CAPÍTULO 15

Apêndice - LESS

"Não podemos solucionar problemas usando a mesma forma de raciocínio que usamos quando os criamos."
— Albert Einstein

LESS é uma linguagem baseada em CSS (mesma ideia, sintaxe familiar) com recursos que fazem falta no CSS em algumas situações. É também chamado de pré-processador pois, na verdade, é usado para gerar um arquivo CSS no final.

Alguns dos recursos apresentados pela linguagem são variáveis, suporte a operações aritméticas, sintaxe mais compacta para representar hierarquias e mixins.

15.1 - VARIÁVEIS

Você já precisou usar a mesma cor no CSS em vários pontos diferentes? Um título e um botão com mesma cor, por exemplo? O CSS tem uma solução pra evitar copiar e colar, que seria o uso de classes. Mas, muitas vezes, usar essa mesma classe em tantas tags diferentes não é uma boa ideia.

Programadores estão acostumados com variáveis pra isso, mas o CSS não tem nada parecido. Mas o LESS sim!

```
@corprincipal: #BADA55;

#titulo {
   font-size: 2em;
   color: @corprincipal;
}
button {
   background-color: @corprincipal;
   color: white;
}
```

Repare no uso da `@corprincipal` que não é CSS puro, mas tem uma sintaxe bem parecida e familiar. Depois de compilado, o LESS vira esse CSS:

```
#titulo {
   font-size: 2em;
   color: #BADA55;
}
button {
   background-color: #BADA55;
   color: white;
}
```

15.2 – Contas

Sabe quando você tem aquele elemento principal com 960px mas que precisa de um padding de 35px e duas colunas lá dentro de tamanhos iguais mas deixando 20px entre elas? Qual o tamanho de cada coluna mesmo? 435px. Aí você coloca no CSS:

```
.container {
   padding: 35px;
   width: 960px;
}
.coluna {
   width: 435px;
}
```

E quando alguém mudar o tamanho do padding, você torce pra lembrarem de refazer a conta da coluna - que, aliás, seria (960px – 35px * 2 – 20px) / 2 = 435px. No LESS, você pode fazer a conta direito na propriedade e o resultado final é calculado:

```
.coluna {
   width: (960px - 35px * 2 - 20px) / 2;
}
```

Melhor ainda, junte com as variáveis que vimos antes e você nem copia e cola valores!

```
@total: 960px;
@respiro: 35px;
@espacamento: 20px;

.container {
   padding: @respiro;
   width: @total;
}
.coluna {
   width: (@total - @respiro * 2 - @espacamento) / 2;
```

```
}
```

E dá pra fazer contas de tudo que é tipo, até com cores!

15.3 - Hierarquia

Você tem um `#topo` com um título `h1` dentro e uma lista `ul` de links. E quer estilizar todos esses elementos. Algo assim:

```
#topo {
   width: 100%;
}
#topo h1 {
   font-size: 2em;
}
#topo ul {
   margin-left: 10px;
}
```

E se você precisar mudar o id do `#topo`? Ou trocá-lo por um `<header>` semântico? Tem que mexer em 3 lugares (e torcer pra ninguém ter usado em outro canto). Fora que o código fica desorganizado já que essas três regras não necessariamente precisam estar agrupadas no arquivo e podiam estar espalhadas por aí, apesar de serem todos sobre nosso cabeçalho.

No LESS, podemos escrever regras de maneira hierárquica, uma dentro da outra, e ele gera os seletores de parent. O mesmo CSS acima podia ser no LESS:

```
#topo {
   width: 100%;

   h1 {
```

```
        font-size: 2em;
    }
    ul {
        margin-left: 10px;
    }
}
```

Podemos usar vários níveis de hierarquia (mas não abuse!), deixando nosso código mais estruturado, flexível e legível.

15.4 – Funções de cores e palhetas automáticas

Provavelmente você já viu algum design que tem uma cor base principal e algumas cores secundárias combinando. Talvez uma versão mais light dessa cor base é usada como fundo e uma cor mais saturada no botão.

Você então pega o código de cada cor no Photoshop e coloca no CSS. E, se precisar mudar a cor, deve gerar as outras secundárias, certo? No LESS, você pode usar funções pra gerar essas cores:

```
@corbase: #BADA55;

body {
    background: lighten(@corbase, 20%);
}
h1 {
    color: @corbase;
}
button {
    background: saturate(@corbase, 10%);
}
```

Vai gerar cores 20% mais lights e 10% mais saturadas:

```
body {
    background:#dceca9;
}
h1 {
    color:#bada55;
}
button {
    background:#bfe44b;
}
```

Você ainda tem: `darken`, `desaturate`, `fadein`, `fadeout`, `spin`, `mix` e até funções matemáticas como `round`.

