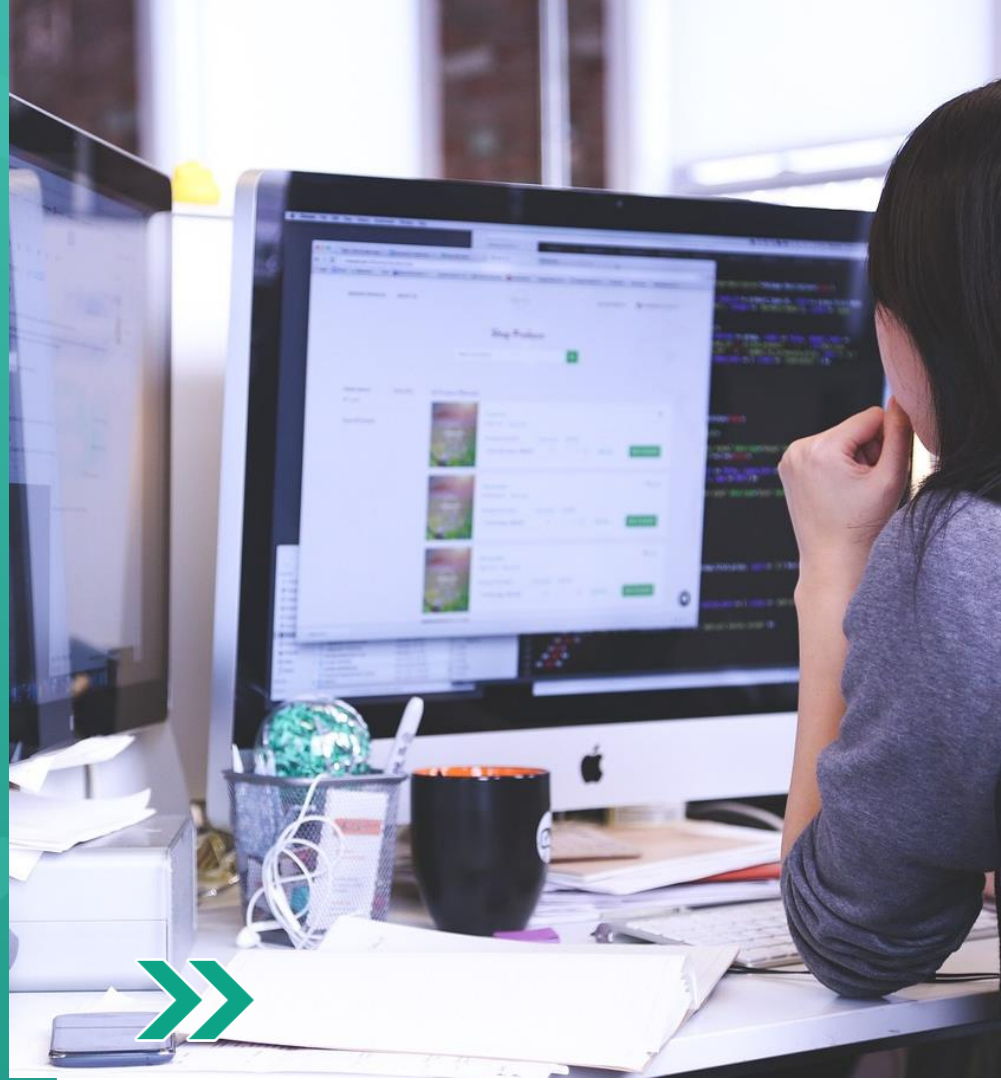
escola britânica de artes criativas & tecnologia

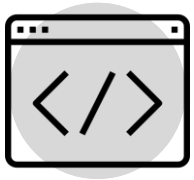
Profissão: Engenheiro Front-End

escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Introdução ao CSS in JS com React



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

- Escreva CSS com o JavaScript
- Produza com React
- Conheça o Styled Components
- Conheça o Atomic Design
- Construa a estrutura
- Estilize Sidebar
- Explore o recurso temas

Escreva CSS com o JavaScript

escola britânica de artes criativas & tecnologia

Styled Components:
Os benefícios do Styled Components incluem:

- Encapsulamento: Os estilos são definidos para um componente específico e não afetam outros elementos fora desse componente.
- Escopo de estilo: O Styled Components gera nomes de classes exclusivos para cada componente, evitando conflitos de estilos em toda a aplicação.
- Interpolação de props: É possível usar props React no template literal para alterar dinamicamente o estilo com base em diferentes estados do componente.
- Integração com temas: É fácil criar temas para a aplicação e aplicar estilos consistentes em toda a interface.

Escreva CSS com o JavaScript

- **Interpolação de string:**
  Use a interpolação de string para tornar o código mais legível e mais fácil de escrever, especialmente quando há várias variáveis a serem incluídas em uma string. Ela também ajuda a evitar erros comuns de concatenação de strings, como esquecer de adicionar espaços entre as partes da string. Além disso, em algumas linguagens, a interpolação de string pode ser mais eficiente em termos de desempenho do que a concatenação tradicional de strings, pois o compilador ou interpretador pode otimizar o código para evitar alocações de memória desnecessárias.

Escreva CSS com o JavaScript

escola britânica de artes criativas & tecnologia

- **Vendor prefixes:**
  A prática de usar vendor prefixes tem sido desencorajada em favor de recursos mais padronizados e a adoção de prefixos foi substituída por outras abordagens, como usar transpiladores CSS ou ferramentas de pós-processamento para adicionar automaticamente os prefixos necessários com base nas configurações de compatibilidade do projeto. Isso ajuda a simplificar o código CSS e a tornar o desenvolvimento mais eficiente, garantindo suporte para navegadores mais antigos, quando necessário.

Produza com React

Acompanhe as dicas sobre o EditorConfig.

- **Comentários descritivos:**
  A configuração do .editorconfig deve ser óbvia, caso contrário, adicione comentários explicativos para cada configuração. Isso ajudará os desenvolvedores a entenderem o propósito de cada configuração e garantirá que elas sejam aplicadas corretamente.

- **Colaboração em equipe:**
  Se você estiver trabalhando em um projeto React com uma equipe de desenvolvimento, é importante comunicar a existência do arquivo .editorconfig e suas configurações para garantir que todos os membros da equipe estejam cientes do estilo de código padronizado e possam segui-lo.

Produza com React

Acompanhe as dicas sobre o EditorConfig.

- **Verificando a formatação:**
  Antes de enviar código para controle de versão ou fazer um pull request, verifique se a formatação está correta usando as configurações do EditorConfig no seu editor. Isso ajudará a garantir que o código esteja seguindo o estilo padronizado definido no arquivo .editorconfig.

- **Atualizando as configurações:**
  À medida que o projeto progride ou novas configurações de estilo são necessárias, atualize o arquivo .editorconfig para refletir essas alterações. Assegure-se de comunicar essas atualizações para toda a equipe.

Produza com React

- **Com o ESLint você pode fazer:**
  - Uso de variáveis não declaradas ou não utilizadas.
  - Espaçamento e indentação inconsistente.
  - Chamadas de função que não estão sendo utilizadas corretamente.
  - Uso de práticas desencorajadas ou obsoletas.
  - Erros de sintaxe e formatação.

Produza com React

O eslint-plugin-react-hooks oferece regras específicas para verificar o uso adequado de hooks, incluindo:

- **Verificar as dependências de hooks:**
  Garante que todos os hooks usem suas dependências corretamente. Isso é importante para evitar problemas de atualização e re-renderização desnecessária de componentes.

- **Enforce Rules of Hooks:**
  Garante que os hooks sejam chamados somente a partir de componentes funcionais do React ou de outros hooks. Isso evita o uso indevido de hooks em lugares inadequados, como em classes ou funções regulares.

