Telerik Academy

SASS

Syntactically Awesome Stylesheets

CSS Styling

Telerik Software Academy

telerikacademy.com

Telerik Academy

Table of Contents

Telerik Academy

SASS Overview

What is SASS?

Telerik Academy

SASS Overview

- Syntactically Awesome Stylesheets is an extension of CSS
  - Makes coding CSS much easier and organized
  - Translates to pure CSS on the server
    - i.e. no slowdown on the client
- SASS introduces many techniques to power the CSS coding
  - Variables, Functions, Mixins, Inheritance, Operators, etc

Telerik Academy

Working with SASS

Ok... but How?

Telerik Academy

Compiling SASS Using Node.js

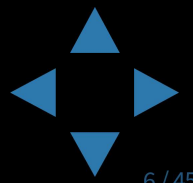
- Compiling SASS with Node.js
  - Usable with any text editor or IDE
  - Install Node.js
  - Install `node-sass` package

```
npm install -g node-sass
```

  - Compile `scss` files on changes:

```
node-sass --watch source.sccs dest.css
```

  - You get the translated CSS



Telerik Academy

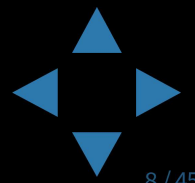
Compiling SASS with Node.js

Demo

Telerik Academy

Coding SASS in Visual Studio

- VS has many plugins to support SASS
  - Some of them are:
    - Web Compiler & Web Essentials
    - SassyStudio
  - Plugins extend core Visual Studio fetures with:
    - Syntax highlighting
    - Intellisense
    - Snippets



Telerik Academy

SASS in Visual Studio

Demo

Telerik Academy

SASS Features

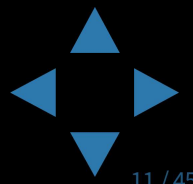
Selector Nesting, Mixins, Variables, Operators etc…

Telerik Academy

Selector Nesting

- SASS features selector nesting

```
body {
    font: normal 16px arial;
    color: #fff;
    background-color: #011b63;

    h1 {
        font-size: 2.3em;
        font-weight: bold;
    }
}
```

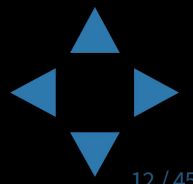


Telerik Academy

Selector Nesting

- The resulting CSS

```
body {
  font: normal 16px arial;
  color: #fff;
  background-color: #011b63;
}

body h1 {
  font-size: 2.3em;
  font-weight: bold;
}
```



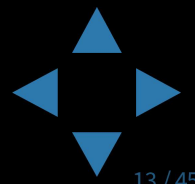
Telerik Academy

Selector Nesting

- Nested selectors in SASS are compile to nested selectors in CSS

```
body {
  font-size: 25px;

  h1 {
    font-style: italic;
  }
}
```

```
body {
  font-size: 25px;
}

body h1{
  font-style: italic;
}
```



Telerik Academy

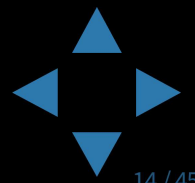
Selector Nesting

- Selectors can also reference themselves inside their selector using the symbol &
- The following SASS code:

```
a {
    color: black;

    &:hover {
        color: lightblue;
    }
}
```

- compiles to the following css code:

```
a {
  color: black;
}

a:hover {
  color: lightblue;
}
```



Telerik Academy

Selector Nesting

Demo

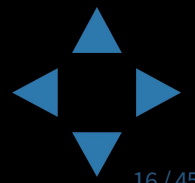
Telerik Academy

SASS Variables

- SASS also has variables
  - Using the $ (dolar) symbol
  - Can be used to store colors, size, etc…
- Usable to set default background-color, font-color, font-size, etc…

```
$link-color: #ffffff;
$v-link-color: #646363;
a {
  color: $link-color;
  &:visited {
    color: $v-link-color;
}
```

```
body a {
  color: white; }
  body a:visited {
    color: #646363; }
```



Telerik Academy

Variables

Demo

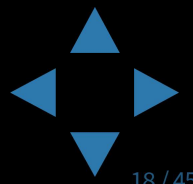
Telerik Academy

Interpolation

- SASS variables can be inserted as CSS properties
  - Using #{ }

```
$border-side:top;
$border-color:blue;
$border-style:ridge;
$border-width:15px;

border-#{$border-side} :
  $border-width $border-style $border-color;
```

