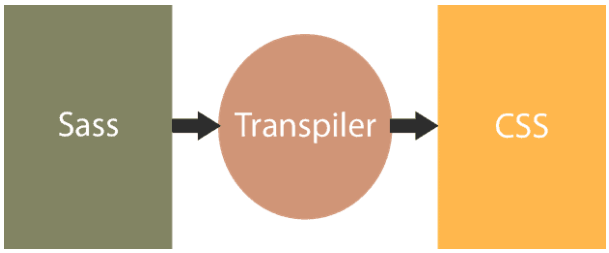
SCSS Interview Questions

Q. *Explain what is Sass? How it can be used?*

When stylesheets are getting larger, more complex, and harder to maintain. This is where a CSS pre-processor can help. Sass (which stands for 'Syntactically awesome style sheets) is an extension of CSS that enables you to use things like variables, nested rules, inline imports and more. It also helps to keep things organised and allows you to create style sheets faster.

Sass works by writing your styles in .scss (or .sass) files, which will then get compiled into a regular CSS file. The newly compiled CSS file is what gets loaded to your browser to style your web application. This allows the browser to properly apply the styles to your web page.



Live Demo: Sass Example

↥ back to top

Q. *What are the SCSS basic features?*

1. Variables

Variables are useful for things like colors, fonts, font sizes, and certain dimensions, as you can be sure always using the same ones. Variables in SCSS start with `$` sign

SCSS Style

```scss
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

CSS Style

```css
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

When the Sass is processed, it takes the variables we define for the `$font-stack` and `$primary-color` and outputs normal CSS with our variable values placed in the CSS. This can be extremely powerful when working with brand colors and keeping them consistent throughout the site.

Live Demo: Sass Variables

2. Nesting

Basic nesting refers to the ability to have a declaration inside of a declaration.

SCSS Style

```scss
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
```

```
    li { display: inline-block; }

    a {
      display: block;
      padding: 6px 12px;
      text-decoration: none;
    }
  }
```

CSS Style

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Live Demo: Sass Nesting

3. Partials

The partial Sass files contain little snippets of CSS that can be included in other Sass files. This is a great way to modularize your CSS and help keep things easier to maintain. A partial is a Sass file named with a leading underscore. You might name it something like `_partial.scss`. The underscore lets Sass know that the file is only a partial file and that it should not be generated into a CSS file. Sass partials are used with the `@use` rule.

4. Modules

This rule loads another Sass file as a module, which means we can refer to its variables, mixins, and functions in our Sass file with a namespace based on the filename. Using a file will also include the CSS it generates in your compiled output!

SCSS Style

```
// _base.scss
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;
```

```
body {
  font: 100% $font-stack;
  color: $primary-color;
}



// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

CSS Style

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}

.inverse {
  background-color: #333;
  color: white;
}
```

5. Mixins

A mixin provide to make groups of CSS declarations that you want to reuse throughout your site. You can even pass in values to make your mixin more flexible.

SCSS Style

```
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}
.box { @include transform(rotate(30deg)); }
```

CSS Style

```
.box {
  -webkit-transform: rotate(30deg);
  -ms-transform: rotate(30deg);
```

```
    transform: rotate(30deg);
  }
```

Live Demo: Sass Mixins

6. Inheritance

Using `@extend` lets you share a set of CSS properties from one selector to another.

SCSS Style

```
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

.success {
  @extend %message-shared;
  border-color: green;
}

.error {
  @extend %message-shared;
  border-color: red;
}

.warning {
  @extend %message-shared;
  border-color: yellow;
}
```

CSS Style

```
/* This CSS will print because %message-shared is extended. */
.message, .success, .error, .warning {
  border: 1px solid #ccc;
  padding: 10px;
```

```
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

Live Demo: Sass Inheritance

7. Operators

Sass has a handful of standard math operators like `+`, `-`, `*`, `/`, and `%`. In our example we're going to do some simple math to calculate widths for an aside & article.

SCSS Style

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complementary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

CSS Style

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 62.5%;
}
```

```
aside[role="complementary"] {
  float: right;
  width: 31.25%;
}
```

Note: Only Dart Sass currently supports `@use`. Users of other implementations must use the `@import` rule instead.

