Documentation MyPeerConnection.js

1. Introduction

La librairie MyPeerConnection.js est une librairie qui a pour objectif de simplifier l'utilisation de WebRTC. elle permet de

- Faire du partage de webcam
- Faire du partage d'écran
- Envoyer des données

Elle est compatible avec les navigateurs suivants

- Chrome 36 (partage de webcam, canal de données, émission et réception pour le partage d'écran)
- Opera 23(partage de webcam, canal de données, réception pour le partage d'écran)
- Firefox 30(Partage de webcam, canal de données)

MyPeerConnection.js permet de résoudre certains problèmes d'interopérabilité entre les navigateurs comme la renégociation qui n'est pas supporté par Firefox quand il communique avec Chrome ou Opera

2. Méthodes publiques

2.1. initialize

2.1.1. Description

La méthode initialize doit **impérativement** être appelée avant de lancer un partage quelconque. Elle va permettre de stocker les serveurs ICE et leurs options, le pointeur sur fonction qui permet la communication client-serveur ainsi que deux autres pointeurs sur fonction pour traiter les déconnexions

2.1.2. Paramètres

- listServers : Tableau des serveurs ICE utilisés pour la création d'une RTCPeerConnection.
- optionsPeerConnection : Tableau des options utilisées pour la création d'une RTCPeerConnection.
- sendMessage : Pointeur sur fonction utilisé pour communiquer avec le serveur. Doit prendre en charge deux paramètres (type et message) et les envoyer au serveur. **Paramètre obligatoire**.
- ptrMyDisconnect : Pointeur sur fonction utilisé lorsque MyPeerConnection.js souhaite me déconnecter.
- ptrOtherDisconnect : Pointeur sur fonction utilisé lorsque l'interlocuteur se déconnecte.
- onError : Pointeur sur fonction utilisé si une erreur est détectée pendant l'initialisation.

2.1.3. Retour

Aucun

2.2. Méthodes relative au partage de la webcam

2.2.1. startWebcamSharing

2.2.1.1. Description

La méthode startWebcamSharing permet de lancer le processus de partage de la webcam de l'utilisateur courant vers l'interlocuteur. Cela comprend une demande d'acceptation pour effectuer le partage, la création d'un objet RTCPeerConnection et les requêtes de pour assurer le partage de part et d'autre.

2.2.1.2. Paramètres

- onSuccess : Pointeur sur fonction qui est appelé par l'utilisateur courant lorsque son interlocuteur à bien reçu le partage de webcam.
- onError : Pointeur sur fonction qui est appelé si un problème est survenu lors de la tentative de partage de webcam.

2.2.1.3. Retour

Aucun

2.2.2. stopWebcamSharing

2.2.2.1. Description

La méthode stopWebcamSharing permet d'arrêter le partage de l'utilisateur courant vers son interlocuteur

2.2.2.2. Paramètres

Aucun

2.2.2.3. Retour

Aucun

2.2.3. beforeReceiveWebcam

2.2.3.1. Description

La méthode beforeReceiveWebcam permet d'appeler un pointeur sur fonction pour définir une action lorsque l'interlocuteur souhaite partager sa webcam avec l'utilisateur courant. Le pointeur sur fonction passé en paramètre doit retourner true si l'utilisateur courant accepte de recevoir la webcam de l'interlocuteur, false sinon. **Retourne true par défaut**.

2.2.3.2. Paramètres

- ptrFunction : Pointeur sur fonction appelé lorsque l'utilisateur courant reçoit une demande de réception de la webcam de l'interlocuteur

2.2.3.3. Retour

Aucun

2.2.4. setOnReceiveWebcam

2.2.4.1. Description

La méthode setOnReceiveWebcam permet de définir deux pointeurs sur fonction. Le premier sera appelé lorsque l'utilisateur courant aura reçu la webcam de son interlocuteur avec succès, le second pointeur sera appelé si une erreur est survenue lors des échanges

2.2.4.2. Paramètres

- onSuccess : Pointeur sur fonction qui est appelé lorsque l'utilisateur courant reçoit la webcam de son interlocuteur
- onError : Pointeur sur fonction qui est appelé lorsqu'une erreur empêche l'utilisateur courant de recevoir la webcam de son interlocuteur

2.2.4.3. Retour

Aucun

2.3. Méthodes relative au partage d'écran

2.3.1. startScreenSharing

2.3.1.1. Description

La méthode startScreenSharing permet de lancer le processus de partage d'écran de l'utilisateur courant vers l'interlocuteur. Cela comprend une demande d'acceptation pour effectuer le partage, la création d'un objet RTCPeerConnection et les requêtes de pour assurer le partage de part et d'autre. Fonctionne uniquement sur Chrome 36 et quelques versions antérieures. Chrome 37 nécessite apparemment la création d'un plugin Chrome pour faire fonctionner le screen sharing.

