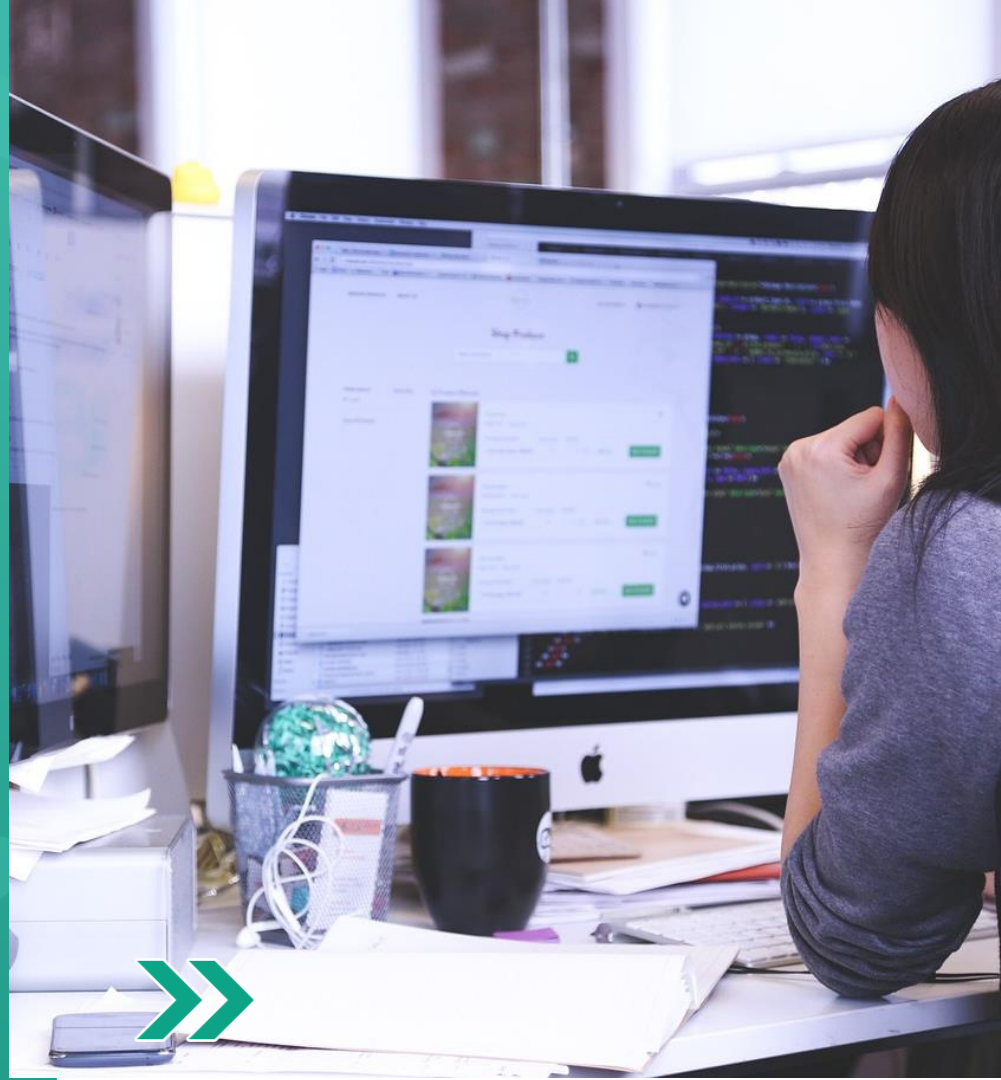
escola britânica de artes criativas & tecnologia

Profissão: Engenheiro Front-End

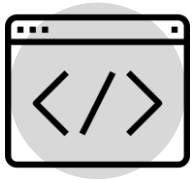
escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Introdução ao LESS

- Conheça o LESS
- Aplique variáveis no LESS
- Aplique o escaping
- Utilize mixins
- Crie mapas



Confira boas práticas da comunidade de Front-End por assunto relacionado às aulas.