- **Verificar ordem de chamada de hooks:**
  Garante que os hooks sejam chamados sempre na mesma ordem em todos os renders de um componente, evitando comportamentos inesperados.

- **Detectar hooks inválidos:**
  Detecta o uso de hooks que não são válidos ou hooks personalizados com implementações erradas.

Produza com React

Conheça alguns dos recursos e benefícios do Prettier:

- **Configuração mínima:**
  O Prettier requer pouca ou nenhuma configuração. Ele já possui regras de formatação predefinidas que funcionam muito bem para a maioria dos projetos, reduzindo a necessidade de ajustes manuais nas configurações.

- **Automatização:**
  O Prettier pode ser facilmente integrado em pipelines de CI/CD (Integração Contínua e Entrega Contínua) e ser executado automaticamente em cada commit, garantindo que o código sempre esteja bem formatado e seguindo as regras estabelecidas.

- **Suporte a várias linguagens:**
  O Prettier oferece suporte a várias linguagens de programação, o que o torna uma escolha versátil para projetos que utilizam diferentes tecnologias.

Produza com React

Conheça alguns dos recursos e benefícios do Prettier:

- **Resolução de conflitos:**
  O Prettier resolve automaticamente conflitos de estilo entre diferentes contribuições de código, tornando a colaboração em equipe mais harmoniosa.

- **Evitar debates sobre estilo:**
  Como o Prettier impõe um padrão de formatação consistente, ele pode ajudar a evitar debates desnecessários sobre estilo de código, permitindo que os desenvolvedores foquem em questões mais relevantes.

Conheça o Styled Components

Acompanhe algumas dicas para usar o Styled Components.

- **Reutilização de estilos:**
  Use o recurso de reutilização de estilos para evitar a duplicação de código. Você pode criar um componente estilizado e reutilizá-lo em vários lugares.

- **Aninhamento:**
  O Styled Components permite aninhar estilos, o que pode ajudar a organizar melhor o código e a criar estilos mais complexos.

- **Utilizando bibliotecas de ícones:**
  Para utilizar bibliotecas de ícones em componentes estilizados, você pode criar um componente estilizado para o ícone e passar as props necessárias para controlar a cor, o tamanho etc.

- **Melhor organização de estilos:**
  À medida que seu aplicativo cresce, pode ser útil organizar seus estilos em arquivos separados. Você pode criar um arquivo styles.js ou styles.js e exportar os componentes estilizados para serem usados em todo o projeto.

Conheça o Styled Components

Styled Components – Extensão:
Com o Styled Components podemos extender o estilo de um componente estilizado para outro, para isso utilizamos a importação do Styled Components como construtor.
Exemplo de um botão perigo:

```
export const Botao = styled.button`
    font-weight: bold;
`;
export const BotaoPerigo = styled(Botao)`
    color: #fff;
    background-color: red;
`;
```

Agora a função styled irá receber como argumento o componente que iremos utilizar como base para a estilização, assim o BotaoPerigo terá além da cor de texto branca, fundo vermelho e texto em negrito como foi configurado no componente original.

Conheça o Atomic Design

Atomic Design:
Ao seguir a metodologia Atomic Design, as equipes de design e desenvolvimento podem criar uma biblioteca de componentes reutilizáveis e padronizados, garantindo uma maior consistência na aparência e comportamento de suas interfaces. Além disso, a abordagem Atomic Design ajuda a facilitar a colaboração entre designers e desenvolvedores, pois todos trabalham com os mesmos conceitos e terminologia.

Essa abordagem é especialmente útil para projetos maiores e equipes que desejam criar uma base sólida para o desenvolvimento e a manutenção de interfaces de usuário complexas. Ela também se encaixa bem com a filosofia de desenvolvimento de componentes do React e outras bibliotecas de UI, permitindo que os conceitos do Atomic Design sejam aplicados de forma prática no desenvolvimento de aplicações web.