↥ back to top

Q. *List out the data types that Sass supports?*

↥ back to top

Q. *Explain the @include, @mixin, @function functions and how they are used. What is %placeholder?*

i) `@mixin` A mixin lets you make groups of CSS declarations that you want to reuse throughout your site

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}


.box { @include border-radius(10px); }
```

ii) `@extend` directive provides a simple way to allow a selector to inherit/extend the styles of another one.

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}

.error {
  @extend .message;
```

```
  border-color: red;
}
```

iii) `%placeholder` are classes that aren't output when your SCSS is compiled

```
%awesome {
    width: 100%;
    height: 100%;
}
body {
    @extend %awesome;
}
p {
    @extend %awesome;
}


/* Output */
body, p {
    width: 100%;
    height: 100%;
}
```

↥ back to top

Q. *List out the differences between LESS and Sass?*

LESS	Sass
– LESS uses JavaScript and processed at client-side	– Sass is coded in Ruby and thus processed to server-side
– Variable names are prefaced with the @symbol	– Variable name are prefaced with $ symbol
– LESS does not inherit multiple selectors with one set of properties	– Sass inherits multiple selectors with one set of properties
– LESS does not work with "unknown" units neither it returns syntax error notification for incompatible units or maths related syntax error	– Sass allows you to work with "unknown" units also returns a syntax error notification for incompatible units

↥ back to top

Q. *Why Sass is considered better than LESS?*

- Saas allows you to write reusable methods and use logic statements, e., loops, and conditionals
- Saas user can access Compass library and use some awesome features like dynamic sprite map generation, legacy browser hacks * and cross-browser support for CSS3 features
- Compass also allows you to add an external framework like Blueprint, Foundation or Bootstrap on top
- In LESS, you can write a basic logic statement using a 'guarded mixin', which is equivalent to Sass if statements
- In LESS, you can loop through numeric values using recursive functions while Sass allows you to iterate any kind of data
- In Sass, you can write your own handy functions

↥ back to top

Q. *What are Sass, Less, and Stylus? Why do people use them? How does something like Compass relate to Sass?*

They are CSS preprocessors. They are an abstraction layer on top of CSS. They are a special syntax/language that compile down into CSS. They make managing CSS easier, with things like variables and mixins to handle vendor prefixes (among other things). They make doing best practices easier, like concatenating and compressing CSS.

↥ back to top

Q. *What is file splitting and why should you use it?*

File splitting helps organize your CSS into multiple files, decreasing page load time and making things easier to manage. How you decide to split them up is up to you, but it can be useful to separate files by component. For example, we can have all button styles in a file called `_buttons.scss` or all your header-specific styles in a file called `_header.scss`, main file, say _app.scss, and we can import those files by writing @import 'buttons';

↥ back to top

Q. *What is the @content directive used for?*

Q. *What is wrong with Sass nesting?*

Q. *What is variable interpolation in Sass?*

Q. *What is the difference between SCSS and Sass?*

Q. *What are the advantages/disadvantages of using CSS preprocessors?*

Q. *Explain what is the use of the @import function in Sass?*

Q. *Explain what is the use of Mixin function in Sass? What is the meaning of DRY-ing out a mixin?*

Q. *Explain what Sass Maps is and what is the use of Sass Maps?*

Q. *Explain how Sass comments are different from regular CSS?*

Q. *Does Sass support inline comments?*

Q. *Explain when can you use the %placeholders in Sass?*

Q. *Is it possible to nest variables within variables in Sass?*

Q. *What are Sass cons and pros?*

Q. *Explain how Mixins is useful?*

Q. *What are the similarities between LESS and Sass?*

Q. *Explain what is the use of &combinator ?*

Q. *What is the way to write a placeholder selector in Sass?*

Q. *What are number operations in Sass?*

Q. *Explain @if, @else, @for, @include, @at-root, @extend, @error, @debug directives?*

Q. *Which directive displays an error message in SASS?*

Q. *How many output styles are there in sass?*

Q. *Which symbol is used to refer parent selector in sass?*

↥ back to top