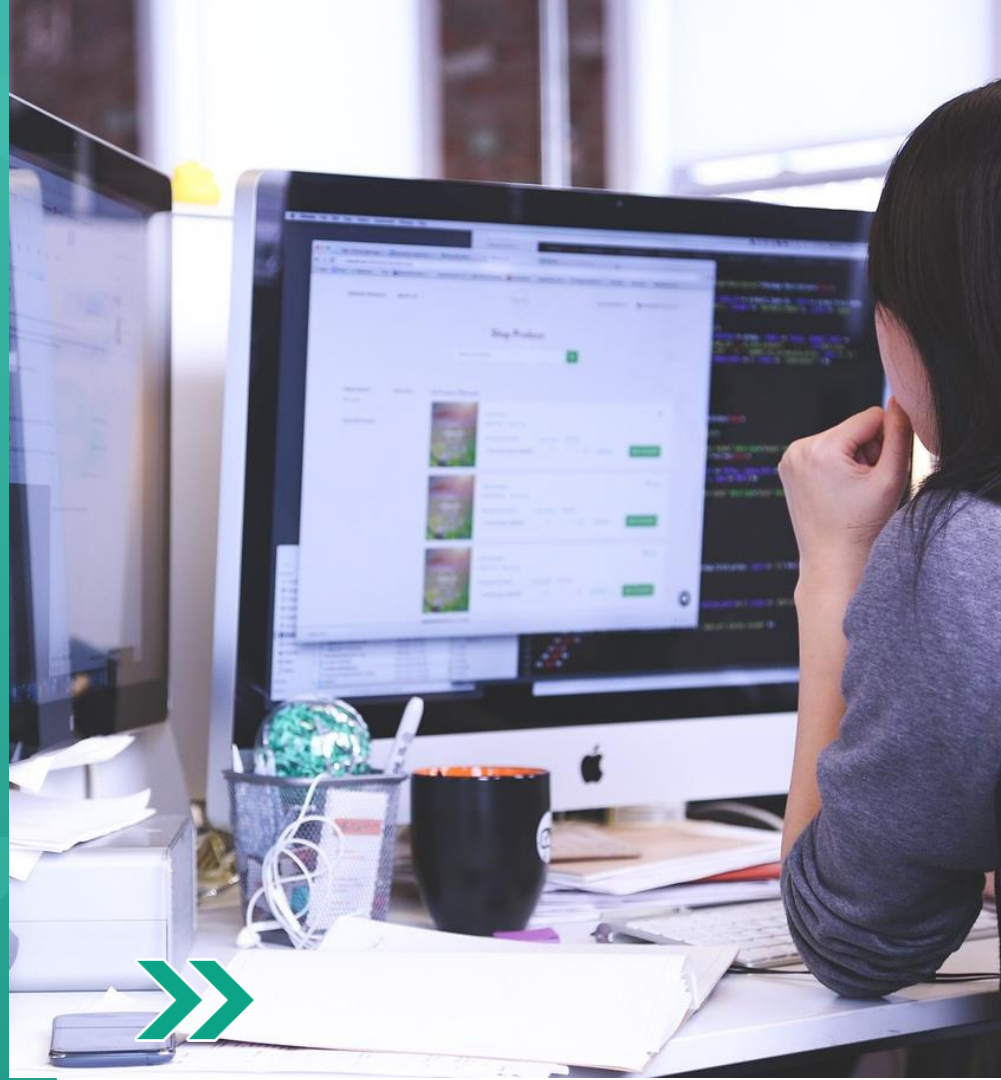
escola britânica de artes criativas & tecnologia

Profissão: Engenheiro Front-End

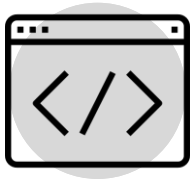
escola britânica de artes criativas & tecnologia

BOAS PRÁTICAS



Introdução ao Redux com React

- Conheça o problema de prop drilling
- Aprenda sobre a Arquitetura flux
- Conheça o Redux
- Explore o Redux Toolkit
- Use seletores
- Conheça o Redux Toolkit Query



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

Conheça o problema de prop drilling

Acompanhe algumas dicas para resolver o prop drilling.

- **Componentes puros (Pure Components):**
  Componentes puros são componentes que não têm efeitos colaterais e sempre retornam o mesmo resultado para o mesmo conjunto de props. Eles podem ajudar a minimizar a propagação de props desnecessárias através da árvore de componentes.

- **Componentes de ordem superior (Higher-Order Components):**
  Você pode usar o padrão de Higher-Order Components (HOC) para envolver componentes que precisam acessar o mesmo conjunto de props em um componente HOC. Isso pode simplificar a passagem de props repetidas vezes.

Conheça o problema de prop drilling

Acompanhe algumas dicas para resolver o prop drilling.

- **Renderização condicional:**
  Utilize renderização condicional para renderizar ou não um componente dependendo da existência de uma prop específica ou do estado.

- **Libraries de gerenciamento de estado:**
  Além do Redux e do Context API, existem outras bibliotecas de gerenciamento de estado, como MobX ou Zustand, que podem ajudar a resolver o problema de prop drilling.

- **Reestruture a árvore de componentes:**
  Às vezes, reestruturar a árvore de componentes pode ser uma solução simples. Considere mover os componentes que precisam das props mais próximos uns dos outros na hierarquia.

Aprenda sobre a arquitetura flux

- **Integre o Redux DevTools:**
  O Redux DevTools é uma ferramenta poderosa para depurar aplicações Flux. Ela permite acompanhar as Actions despachadas, o estado atual da aplicação e até mesmo viajar no tempo para analisar a sequência de mudanças no estado.

Conheça o redux

- **Imutabilidade:**
  O estado no Redux é imutável, o que significa que ele não pode ser modificado diretamente. Sempre que uma alteração é feita no estado, uma nova cópia do estado é criada e o estado anterior permanece inalterado. Isso ajuda a evitar efeitos colaterais e torna a depuração mais fácil.

- **Mantenha a store simples:**
  A store Redux deve conter apenas o estado global essencial para a aplicação. Evite colocar informações temporárias ou derivadas na store, pois isso pode dificultar a manutenção e gerar confusão.

Conheça o redux

- **Divida o estado em slices:**
  Divida o estado global em "slices" menores e independentes. Cada slice deve conter informações relacionadas a uma parte específica da aplicação. Isso facilita a manutenção e evita que a store se torne muito complexa.

- **Evite mutações diretas:**
  O Redux enfatiza a imutabilidade do estado. Evite modificar diretamente o estado dentro dos reducers. Em vez disso, retorne um novo objeto de estado ou utilize bibliotecas como o Immer para facilitar a imutabilidade.

Conheça o redux

- **Aproveite a comunidade:**
  O Redux tem uma comunidade ativa e rica em recursos. Aproveite a vasta quantidade de bibliotecas, exemplos e tutoriais disponíveis para melhorar suas habilidades e resolver desafios específicos.

- **Seja consistente:**
  Mantenha um padrão consistente de nomenclatura para suas actions, reducers, seletores e outros elementos do Redux. Isso torna o código mais legível e facilita a colaboração entre os membros da equipe.

Explore o Redux Toolkit

Acompanhe algumas dicas e boas práticas relacionadas ao uso de letras maiúsculas no Redux Toolkit.

- **Action Types em letras maiúsculas:**
  A convenção mais comum é usar os tipos de ação (action types) em letras maiúsculas e separados por underline (snake_case). Isso torna os types mais legíveis e distintos de outros nomes no código.

- **Nome de Slice em CamelCase:**
  Ao criar um "slice" usando createSlice, é comum usar o formato CamelCase para o nome do slice. Por exemplo, um slice que gerencia o contador pode ser nomeado como counterSlice.

- **Action Creators:**
  Ao criar action creators usando createSlice, eles são gerados automaticamente com nomes em CamelCase. Esses action creators podem ser usados diretamente em dispatch, em vez de chamar manualmente o dispatch com objetos de ação.

- **Constantes em letras maiúsculas:**
  Se você estiver usando constantes para definir os tipos de ação, é uma boa prática nomeá-las em letras maiúsculas.

Explore o Redux Toolkit

Acompanhe agora algumas dicas para usar o Redux Toolkit de forma eficaz.

- **Evite ações assíncronas diretamente no reducer:**
  Os reducers do Redux devem ser funções puras, portanto, evite fazer operações assíncronas diretamente dentro deles. Use middlewares como o redux-thunk ou redux-saga para lidar com operações assíncronas e despachar actions apropriadas.

- **Use configureStore para criar a store:**
  O configureStore é um método do Redux Toolkit que simplifica a criação da store, incluindo a configuração de middlewares e a ativação das ferramentas de desenvolvedor, como o Redux DevTools.

- **Evite a estruturação excessiva do estado:**
  Embora seja importante dividir o estado em fatias independentes, evite criar uma estrutura muito granular ou complexa. Mantenha o estado tão simples quanto possível para facilitar a manutenção e o entendimento.

Use seletores

- **Seletores simples e reutilizáveis:**
  É uma boa prática criar seletores simples e reutilizáveis para extrair partes específicas do estado. Dessa forma, você pode usar o mesmo seletor em diferentes componentes, evitando duplicação de código.