Construa a estrutura

- **Sidebar :**
  O componente sidebar pode variar amplamente em complexidade e recursos, dependendo das necessidades do projeto. Pode incluir ícones, submenus, animações e outras interações personalizadas para melhorar a experiência do usuário. Além disso, o conteúdo exibido na sidebar também pode ser dinâmico e alterar-se com base na autenticação do usuário, estado da aplicação ou outras condições específicas do aplicativo.

Estilize sidebar

Acompanhe algumas dicas para usar o componente sidebar.

- **Separe componentes estilizados:**
  Se a sua sidebar tiver subcomponentes, como um item de navegação, um cabeçalho ou um ícone, crie componentes estilizados separados para cada um deles. Isso ajuda a manter o código mais organizado e reutilizável.

- **Utilize constantes para cores:**
  Em vez de usar valores de cores diretos, crie constantes para as cores utilizadas na sidebar. Isso torna mais fácil alterar as cores em toda a sidebar, caso seja necessário no futuro.

- **Use mixins ou funções para estilos repetitivos:**
  Se houver estilos repetitivos na sidebar, como espaçamentos ou bordas, crie mixins ou funções para reutilizá-los em diferentes partes do código.

Estilize sidebar

Acompanhe algumas dicas para usar o componente sidebar.

- **Utilize templates literais para estilos condicionais:**
  Se a sidebar tiver estilos condicionais com base em props, utilize templates literais para injetar esses estilos de forma dinâmica.
- **Use props para ajustar o layout:**
  Se a sua sidebar for retrátil, você pode usar as props para ajustar o layout conforme necessário.
- **Pseudoclasses e pseudoelementos:**
  Use pseudoclasses e pseudoelementos, como :hover, :active, ::before, ::after, para adicionar efeitos interativos e estilizar componentes forma dinâmica.
- **Separe estilos em arquivos:**
  Se a sidebar for complexa e contiver muitos estilos, considere separar os estilos em arquivos diferentes para manter um melhor controle e organização.
- **Teste em diferentes tamanhos de tela:**
  Garanta que ela seja responsiva e se ajuste adequadamente em dispositivos móveis e outros dispositivos.

Explore o recurso temas

Acompanhe algumas dicas sobre o uso de temas.

- **Organização do tema:**
  Ao criar o objeto theme, tente organizar as variáveis de estilo de forma lógica e coerente. Por exemplo, agrupe as cores, tipografia, espaçamentos e outros estilos relacionados em seções separadas. Isso tornará o objeto theme mais fácil de entender e manter à medida que o projeto cresce.

- **Defina um tema padrão:**
  É uma boa prática definir um tema padrão com valores para as variáveis de estilo, caso algum componente não forneça um tema específico. Isso garante que a aplicação tenha uma aparência consistente, mesmo que nem todos os componentes forneçam um tema personalizado.

- **Temas temáticos:**
  Além dos temas de aparência, o ThemeProvider também pode ser usado para temas temáticos, como temas para diferentes marcas, projetos ou clientes. Isso permite que você reutilize componentes em diferentes projetos, mas aplique diferentes estilos de acordo com o tema escolhido.

Explore o recurso temas

Acompanhe algumas dicas sobre o uso de temas.

- **Composição de temas:**
  Se você tem um tema padrão, mas deseja criar temas personalizados para componentes específicos, considere usar a composição de temas. Isso permite que você herde estilos do tema padrão e substitua apenas as variáveis de estilo necessárias.

- **Temas dinâmicos:**
  Se o seu tema precisar ser alterado dinamicamente durante a execução do aplicativo (por exemplo, para permitir que os usuários personalizem o tema), você pode usar um estado ou um contexto para gerenciar essas alterações e atualizar o ThemeProvider de acordo.

- **Testes:**
  Considere a criação de testes para garantir que o ThemeProvider esteja aplicando corretamente os temas e que as variáveis de estilo estejam sendo acessadas corretamente em toda a aplicação.

escola britânica de artes criativas & tecnologia

Bons estudos!