06

Explicação - Funções no Sass

Funções no Sass

Se aplicarmos a variável `$tamanho-da-fonte-padrao` em todos os lugares em que aparece `width`, veremos que a estrutura se repetirá muitas vezes. Para lembrar a estrutura, observe esse trecho do `planos.scss`:

```
.plano {
  background: white;
  width: 18 * $tamanho-da-fonte-padrao;
  display: inline-block;
  margin: 1em 0 0 1.4em;
  padding-bottom: 2em;
}
```

Quando isso ocorre, sabemos que é vantajoso criar um *mixin*. O arquivo `mixins.scss` está assim:

```
@mixin borda-arredondada($raio: 10px) {
  -webkit-border-radius: $raio;
  border-radius: $raio;
}

%image-replacement {
  text-indent: -9999px;
  overflow: hidden;
  background-repeat: no-repeat;
}

%sombra-padrao {
  -webkit-box-shadow: 0 2px 6.65px 0.35 px rgba (0, 0, 0, 3);
  box-shadow: 0 2px 6.65px 0.35 px rgba (0, 0, 0, 3);
}
```

Abaixo de tudo isso, adicionaremos um *mixin* para a largura (`width`) chamado `retorna-largura`:

```
@mixin borda-arredondada($raio: 10px) {
  -webkit-border-radius: $raio;
  border-radius: $raio;
}

%image-replacement {
  text-indent: -9999px;
  overflow: hidden;
  background-repeat: no-repeat;
}

%sombra-padrao {
  -webkit-box-shadow: 0 2px 6.65px 0.35 px rgba (0, 0, 0, 3);
  box-shadow: 0 2px 6.65px 0.35 px rgba (0, 0, 0, 3);
}
```

```
@mixin retorna-largura {
  width: 18 * $tamanho-da-fonte-padrao;
}
```

Será que quando usarmos o *mixin* o valor de `18` sempre será conveniente? Provavelmente não. Então é interessante deixarmos esse valor variável também. Para isso, o transformaremos em um parâmetro chamado `$multiplicador` e o *mixin* ficará assim:

```
@mixin retorna-largura ($multiplicador){
  width: $multiplicador * $tamanho-da-fonte-padrao;
}
```

Criado o *mixin*, podemos chamá-lo no `plano` usando o `@include` :

```
.plano {
  background: white;
  @include retorna-largura;
  display: inline-block;
  margin: 1em 0 0 1.4em;
  padding-bottom: 2em;
}
```

Ao salvar o arquivo e conferir o compilado, ele dá erro. O Sass percebeu que falta o argumento do *mixin* que acabamos de inserir. Precisamos inserir o `$multiplicador` , que, nesse caso, possue valor `18` :

```
.plano {
  background: white;
  @include retorna-largura(18);
  display: inline-block;
  margin: 1em 0 0 1.4em;
  padding-bottom: 2em;
}
```

O outro lugar em que usamos a variável `$tamanho-da-fonte-padrao` foi o footer:

```
footer .container {
  padding-top: 3 * $tamanho-da-fonte-padrao;
  height: 6.5 * $tamanho-da-fonte-padrao;
  position: relative;
}
```

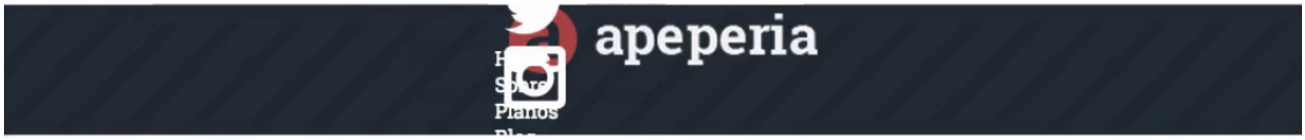
Aplicando o *mixin*:

```
footer .container {
  @include retorna-largura(3);
  height: 6.5 * $tamanho-da-fonte-padrao;
  position: relative;
}
```

O arquivo compilado ficará assim:

```
footer .container {
  width: 48px;
  height: 104px;
  position: relative;
}
```

Conferindo no navegador:



O rodapé ficou todo desconfigurado. Mas por que isso aconteceu? Analisando o código do arquivo compilado:

```
footer .container {
  width: 48px;
  height: 104px;
  position: relative;
}
```

O primeiro item não era `width` originalmente. Era um `padding-top` , mas o *mixin* que criamos só tem a função largura. Então desfaremos essa alteração. Seria interessante criarmos um *mixin* `retorna-padding-top` ? E quando aparecer outra situação na qual precisaremos dessas duas variáveis multiplicadas? Como o que nos interessa é a conta, podemos criar uma função. Então transformaremos o *mixin* em `function` , usando `@function` para defini-la como tal e o `@return` para a multiplicação:

```
@function retorna-largura ($multiplicador){
  @return $multiplicador * $tamanho-da-fonte-padrao;
}
```

Voltaremos ao `plano` para inserir essa nova função:

```
.plano {
  background: white;
  width: retorna-largura(18);
  display: inline-block;
  margin: 1em 0 0 1.4em;
  padding-bottom: 2em;
}
```

Indo ao arquivo `.css` , vemos que funcionou:

```
.plano {
  background: white;
  width: 288px;
  display: inline-block;
  margin: 1em 0 0 1.4em;
```

```
  padding-bottom: 2em;
}
```

Podemos então voltar ao `footer` e manter o `padding-top`:

```
footer .container {
  padding-top: retorna-largura(3);
  height: 6.5 * $tamanho-da-fonte-padrao;
  position: relative;
}
```

Como a função nem sempre retorna uma largura, seu nome não faz mais sentido. Vamos mudá-lo para `multiplica-pela-fonte`:

```
@function multiplica-pela-fonte ($multiplicador){
  @return $multiplicador * $tamanho-da-fonte-padrao;
}
```

Mas, se mudamos seu nome, precisamos chamá-la novamente no `plano`:

```
.plano {
  background: white;
  width: multiplica-pela-fonte(18);
  display: inline-block;
  margin: 1em 0 0 1.4em;
  padding-bottom: 2em;
}
```

E o mesmo para o `footer`, sem nos esquecermos de colocar o argumento para ser o multiplicador. Já podemos aplicar a função também no `height`.

```
footer .container {
  padding-top: multiplica-pela-fonte(3);
  height: multiplica-pela-fonte(6.5);
  position: relative;
```

Caso o arquivo compilado não apresente o valor esperado, mas o nome da função, confira se o arquivo `mixins.scss` foi salvo depois da mudança de nome.

As funções são uma ótima maneira de expandir o uso de contas (não apenas a multiplicação) dentro do Sass. Até a próxima!