Documentação do arquivo `openAI.js`

Introdução

Este arquivo contém a implementação de um componente Lightning Web Component (LWC) chamado `OpenAI`. Ele utiliza a integração com a API do Salesforce Apex para realizar uma funcionalidade de verificação de texto, utilizando o método `textcompletionCeck` da classe Apex `openAIclass`.

Descrição

O componente `OpenAI` permite que o usuário insira um texto em um campo de entrada e, ao clicar em um botão, envie esse texto para o servidor Apex. O servidor processa o texto e retorna uma resposta, que é exibida no componente. Este componente é útil para realizar verificações ou análises de texto utilizando serviços externos, como o OpenAI.

Estrutura

O arquivo contém:

- Importações de módulos necessários para o funcionamento do componente.
- Declaração de variáveis rastreáveis (`@track`) para armazenar o texto inserido pelo usuário e a resposta retornada pelo servidor.
- Métodos para manipular eventos e realizar chamadas ao servidor Apex.

Dependências

Este arquivo depende dos seguintes elementos:

- **Salesforce Apex**: Classe `openAIclass` e método `textcompletionCeck`.
- **LWC Framework**: Módulos `LightningElement`, `track` e `api`.

Imports

Os seguintes módulos e classes são importados:

```
import { LightningElement, track, api } from 'lwc';
import fetchCompletion from '@salesforce/apex/openAIclass.textcompletionCeck';
```

- `LightningElement`: Base para criar componentes LWC.
- `track`: Decorador para rastrear alterações em variáveis.
- `api`: Decorador para expor propriedades públicas.
- `fetchCompletion`: Método Apex para realizar a verificação de texto.

Variáveis

`texttocheck`

- **Descrição**: Armazena o texto inserido pelo usuário no campo de entrada.
- **Tipo**: `String`
- **Decorador**: `@track`

`responseReturned`

- **Descrição**: Armazena a resposta retornada pelo servidor Apex após a verificação do texto.
- **Tipo**: `String`
- **Decorador**: `@track`

Métodos

`assignData(event)`

- **Descrição**: Captura o valor inserido pelo usuário no campo de entrada e o armazena na variável `texttocheck`.
- **Parâmetros**:
  - `event`: Evento disparado pelo campo de entrada.
- **Funcionamento**:
  - Obtém o valor do campo de entrada através de `event.target.value`.
  - Armazena o valor na variável `texttocheck`.
  - Exibe o valor no console para depuração.

`doSearch()`

- **Descrição**: Envia o texto armazenado em `texttocheck` para o servidor Apex e processa a resposta.
- **Funcionamento**:
  - Chama o método Apex `fetchCompletion` passando o texto como parâmetro.
  - Armazena o resultado retornado na variável `responseReturned`.
  - Exibe o resultado no console para depuração.
  - Em caso de erro, exibe um alerta com os detalhes do erro.

Exemplo

Abaixo está um exemplo de como usar o componente `OpenAI`:

```
<template>
    <lightning-input label="Digite o texto para verificar" onchange={assignData}></lightning-input>
    <lightning-button label="Verificar Texto" onclick={doSearch}></lightning-button>
    <p>Resposta: {responseReturned}</p>
</template>
```

Diagrama de Dependência

O diagrama abaixo ilustra as dependências entre os métodos e variáveis do componente:

```
classDiagram
    class OpenAI {
        +@track texttocheck : String
        +@track responseReturned : String
        +assignData(event)
        +doSearch()
    }
    OpenAI --> fetchCompletion : Apex Method
```

Notas

- Certifique-se de que a classe Apex `openAIclass` e o método `textcompletionCeck` estejam corretamente implementados e acessíveis.
- O componente utiliza o decorador `@track` para garantir que as alterações nas variáveis sejam refletidas na interface do usuário.

Vulnerabilidades

- **Erro de conexão com o servidor Apex**: Caso o servidor não esteja acessível ou o método `textcompletionCeck` não esteja implementado corretamente, o componente exibirá um alerta com o erro.
- **Validação de entrada**: Não há validação para o texto inserido pelo usuário. É recomendável adicionar validações para evitar entradas inválidas ou maliciosas.