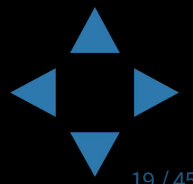
```
border-top : 15px ridge blue
```



Telerik Academy

Interpolation

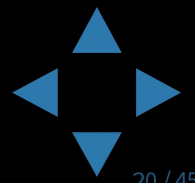
Demo



Telerik Academy

Mixins

- Mixins are kind of developer defined functions
  - The developer can make them for clear and reusable SASS
- Two kind of mixins
  - Parameterless
    - Get a default styles every time
  - With parameters
    - Get style based on some parameters
    - Gradient, borders, etc…



Telerik Academy

Defining Mixins

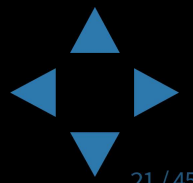
- How to define mixins?
  - Use the following syntax:

```
@mixin <mixin-name> {
/* SASS styles go here */
}
```

  - Then the styles are normal SASS
  - How to use the mixin?
    - Just write `@include <mixin-name>;`

```
@mixin clearfix{
  zoom:1;
  &:after{ display:block; content:""; height:0; clear:both; }
}
```

```
ul#main-nav{
  @include clearfix;
}
```



Telerik Academy

Basic Mixins

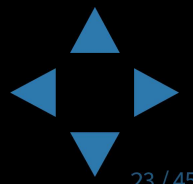
Demo

Telerik Academy

Mixins with Arguments

- Mixins can also be defined with parameters
  - i.e. for gradient-background

```
@mixin opacity($value){
  opacity: $value;
  filter: alpha(opacity = ($value*100));
  zoom: 1;
}
//arguments can take default values
@mixin box($border: none, $bg: rgba(0,0,0,0.7), $size: 200px) {
  width: $size; height: $size;
  border: $border;
  background: $bg;
  padding: 15px;
}
```



Telerik Academy

Mixins with Arguments

Demo

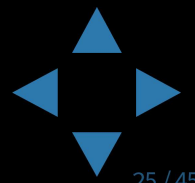
Telerik Academy

Selector Inheritance

- SASS enables the inheritance of selector
  - i.e. in a selector, get the properties of another selector
  - Use `@extend`

```
.clearfix {
  zoom: 1;
  &:after {
    display: block; height: 0;
    content: ""; clear: both;
  }
}
div{
  @extend .clearfix;
}
```

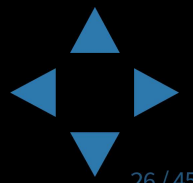
```
.clearfix, body div {
  zoom: 1;
}
.clearfix:after, body div:after {
  display: block;
  height: 0;
  content: "";
  clear: both;
}
```



Telerik Academy

Selector Inheritance

Demo



Telerik Academy

Operators

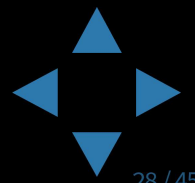
Performing calculations with SASS

Telerik Academy

SASS Operators

- Described with examples here

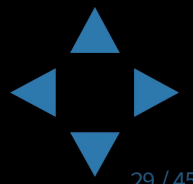
Operators	Role
:	assignment
and, or, not	logical
>, >=, <, <=, ==, !=	comparison
+, -, /, *, %	arithmetical



Telerik Academy

SASS Operators

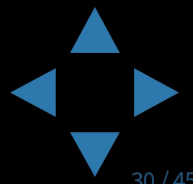
Demo



Telerik Academy

Conditional statements

Generate style rules depending on conditions

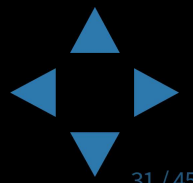


Telerik Academy

Conditional statements

- Using `@if`, `@else` and `@else if` directives:

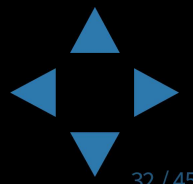
```
@mixin opacity($value) {

    opacity: $value;

    @if(0.75 < $value and $value <= 1) {
      color: blue;
    } @else if(0.25 < $value and $value <= 0.75) {
      color: orange;
    } @else {
      color: black;
    }
}
```



Telerik Academy

Conditional statements

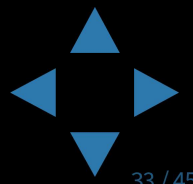
Demo



Telerik Academy

Loops

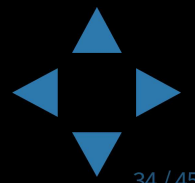
Repeating logic and generating CSS



Telerik Academy

Loops

- SASS provides the following loop directives:
  - `@for` loop
  - `@while` loop
  - `@each` loop
- Similar to loops in C#, JS, C++ and so on
- Can be used to repeat actions and/or generate css rules