- **Evite selecionar o estado inteiro:**
  É tentador selecionar todo o estado global para um componente, mas isso pode levar a uma renderização excessiva e ineficiente. Selecione apenas as partes do estado que o componente realmente precisa para evitar renderizações desnecessárias.

- **Separe seletores por contexto:**
  Se sua aplicação tiver várias partes do estado com lógica específica para cada contexto, é uma boa prática criar seletores para cada contexto em vez de criar um seletor único que abrange toda a aplicação. Isso torna o código mais organizado e mais fácil de manter.

Use seletores

- **Compartilhe seletores personalizados:**
  Se você tiver seletores que podem ser úteis para outras partes da aplicação ou que possam ser reutilizados em projetos futuros, considere colocá-los em um arquivo separado e compartilhá-los como um módulo personalizado.

Conheça o Redux Toolkit Query

Conheça os principais recursos do Redux Toolkit Query.

- **Configuração declarativa:**
  O RTK Query permite que você defina endpoints de consulta de forma declarativa, especificando informações como a URL da API, os métodos HTTP a serem usados, parâmetros, cabeçalhos e muito mais.

- **Caching e refetching automáticos:**
  O RTK Query oferece um sistema de cache embutido que evita solicitações duplicadas à mesma API, reduzindo o tráfego de rede e melhorando o desempenho da aplicação. Ele também suporta recarregar automaticamente os dados expirados no cache quando os componentes são montados ou quando ocorrem ações específicas.

- **Atualizações otimistas:**
  Com o RTK Query, você pode configurar atualizações otimistas para fornecer feedback imediato ao usuário, mesmo antes de os dados serem enviados ao servidor. Isso melhora a experiência do usuário e reduz a percepção de latência.

Conheça o Redux Toolkit Query

Conheça os principais recursos do Redux Toolkit Query.

- **Gerenciamento automático do estado:**
  O RTK Query gerencia automaticamente o estado de cada endpoint de consulta na store do Redux. Isso inclui informações sobre o estado das solicitações (carregando, erro, sucesso), bem como os dados retornados pela API.

- **Customização e interceptação:**
  O RTK Query oferece muitas opções de personalização para atender às necessidades específicas da sua aplicação. Você pode interromper o fluxo de solicitações, adicionar cabeçalhos personalizados, transformar dados antes de armazená-los no estado e muito mais.

- **Integração com outros recursos do Redux Toolkit:**
  O RTK Query é totalmente compatível com outras funcionalidades do Redux Toolkit, como createSlice e configureStore.

Conheça o Redux Toolkit Query

Acompanhe as dicas sobre o uso de Middleware no Redux Toolkit Query.

- **Adicione lógica de autenticação:**
  Você pode usar um middleware para adicionar automaticamente cabeçalhos de autorização (por exemplo, um token JWT) em todas as solicitações HTTP, facilitando a autenticação na API.

- **Trate erros de forma centralizada:**
  Utilize um middleware para capturar e tratar erros de solicitação HTTP em um local centralizado. Isso ajuda a lidar com erros de maneira consistente em toda a aplicação.

- **Transforme dados:**
  Se necessário, faça transformações nos dados de solicitação ou resposta, como converter formatos de data ou modificar o conteúdo da solicitação antes de enviá-la à API.

- **Cache de solicitações:**
  Se você deseja implementar caching de solicitações para evitar chamadas duplicadas à mesma API, um middleware pode ajudar a gerenciar o cache e evitar solicitações redundantes.

Conheça o Redux Toolkit Query

Acompanhe as dicas sobre o uso de Middleware no Redux Toolkit Query.

- **Manipule paginamento:**
  Se a sua API usa paginamento, você pode usar um middleware para automatizar a lógica de carregar mais dados ao chegar ao final de uma página.

- **Monitoramento e logging:**
  Adicione funcionalidades de logging para monitorar o tráfego de solicitações e respostas da API, o que pode ser útil para fins de depuração e análise de desempenho.

- **Intercepte respostas para processamento global:**
  Você pode usar um middleware para processar respostas da API globalmente, como realizar ações específicas com base no status de resposta (por exemplo, atualizar o token JWT expirado).

- **Crie headers personalizados:**
  Se a sua API requer headers específicos em todas as solicitações, um middleware pode adicionar esses headers automaticamente para você.

Conheça o Redux Toolkit Query

Acompanhe as dicas sobre o uso de Middleware no Redux Toolkit Query.

escola britânica de artes criativas & tecnologia

- **Evite replicação de lógica:**
  Utilize middlewares para centralizar e reutilizar lógica comum em toda a aplicação. Isso evita a replicação desnecessária de código em vários endpoints de consulta.

escola britânica de artes criativas & tecnologia

Bons estudos!