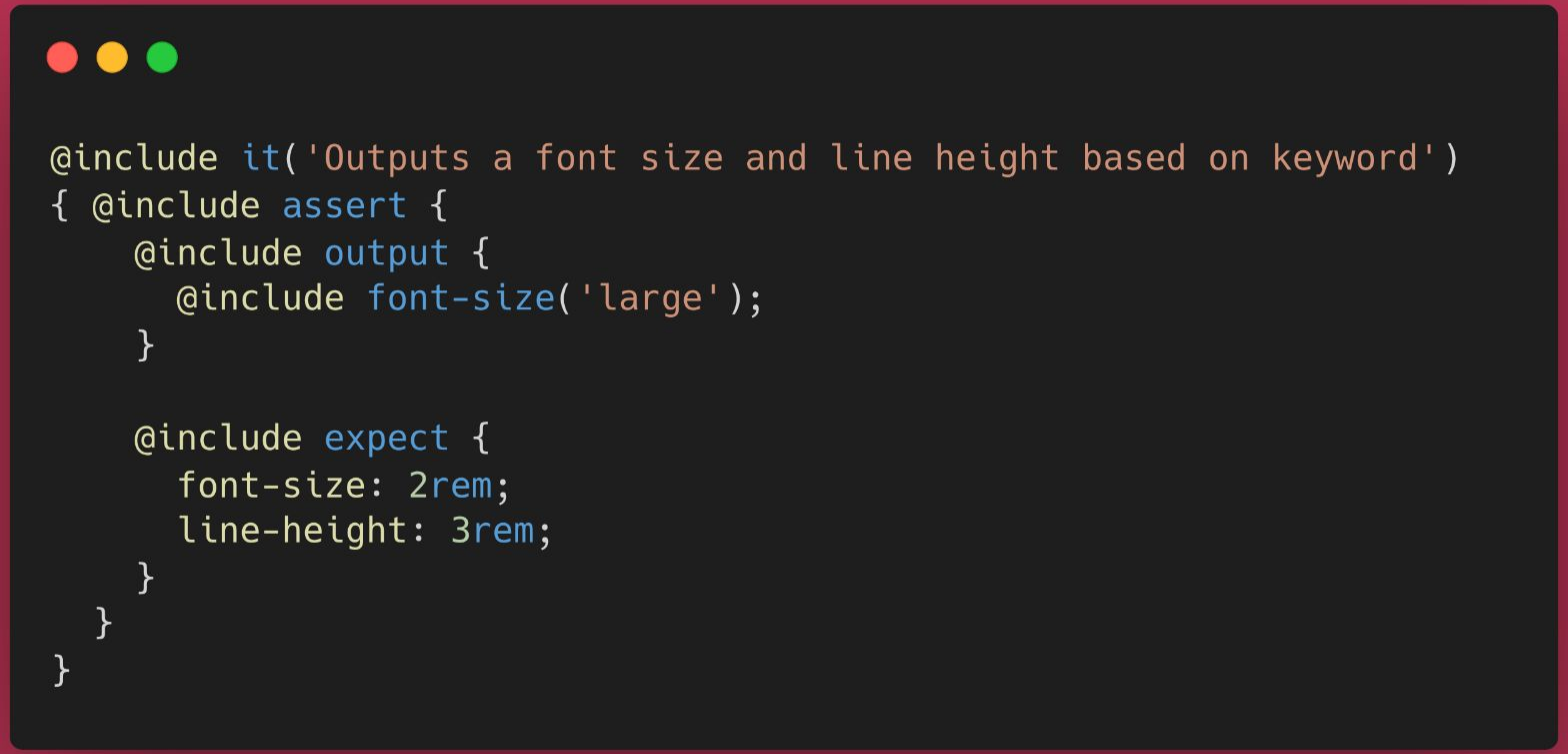
True is a unit-testing tool for Sass code. All of the tests are written in plain Sass, and can be compiled using Dart Sass – but we also provide integration with JavaScript test runners (e.g. Mocha or Jest), for extra features and improved reporting.

