UTFPR

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DIO CEZAR GO

SASS

AGENDA

- o que é?
- pré-processamento
- sass
- variáveis
- aninhamentos
- partials
- import
- mixins
- o que são prefixos e por que utilizar?
- extend
- operadores
- crie a sua stack
- exercícios
- materiais complementares

O QUE É?

- produtividade é a palavra chave;
- pergunta: se você é desenvolvedor web, quantas vezes se pegou copiando e colando códigos CSS com mais de 15 linhas que fazem a mesma coisa?
- pré-processadores CSS vem para suprir essa e outras necessidades;
- os *big players* são:
  - SASS (http://sass-lang.com);
  - LESS (http://lesscss.org);
- mas existem outros → https://goo.gl/HQKc1j

PRÉ-PROCESSAMENTO

- CSS em sí pode ser divertido;
- mas códigos grandes e complexos podem ser difíceis de manter;
- é neste momento que um pré-processador pode ajudar;
- permitem que você adicione funcionalidades que não existem no css:
  - variáveis → chega de dar replace de uma cor em todo o documento;
  - aninhamentos (nesting) → propõe uma nova maneira de organizar os seus elementos;
  - mixins → permite a criação de grupos de declarações CSS e sua reutilização, se comportam como funções;
  - partials e imports → inclusão de arquivos;
  - operadores → permite a utilização de operadores para cálculos de medidas;

PRÉ-PROCESSAMENTO

- um pré-processador lê o código que nele é escrito, e o compila em um arquivo CSS normal, que pode ser utilizado em seu site;
- existem várias formas de compilar seu arquivo;
- vimos a ferramenta gulp e suas duas formas de compilação: única e inspecionando o arquivos por modificações;

SASS

VARIÁVEIS

- as variáveis são uma forma de se armazenar informações para serem reutilizadas em seu CSS;
- você pode armazenar informações como cores, tipos de fontes, ou qualquer outro valor CSS que você precise reutilizar;
- SASS utiliza o símbolo $ para transformar algo em uma variável;

VARIÁVEIS

DECLARANDO VARIÁVEIS EM SASS

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

VARIÁVEIS

- quando o arquivos sass é processado, ele obtém os valores definidos nas variáveis e cria um arquivo css normal com os respectivos valores;
- a grande vantagem?
    - se seu projeto mudar a paleta de cores, ou o tamanho das fontes, ou a fonte primária... você só precisará substituir essa instrução na declaração da variável;

VARIÁVEIS

QUANDO COMPILA-SE PARA CSS

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

ANINHAMENTO (NESTING)

- quando se escreve em HTML você provavelmente está acostumado com uma estrutura aninhada;
- mas o CSS não permite isso;
- com SASS você poderá aninhar os seus elementos da mesma forma hierárquica que está definida em seu HTML;
- mas tenha cuidado → regras excessivamente aninhadas resultarão em um CSS de difícil manutenção, e isso é considerado má prática.

ANINHAMENTO (NESTING)

UTILIZANDO ANINHAMENTO

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

ANINHAMENTO (NESTING)

- note que os elementos *ul*, *li* e *a* estão aninhados dentro do seletor nav;
- essa é uma ótima forma de organizar o seu CSS tornando-o mais legível;
- seu código fica específico → a regra só será aplicada a um elemento que está dentro de outro;

ANINHAMENTO (NESTING)

QUANDO COMPILA-SE PARA CSS

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

nav li {
  display: inline-block;
}

nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

PARTIALS

- outra vantagem interessante do SASS é a possibilidade da criação de partials;
- sim, você já podia fazer isso com CSS *@import url("base.css");*
  - mas não são uma boa prática;
  - gera-se dependências, o que deixa sua velocidade de carregamento comprometida;
- a grande sacada do SASS → ele junta tudo em um único arquivo no momento da compilação!
- uma boa prática é a nomenclatura dos partials iniciando com underscore _, por exemplo: *_partial.scss;*
- dessa forma o Sass saberá que este arquivo é um arquivo parcial e não deverá ser compilado;

IMPORT

- é a diretiva para importação de um arquivo partial;
- vamos imaginar que temos os seguintes arquivos: _reset.scss e base.scss;
- e nós desejamos importar o arquivos _reset.scss no arquivo base.scss;

