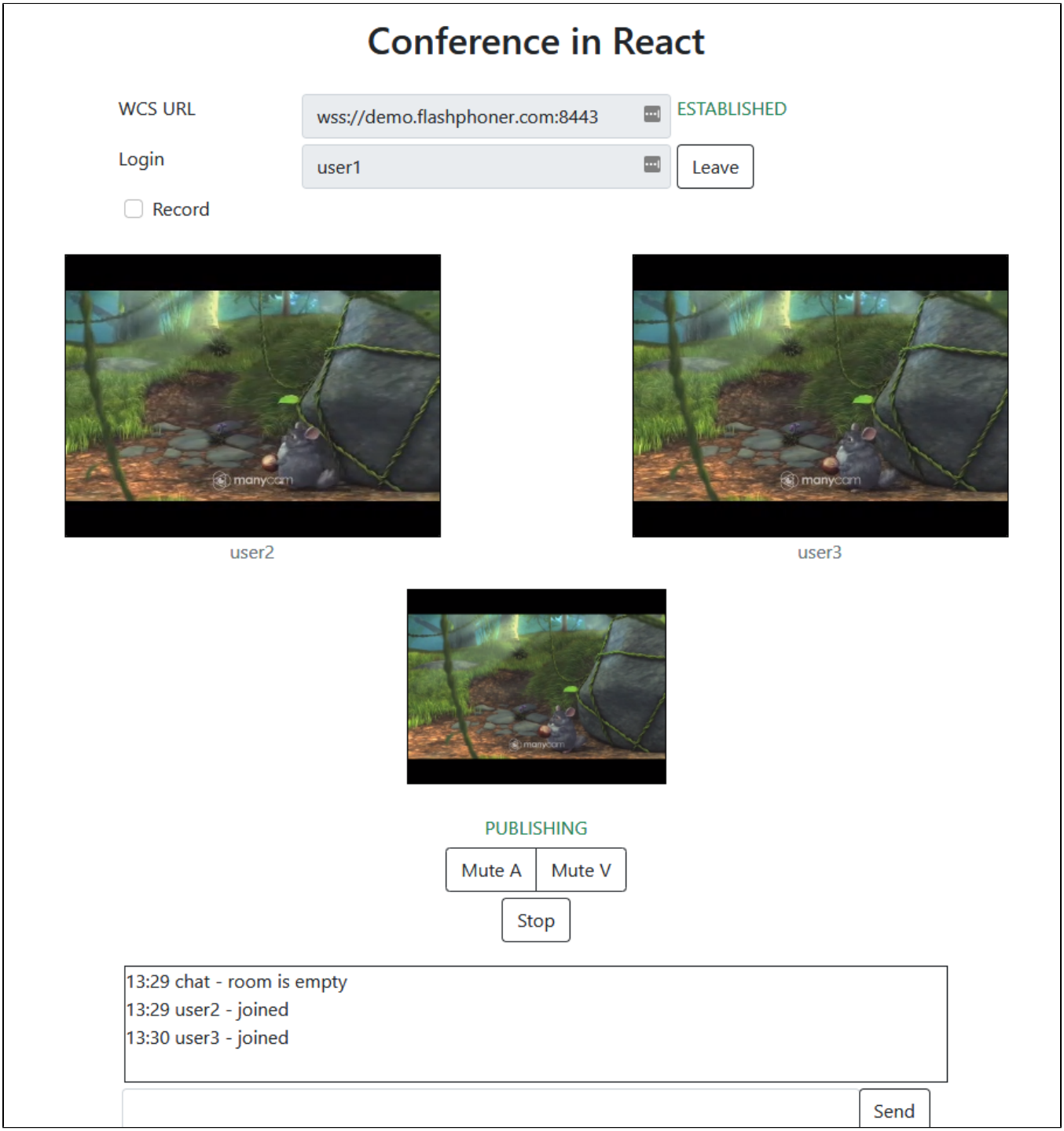
Conference React

- Overview
- Building the project
- Analyzing example code

Overview

Conference React application shows how to use Web SDK and RoomApi in React application to publish and play WebRTC streams in chat room



The project is available on GitHub, inWeb SDK build archivessince build2.0.201by the following path examples/react/conference-react.