15.5 – Reaproveitamento com mixins

Uma das coisas mais legais do LESS é sua capacidade de criar as próprias funções, que ele chama de ***mixins***. É útil quando você tem que repetir a mesma coisa várias vezes, como nas propriedades CSS3 que precisam de prefixos, tipo uma borda redonda.

Você pode definir um mixin recebendo argumento o tamanho da borda e cuspindo as versões pros diversos navegadores:

```
.arredonda(@raio: 5px) {
  -webkit-border-radius: @raio;
     -moz-border-radius: @raio;
          border-radius: @raio;
}
```

Parece uma classe CSS mas ele recebe uma variável como parâmetro (que pode ter um valor default também). E você pode usar esse mixin facilmente:

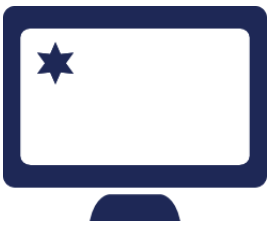
```
.painel {
   .arredonda(10px);
}
.container {
   .arredonda;
   width: 345px;
}
```

Isso gera o CSS:

```
.painel{
   -webkit-border-radius:10px;
      -moz-border-radius:10px;
           border-radius:10px;
}
.container{
   -webkit-border-radius:5px;
      -moz-border-radius:5px;
           border-radius:5px;

   width:345px;
}
```

As possibilidades são infinitas! Pense num mixin pra te ajudar a gerar aqueles gradientes CSS3 chatos ou um mixin próprio `.botaoBonito` que gera botões legais só recebendo uma cor base e um tamanho.

Agora é a melhor hora de aprender algo novo



Se você gosta de estudar essa apostila aberta da Caelum, certamente vai gostar dos novos **cursos online** que lançamos na plataforma **Alura**. Você estuda a qualquer momento com a **qualidade** Caelum.

Conheça a Alura.

15.6 - Executando o LESS

No site do LESS, você pode baixar a versão *standalone* dele. Você pode rodá-lo apenas incluindo um JavaScript na página que faz o parsing dos arquivos `.less` quando ela carrega.

```
<script src="less.js" type="text/javascript"></script>
```

Com isso, podemos incluir diretamente nosso arquivo `.less` usando uma tag `<link>`, colocada **antes** da tag que carrega o `less.js`:

```
<link rel="stylesheet/less" type="text/css" href="styles.less" />
```

Para melhor performance, o ideal seria gerar o CSS antes usando o compilador. Há uma versão em linha de comando usando Node.JS, mas como é JavaScript, você pode rodá-lo em qualquer canto - até no Java com Rhino.

Há também programas visuais pra instalar, como o LESS.app e dá pra testar código rapidamente online mesmo no LessTester.com.

No Windows, você pode usar o **WinLESS** que é um compilador com interface gráfica e bem fácil de usar: http://winless.org/

15.7 - Para saber mais: recursos avançados e alternativas

A linguagem LESS tem recursos ainda mais avançados. Dá pra fazer mixins mais complicados com uma espécie de if/else por exemplo e até usar pattern matching. Você encontra todos os detalhes na documentação oficial.

Além do LESS, existem outros pré-processadores CSS no mercado. O Sass (http://sass-lang.com/) é muito famoso no mundo Ruby e tem zilhões de funções, tornando-o mais poderoso que o LESS mas mais complexo também. Há também o Stylus (http://learnboost.github.io/stylus/), que simplifica ainda mais a sintaxe.

15.8 - Exercícios: LESS

1. Atualmente, nosso arquivo `estilos.css` possui várias regras que usam o seletor de hierarquia (espaço). Essas regras podem ser escritas de forma mais compacta com LESS. Então vamos usar LESS para escrevê-lo. Neste exercício, vamos usar a versão Javascript do LESS, que transforma nosso código em CSS dentro do navegador.

Crie uma nova pasta no seu projeto chamada `less` e copie o `estilos.css` para lá. Em seguida, renomeie-o para `estilos.less`. Por fim, altere o `index.html` para usar o `estilos.less`. Troque a linha

```
<link rel="stylesheet" href="css/estilos.css">
```

por

```
<link rel="stylesheet/less" href="less/estilos.less">
```

E inclua a seguinte linha antes de fechar a tag `body` para carregar o pré-processador LESS:

```
<script src="js/less.js"></script>
```

Teste a página no navegador. Ela deve continuar com a mesma aparência de antes.

