

Sass



CSS With Super Powers



O que é SCSS?
É um pre-processador de CSS.



Como funciona?

`stylesheet.css`



Como funciona?

~~stylesheet.css~~

stylesheet.scss



Como funciona?

main

stylesheet.scss



Como funciona?

main

stylesheet.scss



processamento e compilação



main

stylesheet.scss



processamento e compilação

output

stylesheet.css



Como funciona?





Como se escreve SCSS?

Da mesma maneira que se escreve CSS.
Mas com ligeiras diferenças e com algumas novas ferramentas.



Exemplo de SCSS

```
@mixin button-base() {  
  @include typography(button);  
  @include ripple-surface;  
  @include ripple-radius-bounded;  
  
  display: inline-flex;  
  position: relative;  
  height: $button-height;  
  border: none;  
  vertical-align: middle;  
  
  &:hover { cursor: pointer; }  
  
  &:disabled {  
    color: $mdc-button-disabled-ink-color;  
    cursor: default;  
    pointer-events: none;  
  }  
}
```



Estrutura do ficheiro.

SCSS é baseado em statements.

Declarações feitas de forma utilizar certas ferramentas.

Universal Statements

\$var - variável. key : value

@if e @each - Fluxos de Condição.

@warn, @error, @debug - at-rules

CSS Statements

h1 {...} - regras normais de CSS

@media, @font-face - @-rules

Top Level Statements

@use - Loading de Módulos

@imports - importar ficheiros

@mixin - receitas mágicas

@functions - Blocos de execução.



Estrutura de Dados.

Podemos usar várias formas de expressão quando escrevemos SCSS.

Expressões Literais

Numbers

Strings

Cores

Booleans, etc...

Operações

== ou !=

, - , * , /

< , > , >= , <= , etc...

E outros valores e expressões



Variáveis.

SCSS

```
$base-color: #c6538c;

.alert {
  border: 1px solid $base-color;
}
```

output

CSS

```
.alert {
  border: 1px solid #c6538c;
}
```



Variáveis.

SCSS

```
$base-color: #c6538c;  
$border-dark: rgba($base-color, 0.88);
```

```
.alert {  
  border: 1px solid $border-dark;  
}
```

output

CSS

```
.alert {  
  border: 1px solid rgba(198, 83, 140, 0.88);  
}
```




Scope das variáveis.

As variáveis declaradas à cabeça (top level) são globais.

As variáveis declaradas dentro do bloco são locais.

Os blocos podem aceder as variáveis globais e locais.

SCSS

```
$base-color: #c6538c;

.alert {
  $base-color: #cccccc;
  border: 1px solid $border-dark;
}
```

output

CSS

```
.alert {
  border: 1px solid #cccccc;
}
```



Nesting

SCSS

output

CSS

```
.container span {  
  color: white;  
}
```



Nesting

SCSS

```
.container {  
  span {  
    color: white;  
  }  
}
```

output

CSS

```
.container span {  
  color: white;  
}
```



Selectores Ascendentes

```
<div class="box">  
  <div class="box__container"></div>  
</div>
```

SCSS

CSS

```
.box {  
  display: flex;  
}  
  
.box__container {  
  width: 100px;  
  height: 100px;  
  border-radius: 50px;  
}
```

output



Selectores Ascendentes

```
<div class="box">  
  <div class="box__container"></div>  
</div>
```

Podemos utilizar propriedades dos selectores dos pais através do caracter &.

SCSS

```
.box {  
  display: flex;  
  &-fluid {  
    $size: 100px;  
    width: $size;  
    height: $size;  
    border-radius: $size * 0.5;  
  }  
}
```

output

CSS

```
.box {  
  display: flex;  
}  
  
.box__container {  
  width: 100px;  
  height: 100px;  
  border-radius: 50px;  
}
```



Interpolação



É usada para a atribuição de valores dinâmicos .

SCSS

```
@mixin corner-icon($name, $top-or-bottom, $left-or-right) {
  .icon-#{ $name } {
    background-image: url("/icons/#{ $name }.svg");
    position: absolute;
    #{ $top-or-bottom }: 0;
    #{ $left-or-right }: 0;
  }
}

@include corner-icon("mail", top, left);
```

output



Interpolação



É usada para a atribuição de valores dinâmicos .