2.3.1.2. Paramètres

- onSuccess : Pointeur sur fonction qui est appelé par l'utilisateur courant lorsque son interlocuteur à bien reçu le partage d'écran.
- onError : Pointeur sur fonction qui est appelé si un problème est survenu lors de la tentative de partage d'écran.

2.3.1.3. Retour

Aucun

2.3.2. stopScreenSharing

2.3.2.1. Description

La méthode stopScreenSharing permet d'arrêter le partage de l'utilisateur courant vers son interlocuteur

2.3.2.2. Paramètres

Aucun

2.3.2.3. Retour

Aucun

2.3.3. beforeReceiveScreen

2.3.3.1. Description

La méthode beforeReceiveScreen permet d'appeler un pointeur sur fonction pour définir une action lorsque l'interlocuteur souhaite partager son écran avec l'utilisateur courant. Le pointeur sur fonction passé en paramètre doit retourner true si l'utilisateur courant accepte de recevoir l'écran de l'interlocuteur, false sinon. **Retourne true par défaut**.

2.3.3.2. Paramètres

- ptrFunction : Pointeur sur fonction appelé lorsque l'utilisateur courant reçoit une demande de réception concernant l'écran de l'interlocuteur

2.3.3.3. Retour

Aucun

2.3.4. setOnReceiveScreen

2.3.4.1. Description

La méthode setOnReceiveScreen permet de définir deux pointeurs sur fonction. Le premier sera appelé lorsque l'utilisateur courant aura reçu l'écran de son interlocuteur avec succès, le second pointeur sera appelé si une erreur est survenue lors des échanges

2.3.4.2. Paramètres

- onSuccess : Pointeur sur fonction qui est appelé lorsque l'utilisateur courant reçoit l'écran de son interlocuteur
- onError : Pointeur sur fonction qui est appelé lorsqu'une erreur empêche l'utilisateur courant de recevoir l'écran de son interlocuteur

2.3.4.3. Retour

Aucun

2.4. Méthodes relative au data channel

2.4.1. createDataChannel

2.4.1.1. Description

La méthode createDataChannel est utilisée pour préparer un canal de données entre l'utilisateur courant et son interlocuteur**. La création du canal de données fonctionne uniquement si l'utilisateur courant et son interlocuteur ont appelés la méthode createDataChannel**. Quand un seul des deux participants a appelé la méthode createDataChannel, il est mit en attente pour la création du canal de données.

2.4.1.2. Paramètres

- ptrOnMessageChannel : Pointeur sur fonction utilisé pour traiter un message reçu de l'interlocuteur. **Paramètre obligatoire.**
- ptrOnOpenChannel : Pointeur sur fonction appelé lorsque le canal de données est ouvert.
- ptrOnCloseChannel : Pointeur sur fonction appelé lorsque le canal de données est fermé.
- ptrOnErrorChannel : Pointeur sur fonction pour traiter les erreurs qui apparaissent entre l'ouverture et la fermeture du canal de données.

2.4.1.3. Retour

Aucun

2.4.2. setOnReceiveDataChannel

2.4.2.1. Description

La méthode setOnReceiveDataChannel permet de définir deux pointeurs sur fonction. Le premier sera appelé lorsque l'interlocuteur aura établi un canal de données avec l'utilisateur courant, le second pointeur sera appelé si une erreur est survenue lors des échanges

2.4.2.2. Paramètres

- onSuccess : Pointeur sur fonction qui est appelé lorsque l'utilisateur courant reçoit une connexion pair-à-pair de données de la part de son interlocuteur

- onError : Pointeur sur fonction qui est appelé si un problème est survenu lors de la tentative de création d'un canal de données

2.4.2.3. Retour

Aucun

2.4.3. closeDataChannel

2.4.3.1. Description

La méthode closeDataChannel permet de couper le canal de données entre l'utilisateur courant et l'interlocuteur.

