

## Vídeo 2

### Transcrição

Com uma análise mais atenta, perceberemos que estamos repetindo a cor de fundo.



Estamos usando um azul escuro. De volta ao editor de texto, criaremos a variável `@cor-auxiliar`.

```
@cor-padrao: #c24e4b;  
@cor-auxiliar: #1e2c35;  
@cor-branca: #fff;
```

Agora, aonde aparecia o hex `#1e2c35`, nós substituiremos por `@cor-auxiliar`. Começaremos por `.planos`.

```
/** Planos */  
.planos {  
  background: @cor-auxiliar;  
}  
.planos h2 {  
  color: @cor-branca;  
}
```

Depois, em `footer`.

```
footer:before {  
  content: "";  
  background: @cor-auxiliar;  
  top: 0;  
  left: 0;  
  bottom: 0;  
  right: 0;  
  position: absolute;  
  z-index: -1;  
}
```

Se testarmos no navegador, não veremos mudanças. Mas para testar se tudo saiu bem, teremos que trocar a cor da variável para vermelho ( #f00 ).

```
@cor-padrao: #c24e4b;  
@cor-auxiliar: #f00;  
@cor-branca: #fff;
```

Quando atualizarmos a página no navegador, veremos que a cor do fundo foi alterada.



Mas nós não alteramos a cor do header . Se tentássemos adicionar a cor, o código ficaria assim:

```
/** Header **/  
header {  
  border-top: 5px solid #c24e4b;  
  background: rgba(@cor-auxiliar, 0.8);  
  height: 90px;  
  width: 100%;  
  position: absolute;  
  
}
```

Mas se tentarmos compilar o código como ele está agora, teremos problemas:

```
aluras-Mac-mini:css alura$ lessc estilos.less estilos.css  
ArgumentError: error evaluating function `rgba`: color functions take numbers as parameters in /Users/alura/Documents/natan/less/Arquivos do curso/apeperia/css/estilos.less on line 30, column 15:  
29  border-top: 5px solid @cor-padrao;  
30  background: rgba(@cor-auxiliar, 0.8);  
31  height: 90px;  
  
aluras-Mac-mini:css alura$
```

Ele não reconheceu o código. Infelizmente, não poderemos adicionar a variável desta forma. Quando queremos passar uma opacidade para cor, precisaremos usar uma **função de cor**. O LESS possui diversas delas. Como queremos criar uma opacidade, usaremos a função `fade` , que usa como parâmetro a cor usada e o "quanto" ela será usada.

```
/** Header **/  
header {
```

```
border-top: 5px solid #c24e4b;  
background: fade(@cor-auxiliar, 0.8);  
height: 90px;  
width: 100%;  
position: absolute;  
  
}
```

Agora, se tentarmos compilar, não receberemos a mensagem de erro. Mas no navegador, veremos que a cor ficou excessivamente transparente, quase imperceptível.



Isto significa que não podemos seguir a mesma lógica para definir o valor no `rgba`. Se queremos **80%** de opacidade, adicionaremos o valor `80`.

```
/** Header */  
header {  
  border-top: 5px solid #c24e4b;  
  background: fade(@cor-auxiliar, 80);  
  height: 90px;  
  width: 100%;  
  position: absolute;  
  
}
```



Para conseguir resultados semelhantes ao `rgb` precisamos usar a função de cor `fade`.

Você pode conhecer outras funções de cores na [documentação \(http://lesscss.org/functions/#color-channel\)](http://lesscss.org/functions/#color-channel) do LESS.

Pensando na experiência do usuário, seria interessante visualizarmos um brilho no botão quando passássemos o mouse sobre ele. Para isto teremos que trabalhar a classe `hover`. Vamos adicionar dentro com o `destaque button`, com a partir da cor padrão:

```
.destaque button: hover {  
  background: @cor-padrao-mais-clara;  
}
```

Queremos que o `background` varie com uma cor baseada na padrão, usaremos então a `@cor-padrao-mais-clara`. Mas para que isto funcionasse, teríamos que criar a variável `@cor-padrao-mais-clara` e adicionar o hexadecimal correspondente. Para facilitar, usaremos uma função que já fará clareará a cor: `lighten`.

```
.destaque button: hover {  
  background: lighten(@cor-padrao, 10);
```

Especificamos que queremos que a cor seja **10%** mais clara.



Agora, vemos a variação da cor botão ao passarmos o cursor do mouse. Se testássemos brincar com o valor da porcentagem de clareamento da cor, por exemplo **1%**, quase não veríamos a diferença. Mas com **5%** acho que seria o valor ideal. Deixaremos assim:

```
.destaque button: hover {  
  background: lighten(@cor-padrao, 5);
```

Lembrando que agora, sempre que quisermos alterar a cor padrão, basta mudar na variável. Não precisaremos alterar duas variáveis. Vimos como trabalhar com a função de cores no Less.

Deixamos tudo funcionando como deveria.