IMPORT

_RESET.SCSS

```
html,
body,
ul,
ol {
  margin:  0;
  padding: 0;
}
```

_BASE.SCSS

```
@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

não é necessário incluir a extensão .scss

IMPORT

QUANDO COMPILA-SE PARA CSS

```
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

MIXINS

- algumas coisas em css são tediosas de se escrever;
- especialmente em CSS3 que prevê uma série de prefixos específicos de cada navegador;
- com mixins permitem que você crie grupos de declarações CSS que você poderá reutilizar em todo seu site;
- e o mais bacana! você ainda pode passar valores para tornar seu mixin mais flexível;
- um bom exemplo para o uso de mixins é a definição dos prefixos de navegadores específicos;

O QUE SÃO PREFIXOS E PORQUE USAR?

- é uma forma de manter a compatibilidade da propriedade em vários navegadores;
- propriedades que ainda não estão totalmente especificadas;
- note que a última chamada é a “oficial” e “sobrescreve” todas as anteriores;

PSEUDO SELETORES

```
div {
   -webkit-border-radius: 10px;
   -moz-border-radius: 10px;
   -ms-border-radius: 10px;
   -o-border-radius: 10px;
   border-radius: 10px;
}
```

MIXINS

EXEMPLO DE UTILIZAÇÃO PARA BORDER-RADIUS

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

MIXINS

- para criar um mixin, você precisa utilizar a diretiva #mixin e dar um nome a ele;
- no exemplo, nomeamos o mixin de border-radius;
- também usamos a variável $radius entre parênteses, então podemos passar qualquer raio que desejemos;
- depois que um mixin é declarado, poderá ser utilizado em qualquer trecho do seu código sass começando com @include seguido do nome do mixin;

MIXINS

QUANDO COMPILA-SE PARA CSS

```
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```

EXTEND

- é uma das funções mais úteis do SASS;
- com a diretiva @extend você será capaz de compartilhar um conjunto de propriedades CSS de um seletor para outro;
- então, chega de ficar copiando e colando!

EXTEND

EXEMPLO DE EXTEND

```
.message {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  @extend .message;
  border-color: green;
}
```

CONTINUAÇÃO

```
.error {
  @extend .message;
  border-color: red;
}

.warning {
  @extend .message;
  border-color: yellow;
}
```

EXTEND

- neste exemplo nós pegamos todas as propriedades de *.message* e aplicamos a .success, .error e .warning;
- a mágica acontece quando o css é gerado e isso o ajuda a evitar a escrita de múltiplos nomes de classes no seu HTML;

EXTEND

QUANDO COMPILA-SE PARA CSS

```
.message, .success, .error, .warning {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

OPERADORES

- fazer operações matemáticas no seu CSS pode ser interessante;
- SASS permite que você faça operações com: +, -, *, / e %;
- mas isso também pode ser feito de forma nativa certo?
    - sim! em css puro: calc(100% - 10px);
    - a diferença é que o valor final será gerado em seu css e não processado pelo navegador;

OPERADORES

EXEMPLO DE UTILIZAÇÃO DE OPERADORES

```
.container { width: 100%; }


article[role="main"] {
  float: left;
  width: 600px / 960px * 100%;
}

aside[role="complementary"] {
  float: right;
  width: 300px / 960px * 100%;
}
```

neste exemplo, criamos um grid simples baseada em uma largura de 960px;

OPERADORES

QUANDO COMPILA-SE PARA CSS

```
.container {
  width: 100%;
}

article[role="main"] {
  float: left;
  width: 62.5%;
}

aside[role="complementary"] {
  float: right;
  width: 31.25%;
}
```

CRIE A SUA STACK

- sempre temos formas de organizar e separar os nossos arquivos css;
- eu criei a minha stack → https://github.com/diogocezar/dctb-sass

EXERCÍCIOS

- https://goo.gl/31WJVs
- resolver os exercícios 1, 2, 3 e 4;

MATERIAIS COMPLEMENTARES

- https://goo.gl/YbHD5D
- https://goo.gl/PmM1jS
- https://goo.gl/Aa6R8f
- https://goo.gl/Xu2vSj