Building the project

1. Download WebSDK source code

```
git clone https://github.com/flashphoner/flashphoner_client.git
```

2. Go to the example folder

```
cd flashphoner_client/examples/react/conference-react
```

3. Install dependencies

```
npm install
```

4. Build for local testing

```
npm start
```

or to deploy to your web server

```
npm run build
```

Analyzing example code

To analyze the code take version with hash456b1c7 which is available here and in Web SDK build2.0.201

Application code is in ConferenceApp.js file, additional functions are infp-utils.js file

1. API import

code

```
import * as Flashphoner from '@flashphoner/websdk/src/flashphoner-core.js';
import * as RoomApi from '@flashphoner/websdk/src/room-module.js';
import * as FPUtils from './fp-utils.js';
```

2. API initialization

Flashphoner.init()code

```
componentDidMount() {
  try {
    Flashphoner.init({});
    ...
  }
  catch(e) {
    console.log(e);
    ...
  }
}
```

3. Connecting to the server and receving the event confirming connection is established successfully

RoomApi.connect(), SESSION_STATUS.ESTABLISHED code

```
  createConnection = (url, username) => {
    let app = this;
    let session = this.state.session;

    if (session && session.status() === SESSION_STATUS.ESTABLISHED) {
      ...
    } else {
      console.log("Create new RoomApi session with url " + url + ", login " + username);
      app.setState({joinButtonDisabled: true, serverUrlDisabled: true});
      RoomApi.connect({urlServer: url, username: username}).on(SESSION_STATUS.ESTABLISHED, (session) => {
        app.setState({session: session, sessionStatus: SESSION_STATUS.ESTABLISHED, sessionStatusClass: 'text-
success'});
        app.joinRoom(session);
      }).on(SESSION_STATUS.DISCONNECTED, () => {
        ...
      }).on(SESSION_STATUS.FAILED, () => {
        ...
      });
    }
  }
```

4. Enter participant to the room, receiving room state

connection.join(), ROOM_EVENT.STATE code

The following parameters are passed to connection.join() function:

- room name
- should room streams be written to multiple recorder file

```
  joinRoom = (session) => {
    let app = this;
    let roomName = this.getRoomName();
    let record = this.state.record;

    console.log("Join the room " + roomName + ", record " + record);
    session.join({name: roomName, record: record}).on(ROOM_EVENT.STATE, (room) => {
      let roomParticipants = room.getParticipants();
      let participantsNumber = roomParticipants.length;
      console.log("Current number of participants in the room: " + participantsNumber);
      if (roomParticipants.length >= maxParticipants) {
          console.warn("Current room is full");
          app.setState({sessionFailedInfo: "Current room is full"});
          room.leave().then(() => {app.onLeft();}, () => {app.onLeft();});
          return false;
      }
      app.setInviteUrl(roomName);
      if (participantsNumber > 0) {
        let chatState = "participants: ";
        for (let i = 0; i < participantsNumber; i++) {
            app.installParticipant(roomParticipants[i]);
            chatState += roomParticipants[i].name();
            if (i < participantsNumber - 1) {
                chatState += ",";
            }
        }
        app.addMessage("chat", chatState);
      } else {
        app.addMessage("chat", " room is empty");
      }
      ...
      app.publishLocalMedia(room);
      app.onJoined(room);
      ...
    });
  }
```

5. Publishing stream to the room

Room.publish() code

```
publishLocalMedia = (room) => {
  let app = this;
  let constraints = {
      audio: true,
      video: true
  };
  let display = document.getElementById("localDisplay");

  app.setState({publishButtonDisabled: true});
  room.publish({
      display: display,
      constraints: constraints,
      record: false,
      receiveVideo: false,
      receiveAudio: false
      ...
  });
}
```

6, Receiving the event confirming the stream is successfully published

STREAM_STATUS.PUBLISHING code

```
publishLocalMedia = (room) => {
  ...
  room.publish({
      display: display,
      constraints: constraints,
      record: false,
      receiveVideo: false,
      receiveAudio: false
  }).on(STREAM_STATUS.FAILED, (stream) => {
      ...
  }).on(STREAM_STATUS.PUBLISHING, (stream) => {
      app.setState({publishStatus: STREAM_STATUS.PUBLISHING, publishStatusClass: 'text-success'});
      app.onMediaPublished(stream);
  }).on(STREAM_STATUS.UNPUBLISHED, (stream) => {
      ...
  });
}
```

7. Receiving the event about a participant joining the room

ROOM_EVENT.JOINED code

```
joinRoom = (session) => {
  let app = this;
  let roomName = this.getRoomName();
  let record = this.state.record;

  console.log("Join the room " + roomName + ", record " + record);
  session.join({name: roomName, record: record}).on(ROOM_EVENT.STATE, (room) => {
    ...
  }).on(ROOM_EVENT.JOINED, (participant) => {
    app.installParticipant(participant);
    app.addMessage(participant.name(), "joined");
  }).on(ROOM_EVENT.LEFT, function(participant) {
    ...
  }).on(ROOM_EVENT.PUBLISHED, (participant) => {
    ...
  }).on(ROOM_EVENT.FAILED, (room, info) => {
    ...
  }).on(ROOM_EVENT.MESSAGE, (message) => {
    ...
  });
}
```