Install

https://www.oddbird.net/true/docs/

```
@include test-module('Zip [function]') {
  @include test('Zips multiple lists into a single multi-dimensional list')
{   // Assert the expected results
    @include assert-equal(zip(a b c, 1 2 3), (a 1, b 2, c 3));
  }
}

// Or
@include describe('Zip [function]') {
  @include it('Zips multiple lists into a single multi-dimensional list') {
    // Assert the expected results
    @include assert-equal(zip(a b c, 1 2 3), (a 1, b 2, c 3));
  }
}
```



```
@include it('Outputs a font size and line height based on keyword')
{ @include assert {
    @include output {
      @include font-size('large');
    }

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

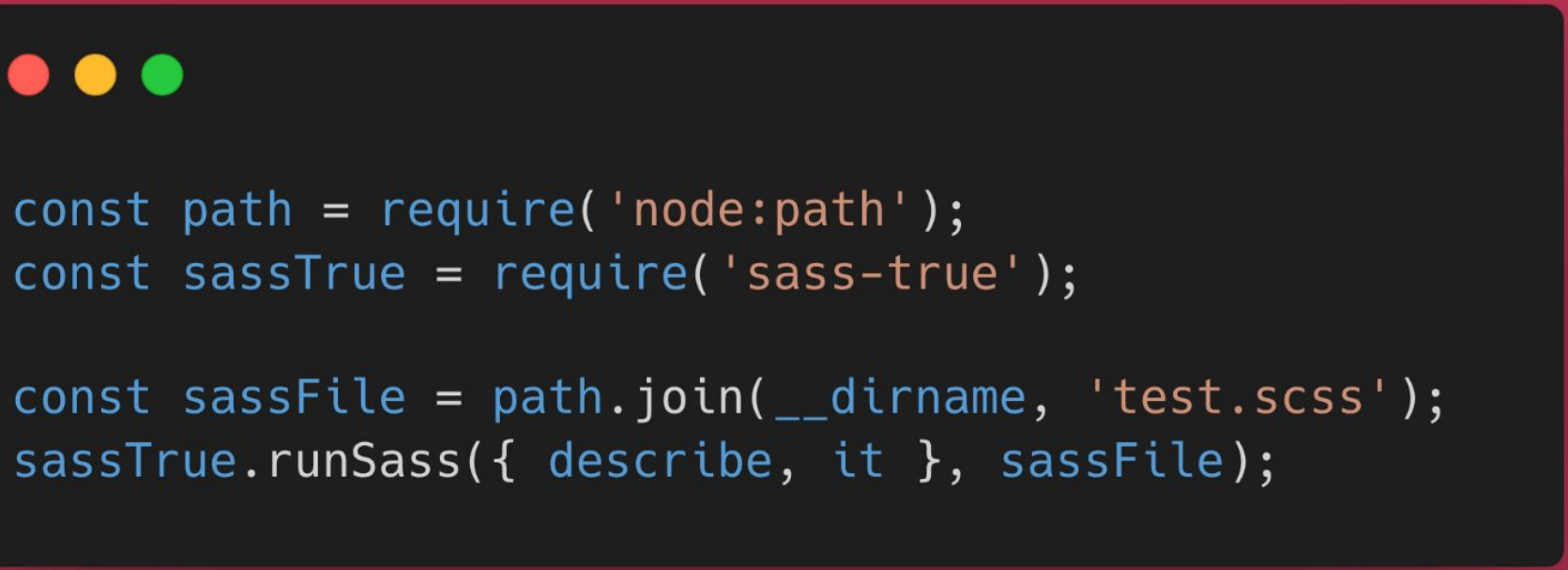
```
@use 'true';
@use '../dictionaries';

@include true.describe('generate-colors mixin') {
    @include true.it('should generate correct color classes based on a dictionary')
{       @include true.assert {
            @include true.output {
                @include dictionaries.generate-colors(
                    'Test',
                    ('primary'),
                    (
                        color: 'default',
                    )
                );
            }

            @include true.expect {
                .Test--primary {
                    --test-color: #29616f;

                    color: var(--test-color);
                }
            }
        }
```

```
@include true.it('should return true for logical placements, false otherwise')
{ @include true.assert {
    @include true.output {
      --is-logical-left: #{placement.is-logical('left')};
      --is-logical-start: #{placement.is-logical('start')};
    }
    @include true.expect {
      --is-logical-left: false;
      --is-logical-start: true;
    }
  }
}
```



