02

Aninhamento

Transcrição

Olhando o `index` do `.scss`, vemos que existe um `menu-opcoes`, um nome que soa bem genérico. Como só temos um menu no site todo, podemos chamá-lo de `menu-principal`:

```
.menu-principal {
  position: absolute;
  right: 0;
  top: -.05em;
  font-size: 1.2em;
  font-weight: lighter;
}

menu-opcoes ul {
  padding-left: 0;
}

.menu-opcoes li {
  display: inline-block;
  width: 5em;
}

menu-opcoes a {
  color: white;
  text-decoration: none;
}

menu.opcoes a:hover {
  text-decoration: underline;
}
```

Ao salvar e atualizar no *browser*, vemos que algo deu errado e o menu está deslocado:

Isso aconteceu porque não renomeamos toda a classe. Substituindo todos os `menu-opcoes` por `menu-principal`, e verificando novamente o resultado no navegador:

Ainda há algo de errado. Renomeamos a classe, mas não a alteramos no *index*, que está assim:

```
<header>
  <div class="container">
    <hl>
      <img src="imagens/logo.png"alt=Logotipo Apeperia">
    </hl>
  <nav class="menu-opcoes">
    <ul>
  ...
  </ul>
  ...
  </header>
```

Ao trocarmos aqui também o `menu-opcoes` por `menu-principal`, essa questão se resolve.

Voltando ao código no arquivo `.scss`, podemos identificar que `menu-principal ul` é "filha" de `menu-principal`:

```
.menu-principal {
  position: absolute;
  right: 0;
  top: -.05em;
  font-size: 1.2em;
  font-weight: lighter;
}

.menu-princial ul {
}
```

No HTML, nós indicamos que um elemento é "filho" de outro quando ele está dentro deste. No Sass, nós temos um recurso interessante para indicar isso, que é o aninhamento (ou *nesting*, em inglês). Para aninhar o `ul` dentro de `menu-principal`, faremos assim:

```
.menu-principal {
  position: absolute;
```

```
    right: 0;
    top: -.05em;
    font-size: 1.2em;
    font-weight: lighter;

    ul {
      padding-left: 0;
    }
  }
```

É importante lembrar-se de deletar o `menu-princial ul` para que ele não fique repetido. Depois disso, o trecho correspondente no arquivo `.css` fica assim:

```
.menu-principal {
  position: absolute;
  right: 0;
  top: -.05em;
  font-size: 1.2em;
  font-weight: lighter;}
  .menu-princial ul {
    padding-left: 0;}
```

Podemos repetir o processo com o `li`, com o `a` e com o `a:hover`.

```
    .menu-principal {
      position: absolute;
      right: 0;
      top: -.05em;
      font-size: 1.2em;
      font-weight: lighter;

      ul {
        padding-left: 0;
      }

      li {
        displau: inline-block;
        width: 5em;
      }

      a {
        color: white;
        text-decoration: none;
      }

      a:hover {
        text-decoration: underline;
      }
    }
```

No arquivo `.css`, já indentado, o trecho ficará assim:

```
  .menu-principal {
    position: absolute;
    right: 0;
    top: -.05em;
    font-size: 1.2em;
    font-weight: lighter;}

    .menu-princial ul {
      padding-left: 0;}

    .menu-principal li {
      display: inline-block;
      width: 5em;}

    .menu-principal a {
      color: white;
      text-decoration: none;}

    .menu-principal a:hover {
    text-decoration: underline;}
```

E agora, caso queiramos mudar o nome de `menu-principal` para `menu-alternativo`, por exemplo, no arquivo `.scss`, todas as ocorrências serão automaticamente atualizadas no `.css`.

Olhando esse trecho de código com mais atenção, notamos que o `a:hover` está atrelado ao `a`. Sendo assim, podemos fazer um aninhamento:

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;

      a:hover {
      text-decoration: underline;
      }
  }
}
```

Quando vamos ao arquivo `.css`, ele está diferente do que esperávamos. Note que o "a" se repete na linha do "hover":

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;}

  .menu-principal a a:hover {
      text-decoration: underline;}

}
```

Precisamos elencar esses dois "a"s para que eles sejam sempre o mesmo, de maneira que, se eu precisar trocar o `a` por um `div`, ele seja substituído automaticamente no `a:hover` e vire `div:hover`. Para que isso aconteça, devemos usar um `&`.

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;

      &:hover {
      text-decoration: underline;
      }
  }
}
```

Depois que salvamos, o código no arquivo `.css` fica correto:

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;}

  .menu-principal a:hover {
      text-decoration: underline;}

}
```

E se trocarmos o `a` por `button` no arquivo `.scss`, a mudança é automaticamente replicada no `.css`:

```
.menu-principal {
  ...
  .menu-principal button {
    color: white;
    text-decoration: none;}

  .menu-principal button:hover {
      text-decoration: underline;}

}
```

Embora o alinhamento do Sass seja um recurso muito útil, ele deve ser usado com cautela quando for necessário. Por exemplo, se quisermos adicionar uma `div` dentro do `a`:

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;

      &:hover {
```

```
        text-decoration: underline;
        }

        div {
          color: red;
          p {
            font-size: 10px;
            span {
              font-weight: bold;
            }
          }
        }
    }
}
```

Parece tudo ok, mas quando vamos conferir no arquivo `.css` :

```
.menu-principal {
  ...
  .menu-principal a {
    color: white;
    text-decoration: none;}

  .menu-principal a:hover {
      text-decoration: underline;}

  .menu principal a div {
    color: red;}

  .menu-principal a div p{
    font-size: 10px;}

  .menu-principal a div p span {
    font-weight: bold;}
}
```

O seletor `a div p span` não é muito interessante, pois hierarquia demais em um seletor gera um problema de performance no CSS. É uma boa prática no CSS evitarmos isso, para facilitar o trabalho de leitura do *browser*. E as boas práticas do CSS se aplicam ao SASS, pois um SASS mal-feito gerará um CSS problemático.

Assim, embora o *nesting* seja uma ferramenta útil, deve ser usada com cautela.