WEBPHONE HIDDEN

The Webphone is WebRTC webphone that was design to work in direct in the browser and TotalVoice service provider create a RESTful approach to communicate with their Janus server. As it works direct on the browser the code must be loaded on that to load webRTC client and connect with SIP server.

The process is very easy.

To work on a Angular 2+ project we need to install (npm i jquery and) and load jquery.

Once the page is loaded the webphone nees only 1.5 ~5 seconds to be ready to use.

For the Webphone works we need to load two items in the page:

```
<div id="widget-evoline-api-container" style="display:inline" width="10"
height="20"></div>
```

And load this script:

```
<script
src="https://api.totalvoice.com.br/w3?key=cd4950a8314ad12b75243c1f517d85a5&tipo=hid
den&ver=2&fechar_fim=1"></script>
```

This src above is a unique URL from where the Webphone will be loaded on the browser. That means each user has their own key that should be inject on the middle

```
<script
src="https://api.totalvoice.com.br/w3?key={webphone_key}&tipo=hidden&ver=2&fechar_f
im=1"></script>
```

Once this URL is invoked it inject an iframe on the page as seem bellow:

```
    (function(){
      var c = $('#widget-evoline-api-container');
      var i = $('<iframe id="webphone" allow="microphone">');
      i.attr('src',
"https://api.totalvoice.com.br/w3/phone.php?key=cd4950a8314ad12b75243c1f517d85a5&ti
po=hidden");
      i.css({padding: 0, margin: 0, width: '100%', height: '100%', border:
'none'});
      c.append(i);
      c.css('display', 'none');
    })();
```

Note: This WEBPHONE uses WebRTC as stack to connect with PSTN, WebRTC works only in some browsers as seem bellow.

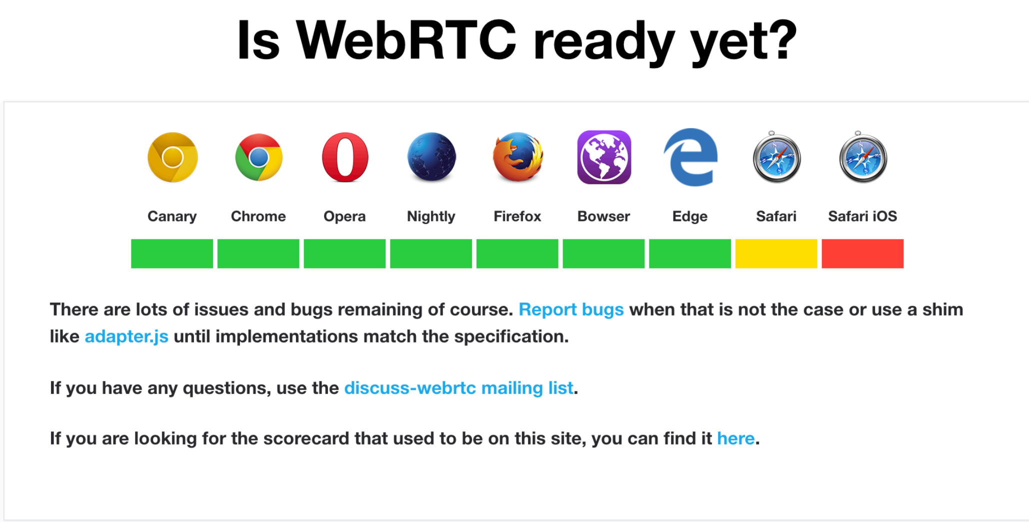

Figure 1 at http://iswebrtcreadyyet.com/

So it's important to advice users about browser support.

Functions and messages to execute commands on SIP server

The webphone_key is generated when an extension is created on TotalVoice REST (https://api.totalvoice.com.br/doc/) API
POST /ramal, make some sense create that and storage the key in the user data to recovery when they are loggedin

Or can be recovered

GET /webphone

All SIP server messages are handled by

```
window.onmessage
```

e.g

```
        window.onmessage = function(e) {
            if (e.data.message == 'chegandoChamada') {
                $("#incomingCall").text("Incoming Call");
            }
```

and you can take actions based on that. Below are all msgs returns from the Server

```
//e.data.message and idenifield a kind of message brought from the server
```

```
// message: chegandoChamada alerts when there is an incoming call
    // e.data.numeroChegando bring incomming call number idenification
    // e.data.numeroDestino bring number is called on webphone
    // e.data.chamadaRecebidaId call_id of incomming call
// message: status alerts about currrent call status changes
    // e.data.status brings current call status (conectado, desconectado, chamando, 
encerrada, conversando)
// message: chamada_id msg with call_id
    // e.data.chamada_id call_id of outgoing call
// message: status_erro msg with SIP server erros
    // e.data.status_erro bring erros msgs from SIP server
// message: stats_webphone msg with webphone status
    // e.data.internet bring internet current status
    // e.data.computador bring computer performance current status
```

All SIP commands are following:

```
// message: 'conectar' , to conect webphone on the SIP server
// message: 'desconectar' ,  to disconnect webphone from the SIP server
// message: 'answer' ,  to answer a new call ringing on webphone
// message: 'hangup' , to hangupcall
// message: 'mute' , to mute microphone
 function sendCmd(message) {
    webphone.contentWindow.postMessage({
        message: message
    }, '*');
}


// message: 'enviaDTMF' ,  to send a DTMF to PSTN/IVR and dtmf: 
(0,1,2,3,4,5,6,7,8,9,*,#)
function sendDtmf(dtmf) {
    webphone.contentWindow.postMessage({
        message: 'enviaDTMF',
        'dtmf': dtmf
    }, '*');
}

// message: 'chamaNumero' , to start a new call to a given number - should send 
destination
function makeCall(phoneNumber) {
    webphone.contentWindow.postMessage({
        message: 'chamaNumero',
        'numero': phoneNumber
    }, '*');
}
```

```
// message: 'transferir' , to transfer the current call to another number - should
send destination
// message: 'transferirConsulta' ,  to transfer the current call to a number after
the destination answer and talk - should send destination
function transferCall(message, destExt) {
    webphone.contentWindow.postMessage({
        message: message,
        'numeroTelefone': destExt
    }, '*');
}
```

See attached a demo page: