Variables & Interpolation:

Sass allows to declare variables that can be used throughout the stylesheet. Variables begin with symbol **$**, and are declared just like properties. They can have any value that is allowed for a CSS property, such as colors, numbers (with units), or text.
Variables can be used for more than just property values. Indeed is possible to use ***#{$var}*** to insert them into property names or selectors.

SASS

```
$defaultLinkColor: #46EAC2;

a {
  color: $defaultLinkColor;
}

$wk: -webkit-;

.rounded-box {
  #{$wk}border-radius: 4px;
}
```

CSS

```
a {
  color: #46EAC2;
}

.rounded-box {
  -webkit-border-radius: 4px;
}
```

Nesting:

Sass allows to avoid repetition of CSS selectors by nesting the child selectors within the parent selector. It is possible to nest also pseudoclass selectors. The special character ***&*** references the parent selector.

names or selectors.

SASS

```
ul {
  list-style-type: none;
  li {
  border: {
    style: solid;
    left: {
      width: 1px;
      color: #999999;
    }
  }
  display: inline-block;
  margin: 0;
  padding: 0 5px;
    a {
      text-decoration: none;
      &:hover { text-decoration: underlin
    }
  }
}
```

CSS

```
ul {
  list-style-type: none;
}
ul li {
  border-style: solid;
  border-left-width: 1px;
  border-left-color: #999999;
  display: inline-block;
  margin: 0;
  padding: 0 5px;
}
ul li a {
  text-decoration: none;
}
ul li a:hover {
  text-decoration: underline;
}
```

Operations and Functions:

In a SASS stylesheet is possible to use the classic arithmetic operations; moreover the SASS engine makes available a large set of new useful functions. The entire list of these functions could be found @ http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html.

SASS

```
$defaultWindowSize: 960px;

#nav.side {
  float: right;
  width: $defaultWindowSize / 3;
  a {
    color: lighten($defaultLinkColor, 10%
  }
}
```

CSS

```
#nav.side {
  float: right;
  width: 320px;
}
#nav.side a {
  color: #74EFD1;
}
```

Mixins:

Mixins allow re-use of styles without having to copy and paste them or move them into a non-semantic class.
To define a mixin is used the ***@mixin*** directive, which takes a block of styles that can then be included in another selector using the ***@include*** directive.

SASS

```
@mixin default-box {
  $borderColor: #666;
  border: 1px solid $borderColor;
  clear: both;
  display: block;
  margin: 5px 0;
  padding: 5px 10px;
}

footer, header{ @include default-box; }
```

CSS

```
footer, header {
  border: 1px solid #666;
  clear: both;
  display: block;
  margin: 5px 0;
  padding: 5px 10px;
}
```

Arguments:

The real power of mixins comes when you pass them arguments.
Arguments are declared as a parenthesized, comma-separated list of variables. Each of those variables is assigned a value each time the mixin is used.

SASS

```
@mixin default-box($color, $boxModel, $pa
  $borderColor: $color;
  border: 1px solid $borderColor;
  clear: both;
  display: $boxModel;
  margin: 5px 0;
  padding: 5px $padding;
}

header{ @include default-box(#666, block,
footer{ @include default-box(#999, inline
```

CSS

```
header {
  border: 1px solid #666;
  clear: both;
  display: block;
  margin: 5px 0;
  padding: 5px 10px;
}

footer {
  border: 1px solid #999;
  clear: both;
  display: inline-block;
  margin: 5px 0;
  padding: 5px 5px;
}
```

Selector Inheritance:

The SASS ***@extend*** directive makes possible for a selector to inherit all the styles of another selector without duplicating the CSS properties.


```
.error {
  border: 1px #F00;
  background: #FDD;
}

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

```
.error, .badError {
  border: 1px #F00;
  background: #FDD;
}

.badError {
  border-width: 3px;
}
```

Import stylesheet

SASS ***@import*** directive allows to break a stylesheet up into multiple files. Any style rules, variables or mixins defined in @imported files are available to the files that import them.
Usually, as naming convention, the name of the files meant to be imported, begins with an underscore. In order to support both .scss and .sass files, SASS allows files to be imported without specifying a file extension.

SASS

```
@import 'partials/_vars';

body {
  color: $color;
}

/* content of _vars.scss */
$color: #333;
```

CSS

```
body {
  color: #333;
}
```