Instrukcja podłączenia biblioteki lib.webphone.client.js po stronie klienta

Krok 1: Dodaj bibliotekę do projektu

W sekcji **<head>** swojego pliku HTML dodaj następujący:

```
<script type="text/javascript"
src="https://webrtc.github.io/adapter/adapter-latest.js"></script>
<script type="module" src="ścieżka/do/lib.webphone.client.js"></script>
```

Powyższy kod dodaje plik adapter.js, który dostarcza wsparcie dla WebRTC w różnych przeglądarkach.

Krok 2: Inicjalizacja biblioteki

1. Najpierw należy zainicjalizować bibliotekę.

```
<script type="module">
    const webPhoneCaller = new WebPhoneCaller('elementId')
```

W powyższym kodzie należy zastąpić `elementId` odpowiednim id dostarczonym przez administratora serwera.

Krok 3: Wykorzystanie funkcji biblioteki

1. Jeżeli w adresie url nie będzie zawarty token połączenie nie zostanie nawiązane.
   https://example.com?token=tokenId

2. Funkcje do obsługi biblioteki:
   a. **call()**

   Funkcja **„call”** umożliwia nawiązanie połączenia. Po wywołaniu tej funkcji rozpocznie się wywoływanie połączenia z numerem pobranym z API. Przykładowe użycie:

```
const webPhoneCaller = new WebPhoneCaller('elementId')
webPhoneCaller.call()
```

   b. **stop()**

   Funkcja **„stop”** umożliwia zakończenie połączenia (podczas łączenia lub w trakcie trwania rozmowy). Po wywołaniu tej funkcji, zostanie wywołany event **„outgoingCallFailed”.** Przykładowe użycie:

```
const webPhoneCaller = new WebPhoneCaller('elementId')
webPhoneCaller.stop()
```

3. Funkcje obsługujące eventy:

Nazwa	Event	Zastosowanie
onPhoneConnected()	connected	Potwierdza podłączenie użytkownika do serwera. Event ten jest wywołany za każdym razem jak użytkownik wywoła połączenie.
onPhoneDisconnected()	disconnected	Potwierdza odłączenie użytkownika od serwera. Event ten jest wywołany zawsze po zakończeniu rozmowy.
onOutgoingCallStart(numer)	outgoingCallStart	Uruchamiana po naciśnięciu przycisku dzwonienia. Funkcja pozwala pokazać numer telefonu na jaki dzwoni (przekazuje numer pobrany z API)
onProgressConnection()	progress	Uruchamiana gdy trwa nawiązywanie połączenia z docelowym numerem. Event ten jest wywoływany jak zostanie naciśnięty przycisk dzwonienia.
onOutgoingCallAnswer()	outgoingCallAnswered	uruchamiana, gdy połączenie zostanie nawiązane
onOutgoingCallEnded(reason)	outgoingCallEnded	wywołana po zakończeniu rozmowy po stronie odbierającego (reason jest ustawiony na „Terminated”). Może być użyta, aby poinformować użytkownika, że rozmowa została zakończona.
onOutgoingCallFailed(reason)	outgoingCallFailed	wywołana po zakończeniu rozmowy po stronie dzwoniącego (użytkownika) – reason jest ustawiony na „Canceled”. Może być użyta, aby poinformować użytkownika, że rozmowa została zakończona.
onGenericError(error)	error	Wykorzystywana do obsługi błędów

Przykładowe użycie:

```
const webPhoneCaller = new WebPhoneCaller('elementId')

webPhoneCaller.onPhoneConnected = () => {
     console.log('phone connected')

}
```

4. Zaawansowana obsługa eventów:

Nazwa	Event	Zastosowanie
onPhoneRegistered()	registered	potwierdza zarejestrowanie użytkownika do serwera. Konfiguracja potrzebna do rejestracji pobierana jest z API i jest zależna od elementId i tokena.
onPhoneUnregistered ()	unregistered	potwierdza odrejestrowanie użytkownika z serwera

onPhoneUnregisteredFailed (error)	registrationFailed	zwraca błąd podczas próby odrejestrowania użytkownika z serwera

5. Przykładowy plik index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Document</title>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <script type="text/javascript"
src="https://webrtc.github.io/adapter/adapter-latest.js"></script>
  <script type="module" src="./lib.webphone.client.js"></script>
</head>
<body>
  <button id="elementId">Zadzwoń</button>

  <script type="module">
    const callBtn = document.getElementById('elementId')
    const webPhoneCaller = new WebPhoneCaller('elementId')
    callBtn.addEventListener('click', () => {
      if (webPhoneCaller) {
        //wywołanie rozmowy przez użytkownika
        webPhoneCaller.call()
      }
    })
    // Funkcja kończąca rozmowę
    const endCallFunc = () => {
      if (webPhoneCaller) {
        // Zakończenie rozmowy przez użytkownika
        webPhoneCaller.stop()
      }
    }

    webPhoneCaller.onPhoneConnected = () => {
      console.log('connected')
    }
    webPhoneCaller.onPhoneDisconnected = () => {
      console.log('disconnected')
    }
    webPhoneCaller.onProgressConnection = () => {
      console.log('ringing')
      // W czasie łączenia rozmowy
      // po ponownym kliknięciu na przycisk dzwonienia
      // można zakończyć rozmowę
      callBtn.addEventListener('click', endCallFunc)
    }
    webPhoneCaller.onOutgoingCallStart = (callNumber) => {
      console.log('start call', callNumber)
    }
    webPhoneCaller.onOutgoingCallAnswered = (value) => {
      console.log('call answered', value)
      // W czasie trwania rozmowy
      // po ponownym kliknięciu na przycisk dzwonienia
```

```
      // można zakończyć rozmowę
      callBtn.addEventListener('click', endCallFunc)
    }
    webPhoneCaller.onOutgoingCallEnded = (value) => {
      console.log('call ended', value)
      //po zakończeniu rozmowy usuwany jest event kończący rozmowę
      callBtn.removeEventListener('click', endCallFunc)
    }
    webPhoneCaller.onOutgoingCallFailed = (value) => {
      console.log('call ended by user', value)
      //po zakończeniu rozmowy usuwany jest event kończący rozmowę
      callBtn.removeEventListener('click', endCallFunc)
    }
    webPhoneCaller.onGenericError = (e) => {
      console.log('onGenericError', e)
    }

  </script>
</body>
</html>
```