Conheça o LESS

Comentários em Less:
Existem duas formas de fazer comentários:
1) Comentários de linha única:

```
// Este é um comentário de linha única em Less.
```

Os comentários de linha única começam com duas barras "//" e abrangem apenas uma linha. Eles são úteis para adicionar notas ou explicações rápidas ao código.

2) Comentários de bloco:

```
/*
  Este é um comentário de bloco em Less.
  Pode abranger várias linhas.
*/
```

Os comentários de bloco começam com "/*" e terminam com "*/". Eles podem ser usados para comentar várias linhas de código de uma só vez. Os comentários de bloco podem ser úteis para desativar temporariamente uma seção de código ou para adicionar documentação mais detalhada.

Aplique variáveis no LESS

- **Hierarquia de variáveis:**
  As variáveis podem ser declaradas em diferentes níveis de escopo. O Less segue a regra de escopo mais próximo para determinar o valor da variável. Por exemplo:

```
@primary-color: #007bff;

.button {
  @primary-color: #ff0000;
  color: @primary-color;  // Usará o valor #ff0000
}
```

- **Cálculos com variáveis:**
  O Less permite realizar cálculos com variáveis, tornando mais fácil manipular valores numéricos. Por exemplo:

```
@base-font-size: 16px;
@large-font-size: @base-font-size * 1.2;
```

Aplique variáveis no LESS

Escopo de variáveis:
Se você deseja limitar o escopo de uma variável a um bloco específico, pode usar chaves para criar um escopo local. Isso evita que a variável vaze para fora do bloco.
Por exemplo:

```
.container {
  @primary-color: #007bff;

  .button {
    color: @primary-color;
  }
}
```

Aplique o escaping

Acompanhe alguns exemplos de uso do escaping no Less.

- **Escapando um valor:**

```
.element {
  width: ~"200px";
}
```

Nesse exemplo, o valor 200px é escapado para ser tratado como um valor literal de CSS, sem qualquer interpretação do Less.

- **Escapando uma expressão:**

```
.element {
  transform: ~"rotate(45deg)";
}
```

Nesse exemplo a expressão rotate(45deg) é escapada, permitindo que seja interpretada diretamente pelo CSS.

Aplique o escaping

Acompanhe alguns exemplos de uso do escaping no Less.

- **Escapando caracteres especiais:**

```
.element {
  content: ~"\"Quotes\"";
}
```

Nesse exemplo o caractere de aspas duplas " é escapado para ser tratado como um literal de CSS válido.

- **Uso do escaping com cautela:**
Recomendamos usar o escaping com cautela, pois você está abdicando dos recursos de processamento e manipulação do Less para essa parte específica do código.

Utilize mixins

O uso de mixins no Less ajuda a reduzir a repetição de código e a manter um estilo consistente em todo o projeto. Acompanhe algumas dicas sobre o uso de mixins no Less.

- **Declaração de mixins:**
  Para criar um mixin, use o nome do mixin e os estilos que deseja incluir. Por exemplo:

```
.name {
  propriedade: valor;
  // ...
}
```

- **Parâmetros de mixins:**
  Os mixins podem aceitar parâmetros para torná-los mais flexíveis e reutilizáveis. Para definir parâmetros, coloque-os entre parênteses na declaração do mixin. Por exemplo:

```
.mixin-name(@param1, @param2) {
  propriedade: @param1;
  // ...
}
```

Utilize mixins

O uso de mixins no Less ajuda a reduzir a repetição de código e a manter um estilo consistente em todo o projeto. Acompanhe algumas dicas sobre o uso de mixins no Less.