2.4.3.2. Paramètres

Aucun

2.4.3.3. Retour

Aucun

2.4.4. sendDataChannel

2.4.4.1. Description

La méthode sendDataChannel permet à l'utilisateur courant d'envoyer des données de toute sorte à travers le canal de données. Ces données seront réceptionnées par l'interlocuteur grâce au pointeur sur fonction ptrOnMessageChannel défini dans la méthode createDataChannel.

2.4.4.2. Paramètres

- message : Les données à envoyer.

2.4.4.3. Retour

Aucun

2.5. Méthodes relatives aux interactions client-serveur

2.5.1. receiveMessage

2.5.1.1. Description

La méthode receiveMessage traite toutes les données qui sont envoyées automatiquement par la librairie.

2.5.1.2. Paramètres

- type : Le premier paramètre envoyé par la méthode sendMessage
- message : Le second paramètre envoyé par la méthode sendMessage

2.5.1.3. Retour

Aucun

2.5.2. setSendMessage

2.5.2.1. Description

La méthode setSendMessage permet de changer le pointeur sur fonction établi par le paramètre sendMessage dans la méthode initialize. Il est utilisé pour communiquer avec le serveur.

2.5.2.2. Paramètres

- ptrFunction : Pointeur sur fonction utilisé pour communiquer avec le serveur. Doit prendre en charge deux paramètres (type et message) et les envoyer au serveur.

2.5.2.3. Retour

Aucun

2.6. Méthodes diverses

2.6.1. setAllVideosElements

2.6.1.1. Description

La méthode setAllVideosElements permet à la librairie de connaître les différents éléments vidéo au sens HTML DOM. Cela permettra ensuite de d'afficher les flux vidéo (webcam de l'utilisateur courant, webcam de l'interlocuteur, l'écran de l'utilisateur courant et le l'écran de l'interlocuteur) dans les bonne balises HTML vidéos.

2.6.1.2. Paramètres

- myWebcamElement : Le nœud HTML DOM qui prend en charge l'affichage de la webcam pour l'utilisateur courant.
- yourWebcamElement : Le nœud HTML DOM qui prend en charge l'affichage de la webcam pour l'interlocuteur.
- myScreenElement : Le nœud HTML DOM qui prend en charge l'affichage de l'écran pour l'utilisateur courant.
- yourScreenElement : Le nœud HTML DOM qui prend en charge l'affichage de l'écran pour l'interlocuteur.

2.6.1.3. Retour

Aucun

2.6.2. getMyBrowser

2.6.2.1. Description

La méthode getMyBrowser permet de récupérer le nom du navigateur de l'utilisateur courant.

2.6.2.2. Paramètres

Aucun

2.6.2.3. Retour

Chaine de caractères : Le nom du navigateur Web courant.

2.6.3. getInterlocutorBrowser

2.6.3.1. Description

La méthode getInterlocutorBrowser permet de récupérer le nom du navigateur de l'interlocuteur.

2.6.3.2. Paramètres

Aucun

2.6.3.3. Retour

Chaine de caractères : Le nom du navigateur Web de l'interlocuteur.

3. Suggestion pour continuer le développement

- Certains commentaires dans le code comme « // /!\ Verifier MyPeerConnection.initialize() ? » sont des points soulevés sur le tard qui n'ont pas eu le temps d'être vérifiés.
- De la même manière, du code comme « console.log('on open channel. Use MyPeerConnection.setOpenChannel(*fonction) for redefine.') ; » (ligne 593 dans setOnopenchannel) ou il y a une référence à une fonction qui n'existe pas.
- La version Chrome 37 n'autorise plus le partage d'écran comme il a été fait dans MyPeerConnection.js. Il semble que Chrome a besoin d'un plugin par site Web pour faire fonctionner le partage d'écran
- Dominique et Marie avaient évoqué la possibilité d'avoir plusieurs « data channel » créable par la librairie
- Implémenter MyPeerConnection.js de façon à accueillir plus de deux participants dans une conversation

4. GitHubs

Le travail qui a été fait est disponible sur les deux GitHub suivant

- https://github.com/W3DevCampus/moodle-mod_webrtc -> Intégration rapide de MyPeerConnection.js dans un plugin Moodle
- https://github.com/CedricDum/WebRTC -> Utilisation avec un serveur Node.js et un serveur WebSocket en PHP.