From CSS to SASS

CSS on Steroids!

Akshat Bhanchawat

Full Stack Web Developer &
Advisory Technical Head – SIAM VIT

Connect with me

/akshat112

/in/akshatbhanchawat

/akshatbhanchawat

1 What is SASS

2 CSS vs SAAS vs SCSS

3 SASS Features

4 Demo application

5 Wrap Up

What is SASS

SASS (Syntactically Awesome Stylesheet) is a CSS pre-processor, which helps to reduce repetition with CSS and saves time. It is more stable and powerful CSS extension language that describes the style of document structurally. Basically, It gives superpowers to your CSS.

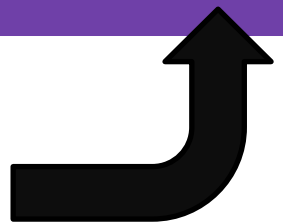
Or as they say

CSS on Steroids

CSS vs SASS vs SCSS

CSS	SASS	SCSS
CSS is ultimately what adds styling to a web page, and will continue to do	SASS basically is a more easy and programmatic way of writing CSS and the code written in SASS Script will ultimately converted to CSS	SCSS utilises the superpowers of SASS and allows us to write code in conventional CSS way. So we get the best of both worlds!

Most used and now the default style



Main features of SASS

1. Variables
2. Nesting CSS
3. Maps
4. Functions
5. Mixins
6. Loops
7. Inheritance
8. Partials

Let’s have a detailed look at them →

Variables

Simply define variables using '$' and the variable name

```
// FROM CSS to SASS

$primaryColor: #673ab7;
$secondaryColor: #7e57c2;

body {
  background-color: $primaryColor;
}
```

Nesting

Putting selectors inside parent selectors makes it easy to manage the code

```scss
// FROM CSS to SASS

$primary: blue;
$secondary: lightblue;

body{
  p{
    color: $primary;
  }

  #lightText{
    color: $secondary;
  }
}
```

Compiled to

```css
// FROM CSS to SASS


body p {
  color: blue;
}
body #lightText {
  color: lightblue;
}
```

Maps

Our conventional key-value pairs, now possible with SASS

```
// FROM CSS to SASS

$weights : (
    "regular": 100,
    "medium": 500,
    "bold": 700
);

#myBoldText{
  font-weight: map-get($weights, bold);
}
```

Functions

Yes! You can now create your own functions that returns a value for further use

```
// FROM CSS to SASS

//In case of maps, we need to use map-get($weights, type)
//everytime we want to access the property

@function weights($weight-name){
  @return map-get($weights, $weight-name);
}

#myText{
  font-weight: weights(bold);
}
```

Mixins

Typing same code again and again? Use mixins!

```
// FROM CSS to SASS

@mixin myStyles {
  color: $primary;
  font-weight: weights(bold);
  text-transform: uppercase;
  font-family: "Open Sans";
  @content; // Anything else added by user at later stage
}

#textOne{
  @include myStyles;
}
#textTwo{
  @include myStyles{
    color: $secondary; // Adding this is possible because of @content decorator
  }
}
```

Loops

Mostly used when we want to set variations of a property to multiple children elements respectively

```
// FROM CSS to SASS

// We have following HTML code
//      <ul>
//          <li> Number 1 </li>
//          <li> Number 2 </li>
//          <li> Number 3 </li>
//      </ul>

@for $i from 1 through 3 {
  li:nth-child(#{i}) {
    background-color: lighten(red, $i * 5%);
  }
}
```

Inheritance

Inherit all the properties from other classes using @extend

```
// FROM CSS to SASS

.error {
  border: 1px #f00;
  background-color: #fdd;

  &--serious {
    @extend .error;
    border-width: 3px;
  }
}
```

Partials

Partials helps you manage component specific and utility CSS in separate files and combine them later into one single file.

We will see how to use partials in demo.

Time to get hands dirty!

Let’s look at the landing page for some hypothetical company I built and see the use of the concepts that we learnt till now.

Download code: https://bit.ly/fromcss2sass

Thanks!!

 /akshat112

 /in/akshatbhanchawat

 /akshatbhanchawat

Also, feel free to drop me a mail at akshat112@gmail.com in case of any doubt! ☺