- **Passagem de argumentos para mixins:**
  Ao usar um mixin que possui parâmetros, você pode passar valores para esses parâmetros ao incluir o mixin. Por exemplo:

```
.selector {
  .mixin-name(valor1, valor2);
}
```

- **Valores padrão para parâmetros:**
  Você pode definir valores padrão para os parâmetros de um mixin, caso nenhum valor seja fornecido ao usá-lo. Para isso, atribua um valor padrão aos parâmetros na declaração do mixin. Por exemplo:

```
.mixin-name(@param: valor-padrão) {
  propriedade: @param;
  // ...
}
```

Utilize mixins

O uso de mixins no Less ajuda a reduzir a repetição de código e a manter um estilo consistente em todo o projeto. Acompanhe algumas dicas sobre o uso de mixins no Less.

- **Mixins com regras condicionais:**
  É possível usar lógica condicional, como when e if, dentro dos mixins para criar comportamentos dinâmicos com base nos valores dos parâmetros ou outras condições.

- **Mixins aninhados:**
  Você pode aninhar mixins dentro de outros mixins para criar estruturas de estilos mais complexas e reutilizáveis.

- **Importação de mixins:**
  Se você definir mixins em um arquivo separado, pode importá-los em outros arquivos Less usando a diretiva @import.

Crie mapas

Conheça exemplos de uso da função transform para manipular algumas propriedades.

- **Translate:**
  transform(translateX(100px));
  transform(translateY(-50%));
  transform(translate(50px, -20px));

- **Rotate:**
  transform(rotate(45deg));
  transform(rotateX(180deg));
  transform(rotateY(90deg));

- **Scale:**
  transform(scale(1.5));
  transform(scaleX(0.8));
  transform(scaleY(1.2));
  transform(scale(0.5, 1.2));

- **Skew:**
  transform(skewX(10deg));
  transform(skewY(-20deg));
  transform(skew(10deg, -20deg));

Crie mapas

- **Sobre a função transform:**
  É importante notar que a função transform é uma função de **saída**, o que significa que o Less a compila para CSS sem alterar sua sintaxe. Portanto, você pode usar a função transform diretamente no seu código Less e ela será traduzida corretamente para CSS ao compilar.

- **Funções de tempo para transição:**
  Além da função ease, há ainda outras funções como "linear", "ease-in", "ease-out", "ease-in-out" e também temos a opção de usar funções de tempo personalizadas com a função cubic-bezier().
  Cada uma dessas funções tem características diferentes e pode ser adequada para diferentes tipos de animações e efeitos de transição.

Crie mapas

Acompanhe agora algumas dicas sobre o uso de mapas no Less.

- **Declare um map:**
Para criar um map utilizar o @ antes do nome do map:

```
@cores: {
  Primaria: red;
  Secundária: blue;
}
```

- **Acesso a valores do map:**
Para acessar um valor específico de um map, utilize a notação de colchetes [] com a chave correspondente.
Por exemplo:

```
@map: (
  chave1: valor1,
  chave2: valor2,
  // ...
);
.selector {
  propriedade: @map[chave1];
}
```

Crie mapas

Acompanhe agora algumas dicas sobre o uso de mapas no Less.

Iteração sobre um map:
É possível percorrer todos os pares chave-valor de um map usando uma diretiva each.
Por exemplo:

```
@map: (
  chave1: valor1,
  chave2: valor2,
  // ...
);
.selector {
  each(@map, {
    @key: @value;
    // Fazer algo com a chave e o valor
  });
}
```

Crie mapas

Acompanhe agora algumas dicas sobre o uso de mapas no Less.

- **Atualização de valores do map:**
Você pode atualizar ou adicionar novos valores a um map atribuindo um novo valor a uma chave existente ou adicionando uma nova chave-valor.
Por exemplo:

```
@map: (
  chave1: valor1,
  chave2: valor2,
  // ...
);

@map[chave1]: novo-valor;
@map[chave3]: valor3;
```

- **Uso de maps com mixins**:
Os maps são especialmente úteis quando combinados com mixins. Você pode passar um map como argumento para um mixin e utilizar os valores correspondentes dentro do mixin. Isso permite criar mixins altamente flexíveis e configuráveis.

escola britânica de artes criativas & tecnologia

Bons estudos!