

# Conference React

## Описание


Приложение Conference React показывает пример использования Web SDK и RoomApi в React приложении для публикации и проигрывания WebRTC потоков в чат-комнате

### Conference in React


WCS URL  ESTABLISHED

Login  Leave


Record



user2



user3



PUBLISHING

Mute A Mute V

Stop

13:29 chat - room is empty  
13:29 user2 - joined  
13:30 user3 - joined

Send

Проект доступен на [GitHub](#), а также в [архивах сборок Web SDK](#) по следующему пути `examples/react/conference-react`, начиная со сборки `2.0.201`.

## Сборка проекта

1. Загрузите исходные тексты WebSDK

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

2. Перейдите в каталог примера

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

3. Установите зависимости

```
npm install
```

4. Запустите сборку для локального тестирования

```
npm start
```

или для выгрузки на свой веб-сервер

```
npm run build
```

## Работа с кодом примера

Для разбора исходного кода примера возьмем версию с хэшем `456b1c7`, которая доступна [здесь](#)

Код приложения расположен в файле `ConferenceApp.js`, дополнительные функции в файле `fp-utils.js`

### 1. Импорт API

[code](#)

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

### 2. Инициализация API

`Flashphoner.init()` [code](#)

```

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

```

### 3. Установка соединения с сервером, получение события, подтверждающего установку соединения

`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. Вход участника в комнату, получение текущего состояния комнаты

`Session.join()`, `ROOM_EVENT.STATE` [code](#)

Функции `join()` передаются параметры:

- имя комнаты
- признак записи потоков в комнате в один файл

```

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.sendMessage("chat", chatState);
  } else {
    app.sendMessage("chat", " room is empty");
  }
  ...
  app.publishLocalMedia(room);
  app.onJoined(room);
  ...
});
}

```

## 5. Публикация потока в комнате

`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. Получение сообщения, подтверждающего публикацию

`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. Получение события о присоединении участника к комнате

`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. Получение события о публикации потока участником

`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. Воспроизведение потока участника, изменение размеров картинки под `div` элемент

`Stream.play()`, `STREAM_STATUS_PLAYING`, `FPUtills.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) => {  
        FPUtills.resizeVideo(event.target);  
      });  
    }  
  });  
  app.setParticipantStream(name, participantStream);  
}
```

## 10. Отправка сообщения участникам

`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. Получение сообщения от другого участника

`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. Получение сообщения о выходе другого участника из комнаты

`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. Остановка публикации потока

`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. Выход участника из комнаты

`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();});
});
}
};
```