**IFrame API**

You can use the Jitsi Meet API to embed Jitsi Meet in to your application. You are also welcome to use it for embedding the globally distributed and highly available deployment on meet.jit.si itself. The only thing we ask for in that case is that you please DO NOT remove the jitsi.org logo from the top left corner.

**Installation**

To embed Jitsi Meet in your application you need to add the Jitsi Meet API library:

meet.jit.si:

<script src='https://meet.jit.si/external\_api.js'></script>

**API**

**api = new JitsiMeetExternalAPI(domain, options)**

The next step for embedding Jitsi Meet is to create the Jitsi Meet API object. Its constructor takes a number of options:

* **domain**: domain used to build the conference URL, 'meet.jit.si' for example.
* **options**: object with properties - the optional arguments:
  + **roomName**: (optional) name of the room to join.
  + **parentNode**: (optional) HTML DOM Element where the iframe will be added as a child.
  + **configOverwrite**: (optional) JS object with overrides for options
  + **invitees**: (optional) Array of objects containing information about new participants that will be invited in the call.
  + **userInfo**: (optional) JS object containing information about the participant opening the meeting, such as email.

Example:

**const** domain = 'meet.jit.si';

**const** options = {

roomName: 'JitsiMeetAPIExample',

width: 700,

height: 700,

parentNode: document.querySelector('#meet')

};

**const** api = **new** JitsiMeetExternalAPI(domain, options);

You can set the userInfo (email, display name) for the call:

**var** domain = "meet.jit.si";

**var** options = {

...

userInfo: {

email: 'email@jitsiexamplemail.com',

displayName: 'John Doe'

}

}

**var** api = **new** JitsiMeetExternalAPI(domain, options);

**Controlling the embedded Jitsi Meet Conference**

* **getParticipantsInfo** - Returns an array containing participants information like participant id, display name, avatar URL and email.

api.getParticipantsInfo();

* **pinParticipant** - Elects the participant with the given id to be the pinned participant in order to always receive video for this participant (even when last n is enabled).

api.pinParticipant(participantId);

You can control the embedded Jitsi Meet conference by calling executeCommand on the JitsiMeetExternalAPI object:

api.executeCommand(command, ...arguments);

The command parameter is a string containing the name of the command. The following commands are currently supported:

* **displayName** - Sets the display name of the local participant. This command requires one argument - the new display name to be set.

api.executeCommand('displayName', 'New Nickname');

* **password** - Sets the password for the room. This command requires one argument - the password name to be set.

// set new password for channel

api.addEventListener('participantRoleChanged', **function**(event) {

**if** (event.role === "moderator") {

api.executeCommand('password', 'The Password');

}

});

// join a protected channel

api.on('passwordRequired', **function** ()

{

api.executeCommand('password', 'The Password');

});

* **hangup** - Hangups the call. No arguments are required.

api.executeCommand('hangup');

mode: string //recording mode to stop, `stream` or `file`

);

You can also execute multiple commands using the executeCommands method:

api.executeCommands(commands);

The commands parameter is an object with the names of the commands as keys and the arguments for the commands as values:

api.executeCommands({

displayName: [ 'nickname' ],

toggleAudio: []

});

* **passwordRequired** - event notifications fired when failing to join a room because it has a password.
* **videoConferenceJoined** - event notifications fired when the local user has joined the video conference. The listener will receive an object with the following structure:

{

roomName: string, // the room name of the conference

id: string, // the id of the local participant

displayName: string, // the display name of the local participant

avatarURL: string // the avatar URL of the local participant

}

* **videoConferenceLeft** - event notifications fired when the local user has left the video conference. The listener will receive an object with the following structure:

{

roomName: string // the room name of the conference

}

You can get the email of a participant in the conference with the following API function:

**const** email = api.getEmail(participantId);

You can get the iframe HTML element where Jitsi Meet is loaded with the following API function:

**const** iframe = api.getIFrame();

You can invite new participants to the call with the following API function:

api.invite([ {...}, {...}, {...} ]).then(() => {

// success

}).catch(() => {

// failure

});

**NOTE: The format of the invitees in the array depends on the invite service used for the deployment.**

You can remove the embedded Jitsi Meet Conference with the following API function:

api.dispose();

NOTE: It's a good practice to remove the conference before the page is unloaded.

*Last updated on 11/13/2020*

**Aussi: https://github.com/jitsi/jitsi-meet/blob/master/config.js**

**Aussi: https://github.com/jitsi/jitsi-meet/blob/master/interface\_config.js**