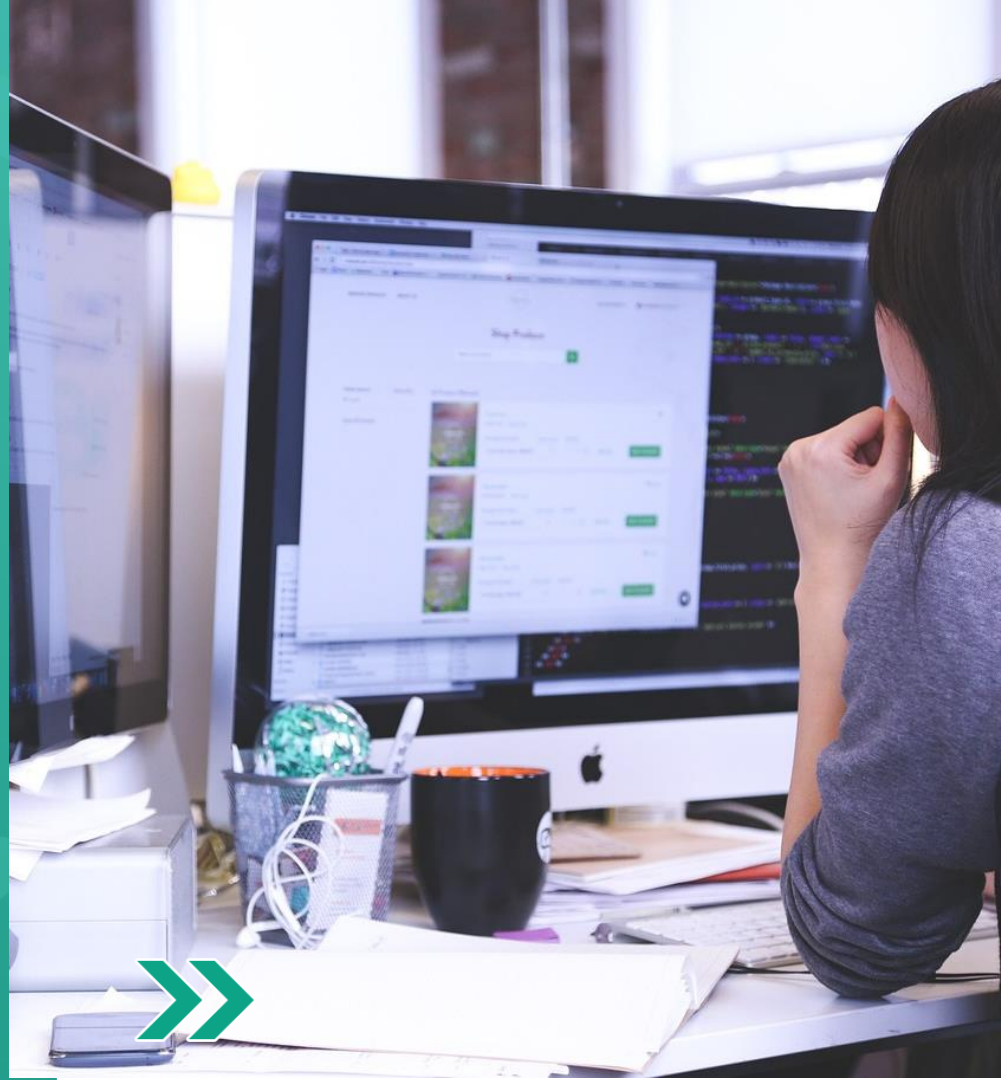
escola britânica de artes criativas & tecnologia

Profissão: Engenheiro Front-End

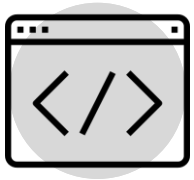
escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Introdução ao SASS

- **Conheça o Node**
- **Declare variáveis no SASS**
- **Crie módulos**
- **Use funções e operadores**



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

Conheça o Node

SASS:
Oferece recursos de herança, onde você pode criar estilos base que podem ser estendidos e modificados por seletores específicos. Isso pode reduzir a duplicação de código e facilitar a manutenção de estilos.

NPM:
Com o NPM você pode realizar várias tarefas, como:

- Instalar pacotes: Use o comando npm install para instalar pacotes e suas dependências a partir do repositório do NPM.
- Gerenciar dependências: O arquivo package.json do seu projeto contém a lista de dependências e versões específicas necessárias para o seu projeto. O NPM permite que você instale, atualize e remova pacotes com facilidade.
- Executar scripts: O NPM permite definir scripts personalizados no arquivo package.json para automatizar tarefas comuns, como a compilação de código, execução de testes ou qualquer outra ação desejada.

Conheça o Node

- Publicar pacotes: Se você criar um pacote ou uma biblioteca JavaScript útil, poderá publicá-lo no repositório do NPM para que outras pessoas possam instalá-lo e usá-lo em seus projetos.