2. Vamos começar a migrar nosso código CSS para LESS. Um seletor bastante repetido no código é `.painel`. Agrupe todas as regras que usam esse seletor num único seletor no LESS. O código final deve ficar parecido com este:

```
.painel {
    /* regras que estavam em .painel {...} */
    li {
        /* regras que estavam em .painel li {...} */

        &:hover {
            /* regras que estavam em .painel li:hover {...} */

            &:nth-child(2n) {
                /* regras que estavam em .painel li:nth-child(2n):hover
{...} */
            }
        }
    }

    h2 {
        /* regras que estavam em .painel h2 {...} */

        &:before {
```

```
            /* regras que estavam em .painel h2:before {...} */
        }
    }

    a {
        /* regras que estavam em .painel a {...} */
    }

    .mostra-mais {
        /* regras que estavam em .painel .mostra-mais {...} */
    }
}
```

Teste novamente no navegador. A página não deve mudar, mas veja que o código fica mais organizado e curto!

3. A cor `#333333` (cinza escuro) se repete algumas vezes no nosso estilo. Vamos isolá-la numa variável para facilitar a manutenção:

```
@escuro: #333333;
```

Procure os lugares que usam a cor e use a variável no lugar. Por exemplo:

```
body {
    color: @escuro;
    /* outras regras */
}
.mostra-mais {
    background: @escuro;
    /* outras regras */
}
```

Experimente trocar o valor da variável e veja o efeito: para mudar a cor de vários elementos da página agora basta mexer num único lugar!

4. Vamos deixar o nosso código de transições mais limpo isolando os prefixos num único lugar. Para isso, vamos criar um *mixin*:

```
.transicao(@propriedades, @tempo) {
    -webkit-transition: @propriedades @tempo;
    -moz-transition: @propriedades @tempo;
    -ms-transition: @propriedades @tempo;
    -o-transition: @propriedades @tempo;
    transition: @propriedades @tempo;
}
```

Agora altere o código que escala e rotaciona as fotos dos produtos quando o mouse passa em cima: apague as declarações `transition` e coloque no lugar

```
.transicao(all, 0.7s);
```

Faça o mesmo com os gradientes dos painéis:

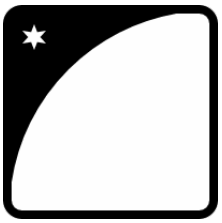
```
.gradiente(@cor1, @cor2) {
    background: @cor1; /* Navegadores antigos */
    background: -moz-linear-gradient(top,
                        @cor1 0%, @cor2 100%); /* FF3.6+ */
    background: -webkit-gradient(linear, left top, left bottom,
                        color-stop(0%, @cor1),
                        color-stop(100%, @cor2)); /* Chrome,Safari4+ */
    background: -webkit-linear-gradient(top,
                        @cor1 0%, @cor2 100%); /* Chrome10+,Safari5.1+ */
    background: -o-linear-gradient(top,
                        @cor1 0%, @cor2 100%); /* Opera 11.10+ */
    background: -ms-linear-gradient(top, @cor1 0%, @cor2 100%); /* IE10+ */
    background: linear-gradient(to bottom, @cor1 0%, @cor2 100%); /* W3C */
    filter: progid:DXImageTransform.Microsoft.gradient(
                    startColorstr='@cor1',
                    endColorstr='@cor2',
                    GradientType=0 ); /* IE6-9 */
}

.novidades {
    .gradiente(#f5dcdc, #f4bebe);
}

.mais-vendidos {
    .gradiente(#dcdcf5, #bebef4);
}
```

Dica: você pode até usar o gerador de gradientes do ColorZilla (http://www.colorzilla.com/gradient-editor/) para gerar o código desse mixin.

5. (opcional) Podemos melhorar nosso *mixin* de gradiente fazendo uma versão que só recebe uma cor e calcula a segunda automaticamente, fazendo ela ser 10% mais escura que a cor dada. Podemos ainda fazer com que essa nova versão já aproveite o *mixin* já existente, passando a segunda cor calculada para ele:

```
.gradiente-automatico(@cor1) {
    .gradiente(@cor1, darken(@cor1, 10%));
}
```

Faça o teste nos gradientes dos painéis: use essa versão do *mixin* e veja o efeito ser aplicado automaticamente.



A Caelum oferece o **curso WD-43** presencial nas cidades de São Paulo, Rio de Janeiro e Brasília, além de turmas incompany.

Consulte as vantagens do curso *Desenvolvimento Web com HTML, CSS e JavaScript*.

CAPÍTULO ANTERIOR:

Apêndice - Otimizações de front-end

PRÓXIMO CAPÍTULO:

Apêndice - Subindo sua aplicação no cloud

Você encontra a Caelum também em:

Blog Caelum

Cursos Online

Facebook

Newsletter

Casa do Código

Twitter