@for loop

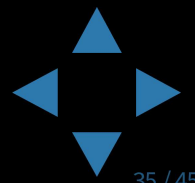
- Syntax:

```
@for <index-name> <start> through <end> { <loop-body> }
```

  - Repeats `<loop-body>` for each value from `start` to `end`
  - The index is accessible through the variable with `<index-name>`

```
@for $index from 1 through 3 {
    .margin-left-#{$index} {
        margin-left: $index * 10%
    }
}
```

```
/* Resulting CSS */
.margin-left-1 { margin-left: 10%; }

.margin-left-2 { margin-left: 20%; }

.margin-left-3 { margin-left: 30%; }
```



@while loop

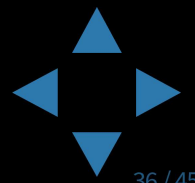
- Syntax:

```
@while <condition> { <loop-body> }
```

  - Repeats <loop-body> while the <condition> evaluates to true

```
$i: 0;
@while $i < 6 {
    .width-#{$i} {
        width: $i * 20%
    }
    $i: $i + 2
}
```

```
/* Resulting CSS */
.width-0 { width: 0%; }

.width-2 { width: 40%; }

.width-4 { width: 80%; }
```



Telerik Academy

@each loop and lists in SASS

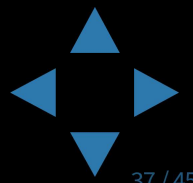
- SASS supports declaration of lists of values, as well as a way of iterating over those values with @each
- Syntax of list declaration:

```
$list-name: <value1> <value2> <value3> ... ;
```

- Declaring a list:

```
$selectors: .link .clickable .visited;
```

- SASS Lists in more details here



Telerik Academy

@each loop and lists in SASS

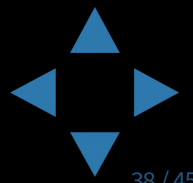
- Iterating over a list with @each is done using the following syntax:

```
@each <variable-name> in <list-name> { <loop-body> }
```

- Very similar behavior to foreach in C#

```
.half-width { width: 50% }
$selectors: div p section article;

@each $s in $selectors {
    #{$s} {
        @extend .half-width
    }
}
```

```
/* Resulting CSS */
.half-width, div, p, section, article { width: 50%; }
```



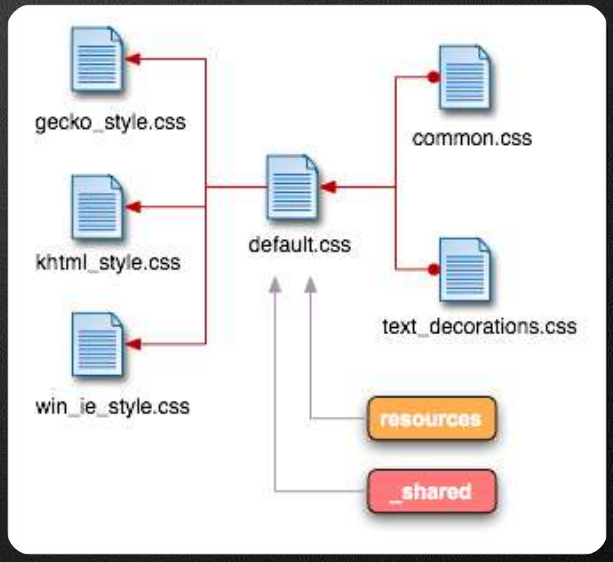
Telerik Academy

SASS Loops

Demo

Telerik Academy

Importing SASS



Telerik Academy

Importing SASS Files

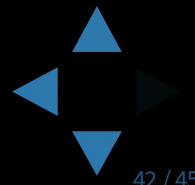
- SASS files can be imported in other SASS files
  - Like CSS files can be imported in CSS files
  - Use the `@import` directive
- SASS defines partials
  - i.e. SASS files that are meant to be imported
  - The naming convention suggests that all partial names start with _ (underscore)

```
@import '_gradients.scss'
//can use the items from gradients.scss
```

Telerik Academy

Importing SASS

Demo



Telerik Academy

SASS Summary

- Preprocessor for CSS with a lot of features
  - Aims to make CSS coding more powerful and flexible
  - It's features include variables, nesting, inheritance, mixins, functions, flow-control and more
- Used on the server
- SASS can be compiled to CSS using Ruby, or by using a plugin for your favorite editor
- SASS allows importing of other files and declaration of partials