8. Receiving the event about a participant publishing a stream

ROOM_EVENT.PUBLISHED code

```
joinRoom = (session) => {
  let app = this;
  let roomName = this.getRoomName();
  let record = this.state.record;

  console.log("Join the room " + roomName + ", record " + record);
  session.join({name: roomName, record: record}).on(ROOM_EVENT.STATE, (room) => {
    ...
  }).on(ROOM_EVENT.JOINED, (participant) => {
    ...
  }).on(ROOM_EVENT.LEFT, function(participant) {
    ...
  }).on(ROOM_EVENT.PUBLISHED, (participant) => {
    app.playParticipantsStream(participant);
  }).on(ROOM_EVENT.FAILED, (room, info) => {
    ...
  }).on(ROOM_EVENT.MESSAGE, (message) => {
    ...
  });
}
```

9. Playing participant stream with resizing the stream picture to div

Stream.play(), STREAM_STATUS_PLAYING, FPUtils.resizeVideo() code

```
playStream = (stream, remoteVideo, name) => {
  let app = this;
  let participantStream = null;

  participantStream = stream.play(remoteVideo).on(STREAM_STATUS.PLAYING, (playingStream) => {
    let video = document.getElementById(playingStream.id());
    if (video) {
      video.addEventListener('resize', (event) => {
        FPUtils.resizeVideo(event.target);
      });
    }
  });
  app.setParticipantStream(name, participantStream);
}
```

10. Sending a text message to all the participants

Participant.sendMessage() code

```
onSendClick = () => {
  let session = this.state.session;
  let room = this.state.room;
  let message = this.state.message;

  if (session && room) {
    let participants = room.getParticipants();
    this.addMessage(session.username(), message);
    for (let i = 0; i < participants.length; i++) {
        participants[i].sendMessage(encodeURIComponent(message));
    }
    this.setState({message: ''});
  }
}
```

11. Receiving text message from other participant

ROOM_EVENT.MESSAGE code

```
joinRoom = (session) => {
  let app = this;
  let roomName = this.getRoomName();
  let record = this.state.record;

  console.log("Join the room " + roomName + ", record " + record);
  session.join({name: roomName, record: record}).on(ROOM_EVENT.STATE, (room) => {
    ...
  }).on(ROOM_EVENT.JOINED, (participant) => {
    ...
  }).on(ROOM_EVENT.LEFT, function(participant) {
    ...
  }).on(ROOM_EVENT.PUBLISHED, (participant) => {
    ...
  }).on(ROOM_EVENT.FAILED, (room, info) => {
    ...
  }).on(ROOM_EVENT.MESSAGE, (message) => {
    if (message.from && message.text) {
      app.addMessage(message.from.name(), decodeURIComponent(message.text));
    }
  });
}
```

12. Receiving the event about participant is left the room

ROOM_EVENT.LEFT code

```
joinRoom = (session) => {
  let app = this;
  let roomName = this.getRoomName();
  let record = this.state.record;

  console.log("Join the room " + roomName + ", record " + record);
  session.join({name: roomName, record: record}).on(ROOM_EVENT.STATE, (room) => {
    ...
  }).on(ROOM_EVENT.JOINED, (participant) => {
    ...
  }).on(ROOM_EVENT.LEFT, function(participant) {
    app.removeParticipant(participant);
    app.addMessage(participant.name(), "left");
  }).on(ROOM_EVENT.PUBLISHED, (participant) => {
    ...
  }).on(ROOM_EVENT.FAILED, (room, info) => {
    ...
  }).on(ROOM_EVENT.MESSAGE, (message) => {
    ...
  });
}
```

13. Stopping stream publishing

Stream.stop() code

```
onPublishClick = () => {
  let stream = this.state.publishStream;
  let room = this.state.room;

  if (!room) return;
  this.setState({publishButtonDisabled: true});
  if (!stream) {
    this.publishLocalMedia(room);
  } else {
    stream.stop();
  }
};
```

14. Leaving the room

Room.leave() code

```
onJoinClick = () => {
  let app = this;
  ...
  let room = this.state.room;
  let participants = this.state.participants;

  if (!room) {
    ...
  } else {
    this.setState({joinButtonDisabled: true}, () => {
      participants.forEach((participant) => {
        // Stop all the playing participants streams
        app.stopParticipantStream(participant.stream);
      });
      room.leave().then(() => {app.onLeft();}, () => {app.onLeft();});
    });
  }
};
```