Declare variáveis no SASS

- **Source ou src:**
  A escolha de uma pasta específica para armazenar os arquivos de origem ou código fonte do SASS é uma prática recomendada para manter a estrutura do projeto organizada e facilitar o processo de compilação. Entretanto, você pode escolher um nome diferente para a pasta que melhor se adeque à estrutura do seu projeto.

Crie módulos

Acompanhe, a seguir, algumas razões pelas quais usamos o @use no SASS.

- **Encapsulamento de estilos:**
  O @use ajuda a evitar o vazamento de estilos globais, pois ele importa apenas o que é explicitamente referenciado. Isso evita conflitos de nomes de variáveis e mixins, pois os módulos importados têm seu próprio escopo.

- **Melhor resolução de conflitos:**
  O @use possui um mecanismo de resolução de conflitos mais rigoroso em comparação com o @import. Ele não permite a duplicação de membros (variáveis, mixins, funções) por padrão. Isso ajuda a evitar problemas de conflito quando vários arquivos são importados.

- **Carregamento mais eficiente:**
  O @use carrega os módulos sob demanda. Isso significa que ele carrega apenas o que é necessário, tornando o processo de compilação mais eficiente e reduzindo o tamanho dos arquivos CSS resultantes..

Crie módulos

Acompanhe, a seguir, algumas razões pelas quais usamos o @use no SASS.

- **Melhor organização de código:**
  O @use encoraja a divisão do código em módulos, o que facilita a organização e a manutenção do código. Você pode importar apenas os módulos necessários em um determinado arquivo, tornando-o mais focado e conciso.

- **Melhor legibilidade e compreensão:**
  O @use torna mais claro quais módulos são usados em um arquivo SASS, tornando o código mais legível e facilitando a compreensão das dependências do projeto.

Use funções e operadores

É encorajado o uso de unidades relativas em vez de unidades absolutas, como pixels. A principal razão para isso é criar um design responsivo e flexível, que se adapte a diferentes tamanhos de tela e dispositivos. Acompanhe algumas vantagens de usar unidades relativas, como ems (em) e porcentagens (%), em vez de pixels (px) no SASS.

- **Responsividade**:
  Unidades relativas permitem que os elementos se ajustem automaticamente a diferentes tamanhos de tela. Por exemplo, definir a largura de um elemento em porcentagem (%) faz com que ele ocupe uma porção específica do espaço disponível, independentemente do tamanho da tela.

- **Escalabilidade**:
  Unidades relativas facilitam a criação de layouts escaláveis, onde os elementos se adaptam proporcionalmente ao tamanho do elemento pai. Isso é particularmente útil quando se trata de criar layouts flexíveis que precisam se ajustar a diferentes dispositivos e tamanhos de tela.

Use funções e operadores

É encorajado o uso de unidades relativas em vez de unidades absolutas, como pixels. A principal razão para isso é criar um design responsivo e flexível, que se adapte a diferentes tamanhos de tela e dispositivos. Acompanhe algumas vantagens de usar unidades relativas, como ems (em) e porcentagens (%), em vez de pixels (px) no SASS.

- **Acessibilidade:**
  Usar unidades relativas permite que os usuários personalizem o tamanho da fonte em seus navegadores. Isso significa que, ao usar ems (em) para definir o tamanho da fonte, por exemplo, o texto será dimensionado de acordo com as preferências do usuário, garantindo uma melhor acessibilidade.

- **Facilidade de manutenção:**
  Ao usar unidades relativas, como ems (em), você pode definir estilos de forma mais consistente e reutilizável. Se você precisar ajustar o tamanho ou a escala do seu design, pode simplesmente atualizar os valores das unidades relativas em um local centralizado, em vez de precisar ajustar manualmente todos os valores em pixels em todo o código.

Use funções e operadores

É encorajado o uso de unidades relativas em vez de unidades absolutas, como pixels. A principal razão para isso é criar um design responsivo e flexível, que se adapte a diferentes tamanhos de tela e dispositivos. Acompanhe algumas vantagens de usar unidades relativas, como ems (em) e porcentagens (%), em vez de pixels (px) no SASS.

- **Uso do pixel:**
  Apesar destas dicas, é importante mencionar que em alguns casos, o uso de pixels (px) pode ser apropriado e necessário, especialmente quando se trata de propriedades que não são afetadas pelo redimensionamento do layout, como bordas, sombras e imagens.

Use funções e operadores

Gitignore
No Git temos a possibilidade de ignorar alguns arquivos, removendo-os do sistema de versionamento e por consequência não enviando-os ao Github.
Para ignorar arquivos e pastas é necessário criar um arquivo chamado .gitignore e inserir nele o caminho do arquivo e/ou pasta que deverá ser ignorado.
É altamente recomendável que não se versione a pasta node_modules, que é criada ao instalar algum pacote Node. Isso se deve ao fato desta pasta conter arquivos binários que nem sempre serão os mesmos para todos os sistemas operacionais, além disto esta pasta pode ficar pesada dificultando o versionamento do projeto.

escola britânica de artes criativas & tecnologia

Bons estudos!