SCSS

```
@mixin corner-icon($name, $top-or-bottom, $left-or-right) {
  .icon-#{ $name } {
    background-image: url("/icons/#{ $name }.svg");
    position: absolute;
    #{ $top-or-bottom }: 0;
    #{ $left-or-right }: 0;
  }
}

@include corner-icon("mail", top, left);
```

output

CSS

```
.icon-mail {
  background-image: url("/icons/mail.svg");
  position: absolute;
  top: 0;
  left: 0;
}
```



Top Level Statements

@rules

Existem várias @rules que servem vários propósitos.



Top Level Statements

@rules

Existem várias @rules que servem vários propósitos.

@use - Injeta mixins, funções, variáveis de outras stylesheets.

@import - Importa funções, mixins e outras variáveis.

@extend - Autoriza os selectores a herdar propriedades de outros elementos.

@mixin e **@include** - Reutilizam pedaços de códigos.

Outros são:

@at-root, **@error**, **@warn**, **@debug**

Fluxos de Controlo são:

@if, **@each**, **@for** e **@while**



@mixin

É um grupo de declarações que podem ser reutilizadas.

SCSS

```
@mixin flex($direction, $wrap-nowrap) {  
  display: flex;  
  flex-direction: $direction;  
  flex-wrap: $wrap-nowrap;  
}
```

```
.selector {  
  @include flex(row, wrap);  
}
```

output



@mixin

É um grupo de declarações que podem ser reutilizadas.

SCSS

```
@mixin flex($direction, $wrap-nowrap) {  
  display: flex;  
  flex-direction: $direction;  
  flex-wrap: $wrap-nowrap;  
}
```

```
.selector {  
  @include flex(row, wrap);  
}
```

output

CSS

```
.selector {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
}
```



@if

Processa valores de acordo com certas condições.

SCSS

```
@mixin flex($direction, $wrap-nowrap: wrap) {  
  display: flex;  
  flex-direction: $direction;  
  flex-wrap: $wrap-nowrap;  
  @if $direction == row {  
    align-items: flex-start;  
  }  
}  
  
.selector {  
  @include flex(row);  
}
```

output



@if

Processa valores de acordo com certas condições.

SCSS

```
@mixin flex($direction, $wrap-nowrap: wrap) {  
  display: flex;  
  flex-direction: $direction;  
  flex-wrap: $wrap-nowrap;  
  @if $direction == row {  
    align-items: flex-start;  
  }  
}  
  
.selector {  
  @include flex(row);  
}
```

output

CSS

```
.selector {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
  align-items: flex-start;  
}
```



@if @else

Processa valores de acordo com certas condições.

SCSS

```
@mixin flex($direction, $wrap-nowrap: wrap) {  
  display: flex;  
  flex-direction: $direction;  
  flex-wrap: $wrap-nowrap;  
  @if $direction == row {  
    align-items: flex-start;  
  } @else {  
    align-items: center;  
  }  
}  
  
.selector {  
  @include flex(row);  
}
```

output

CSS

```
.selector {  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
  align-items: center;  
}
```



@each

Itera sobre os valores definidos em \$sizes e processa os seus valores.

SCSS

\$sizes: 40px,50px

```
@each $size in $sizes {  
  .icon-#{ $size } {  
    font-size: $size  
  }  
}
```

output



@each

Itera sobre os valores definidos em \$sizes e processa os seus valores.

SCSS

```
$sizes: 40px,50px
```

```
@each $size in $sizes {  
  .icon-#{ $size } {  
    font-size: $size  
  }  
}
```

output

CSS

```
.icon-40px {  
  font-size: 40px;  
}
```

```
.icon-50px {  
  font-size: 40px;  
}
```




@for

Similar ao @each mas neste caso \$i é um valor <number> iterado.

SCSS

```
@for $i from 1 through 4 {  
  .col-#{ $i } {  
    width: calc(100% / 12 * $i);  
  }  
}
```

output



@for

Similar ao @each mas neste caso \$i é um valor <number> iterado.

SCSS

```
@for $i from 1 through 4 {  
  .col-#{ $i } {  
    width: calc(100% / 12 * $i);  
  }  
}
```

output

CSS

```
.col-1 {  
  width: calc(100% / 12 * 1);  
}  
  
.col-2 {  
  width: calc(100% / 12 * 2);  
}  
  
.col-3 {  
  width: calc(100% / 12 * 3);  
}  
  
.col-4 {  
  width: calc(100% / 12 * 4);  
}
```



Diferença entre

SCSS SCSS

Sassy CSS Syntatically Awesome Style Sheet

Similar ao CSS - §Utiliza {} e ; Utiliza indentação em vez de {} e ;

Qualquer CSS é válido em SCSS “Não aceita” CSS

Extensão .scss Extensão .sass