

```
<!DOCTYPE html>
<html lang="en">

<head>
  <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">
  <title>Document</title>
  <!-- biblioteka potrzebna do obsługi Apple -->
  <script type="text/javascript" src="https://webrtc.github.io/adapter/adapter-
latest.js"></script>
  <script type="module" src="./lib.webphone.client.js"></script>
  <script type="module" src="./lib.vcccallbtn.js"></script>
  <script type="module" src="./lib.vccalerttoast.js"></script>
  <!-- biblioteka font-awesome potrzebna do wyświetlenia ikon w tagach vcc-alert-
toast i vcc-call-btn -->
  <link href="/www/style/fa/css/all.min.css" rel="stylesheet" />
  <style>
    .container {
      display: flex;
      align-items: center;
      position: absolute;
      bottom: 20px;
      right: 20px;
    }

    /* Odkomentuj w celu zmiany kolorów buttona i alertu */
    /* vcc-call-btn,
    vcc-alert-toast {
      --standbyColor: #527CB9;
      --standbyColorFont: #ffffff;
      --errorColor: #fc1414;
      --errorColorFont: white;
      --endedColor: #527CB9;
      --endedColorFont: #ffffff;
      --ringingColor: #F29A05;
      --ringingColorFont: #ffffff;
      --answeredColor: #6aa84f;
      --answeredColorFont: #ffffff;
      --disabledColor: #000000;
    } */
  </style>

</head>

<body>
```

```

<div class="container">
  <!--
  wywołanie tagu z alertami
  -> mode:
    - standby - informacyjny alert z domyślną wiadomością "Skontaktuj się z nami,
aby uzyskać pomoc"
    - ringing - alert dzwonienia z domyślną wiadomością "Łączenie..."
    - answered - alert odebranej rozmowy z domyślną wiadomością "Rozmowa odebrana"
    - ended - alert zakończonej rozmowy z domyślną wiadomością "Rozmowa zakończona"
    - error - alert z błędem z domyślną wiadomością "Problem z połączeniem. Proszę
spróbować później."
  -> message: Ustawia własną wiadomość w okienku
  -> auto-close: true | false - czy ma samo zamykać się po czasie
  -> close-button: true | false - czy ma być wyświetlony przycisk w prawym górnym
rogu zamykający okienko
  -->
  <vcc-alert-toast id="alertToast" mode="" message="" auto-close="true" close-
button="true"></vcc-alert-toast>

  <!--
  wywołanie buttona do dzwonienia
  -> mode:
    - standby - tryb podstawowy
    - ringing - tryb dzwonienia
    - ended - tryb końca rozmowy
    - answered - tryb odebranej rozmowy
    - off - tryb wyłączony
  -->
  <vcc-call-btn id="elementId" mode="standby"></vcc-call-btn>
</div>

<script type="module">
  /*
    new WebPhoneCaller(elementId: string, adresUrl: string)
      - elementId = id podany przez administratora serwera,
      - adresUrl = adresUrl serwera
  */
  const webPhoneCaller = new WebPhoneCaller('elementId')

  const vccCallBtn = document.getElementById('elementId')

  const vccAlertToast = document.getElementById('alertToast')
  //po kliknięciu zaczyna dzwonić
  vccCallBtn.addEventListener('click', () => {
    if (webPhoneCaller) {
      //nawiązanie połączenia
      webPhoneCaller.call()
    }
  })

```

```

})

//wyświetlenie informacyjnego alertu o treści "Skontaktuj się z nami, aby
uzyskać pomoc"
setTimeout(() => {
  if (vccAlertToast.getAttribute('mode').length == 0) {
    vccAlertToast.setAttribute('mode', 'standby')
  }
}, 3000)

// Funkcja kończąca rozmowę
const endCallFunc = () => {
  if (webPhoneCaller) {
    // Zakończenie rozmowy przez użytkownika
    webPhoneCaller.stop()
  }
}

webPhoneCaller.onProgressConnection = () => {
  // W czasie łączenia rozmowy
  // po ponownym kliknięciu na przycisk dzwonienia
  // można zakończyć rozmowę
  vccCallBtn.addEventListener('click', endCallFunc)
  //przełączenie tagów w tryb 'ringing'
  vccCallBtn.setAttribute('mode', 'ringing')
  vccAlertToast.setAttribute('mode', 'ringing')
}

webPhoneCaller.onOutgoingCallAnswered = (value) => {
  // W czasie rozmowy
  // po ponownym kliknięciu na przycisk dzwonienia
  // można zakończyć rozmowę
  vccCallBtn.addEventListener('click', endCallFunc)
  //przełączenie tagów w tryb 'answered'
  vccAlertToast.setAttribute('mode', 'answered')
  vccCallBtn.setAttribute('mode', 'answered')
}

webPhoneCaller.onOutgoingCallFailed = (value) => {
  //usunięcie możliwości zakończenia rozmowy na kliknięcie
  vccCallBtn.removeEventListener('click', endCallFunc)

  if (value == 'Canceled') {
    //przełączenie tagów w tryb 'ended'
    vccCallBtn.setAttribute('mode', 'ended')
  }
}

```

```

    vccAlertToast.setAttribute('mode', 'ended')
    //po czasie przełączenie przycisków w stan standby
    setTimeout(() => {
        if(vccCallBtn.getAttribute('mode') == 'ended') {
            vccCallBtn.setAttribute('mode', 'standby')
            vccAlertToast.setAttribute('mode', 'standby')
        }
    }, 2500)
} else {
    vccCallBtn.setAttribute('mode', 'off')
    alertToast.setAttribute('mode', 'error')
    alertToast.setAttribute('message', value)
}
}

webPhoneCaller.onOutgoingCallEnded = (value) => {
    vccCallBtn.removeEventListener('click', endCallFunc)

    //przełączenie tagów w tryb 'ended'
    vccCallBtn.setAttribute('mode', 'ended')
    vccAlertToast.setAttribute('mode', 'ended')
    //po czasie przełączenie przycisków w stan standby
    setTimeout(() => {
        if(vccCallBtn.getAttribute('mode') == 'ended') {
            vccCallBtn.setAttribute('mode', 'standby')
            vccAlertToast.setAttribute('mode', 'standby')
        }
    }, 2500)
}

webPhoneCaller.onGenericError = (e) => {
    console.log('onGenericError', e)

    //przełączenie tagów w tryb 'off', który blokuje przycisk i wznowić można
    tylko na odświeżenie
    vccCallBtn.setAttribute('mode', 'off')
    vccAlertToast.setAttribute('mode', 'error')
    vccAlertToast.setAttribute('message', e.message)
}

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

```