```
const path = require('node:path');
const sassTrue = require('sass-true');

const sassFile = path.join(__dirname, 'test.scss');
sassTrue.runSass({ describe, it }, sassFile);
```

```ts
// scss.test.ts
/* @jest-environment node */

import { sync } from 'glob';
import { resolve } from 'path';
import { runSass } from 'sass-true';
import { pathToFileURL } from 'url';

const importers = [
  // Make @tokens work
  {
    findFileUrl(url) {
      if (!url.startsWith('@')) {
        return null;
      }

      return new URL(
        pathToFileURL(resolve(process.cwd(), '../../node_modules/@lmc-eu/spirit-design-tokens/src/scss', url)),
      );
    },
  },
];

describe('Sass', () => {
  const sassTestFiles = sync(resolve(process.cwd(), 'src/**/*.test.scss'));

  sassTestFiles.forEach((file) => runSass({ describe, it }, file, { importers }));
});
```

```
PASS src/js/__tests__/BaseComponent.test.ts
PASS src/js/dom/__tests__/EventHandler.test.ts
PASS src/js/utils/__tests__/ComponentFunctions.test.ts
PASS tests/scss.test.ts

Test Suites: 23 passed, 23 total
Tests:       256 passed, 256 total
Snapshots:   0 total
Time:        2.298 s, estimated 7 s
Ran all test suites.
```

```
@include true.it('should return true for logical placements, false otherwise')
{ @include true.assert {
    @include true.output {
      --is-logical-left: #{placement.is-logical('left')};
      --is-logical-start: #{placement.is-logical('start')};
    }
    @include true.expect {
      --is-logical-left: false;
      --is-logical-start: true;
    }
  }
}
```

```
@include true.it('should return true for logical placements, false otherwise')
{ @include true.assert {
    @include true.output {
      --is-logical-left: #{placement.is-logical('left')};
      --is-logical-start: #{placement.is-logical('start')};
    }
    @include true.expect {
      --is-logical-left: true; // <-- wrong
      --is-logical-start: true;
    }
  }
}
```

```
FAIL tests/scss.test.ts
  ● Sass › placement functions and mixins › should return true for logical placements, false otherwise

    assert.fail(received, expected)

    Message:
       [type: equal]

    - Expected
    + Received

      .test-output {
    -   --is-logical-left: true;
    +   --is-logical-left: false;
        --is-logical-start: true;
      }


      at ../../node_modules/sass-true/src/index.ts:125:20
      at arrayEach (../../node_modules/lodash/lodash.js:530:11)
      at forEach (../../node_modules/lodash/lodash.js:9410:14)
      at Object.<anonymous> (../../node_modules/sass-true/src/index.ts:123:16)


Test Suites: 1 failed, 22 passed, 23 total
```

```scss
@mixin hide-text() {
    position: absolute;
    width: 1px;
    height: 1px;
    padding: 0;
    margin: -1px;
    overflow: hidden;
    clip: rect(0, 0, 0, 0);
    white-space: nowrap;
    border: 0;
}
```

```scss
@mixin min-tap-target($size, $center: true) {
    position: relative;

    &::before {
        content: '';
        position: absolute;
        width: $size;
        height: $size;

        @if $center {
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
        }
    }
}
```

```scss
@include true.expect {
  .tap-target-centered {
    position: relative;
  }

  // stylelint-disable order/properties-order -- Disabling rule due to conditional rendering
affecting property order
  .tap-target-centered::before {
    content: '';
    position: absolute;
    width: 40px;
    height: 40px;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  // stylelint-enable order/properties-order
}
```

literat commented on Oct 27, 2023 Member

👋 I am very happy to see this PR. It really helps me to understand what is going on in the SCSS mixins 😅 Thanks a lot ❤️

Alternativy

- https://www.npmjs.com/package/sass-true ✅
- https://www.npmjs.com/package/sassaby 🤷
- https://www.npmjs.com/package/sassy-test 🤷

Díky za pozornost, dotazy?

krysp.in