

# main.js - основная логика приложения

Данный модуль описывает основную логику приложения

## 1. Локальные переменные

Объявление локальных переменных для работы с константами, SFU SDK, для отображения локального видео и работы с конфигурацией клиента

[code](#)

```
const constants = SFU.constants;
const sfu = SFU;
let localDisplay;
let cControls;
```

## 2. Конфигурация по умолчанию

Объявление конфигурации комнаты и публикации потоков по умолчанию, на случай, если нет файла конфигурации `config.json`

[code](#)

Клиент настраивается на соединение с сервером по WSS по адресу localhost для входа в комнату `ROOM1` с пин-кодом `1234` под именем `Alice`. Секция media задает публикацию аудио и видео дорожек. Видео публикуется двумя дорожками с качествами high (h) и medium (m)

```
const defaultConfig = {
  room: {
    url: "wss://127.0.0.1:8888",
    name: "ROOM1",
    pin: "1234",
    nickName: "Alice"
  },
  media: {
    audio: {
      tracks: [
        {
          source: "mic",
          channels: 1
        }
      ]
    },
    video: {
```

```

        tracks: [
            {
                source: "camera",
                width: 1280,
                height: 720,
                codec: "H264",
                encodings: [
                    { rid: "h", active: true, maxBitrate: 900000 },
                    { rid: "m", active: true, maxBitrate: 300000,
                scaleResolutionDownBy: 2 }
                ]
            }
        ]
    }
}
};

```

### 3. Инициализация

`init()` code

Функция `init()` вызывается после того, как страница загрузится. Функция загружает `config.json` или конфигурацию по умолчанию, создает элемент для отображения локального видео и открывает модальное окно входа

```

/**
 * load config and show entrance modal
 */
const init = function() {
    //read config
    $.getJSON("config.json", function(config){
        cControls = createControls(config);
    }).fail(function(){
        //use default config
        cControls = createControls(defaultConfig);
    });
    //create local display to show local streams
    localDisplay = initLocalDisplay(document.getElementById("localDisplay"));
    //open entrance modal
    $('#entranceModal').modal('show');
}

```

### 4. Соединение с сервером и создание либо вход в комнату

`connect()` code

Функция вызывается по щелчку пользователя по кнопке `Enter` в модальном окне входа



## connect



Скрытие модального окна входа и отключение полей ввода до установки соединения с сервером

code

```
//hide modal
$('#entranceModal').modal('hide');
//disable controls
cControls.muteInput();
```

Создание объекта `PeerConnection` и подготовка объекта конфигурации комнаты

code

```
//create peer connection
const pc = new RTCPeerConnection();
//get config object for room creation
const roomConfig = cControls.roomConfig();
roomConfig.pc = pc;
```

Создание сессии и установка соединения с сервером

code

```
const session = sfu.createRoom(roomConfig);
```

Подписка на событие сессии `CONNECTED`

code

```
session.on(constants.SFU_EVENT.CONNECTED, function(room) {
  ...
});
```

Инициализация чата после установки соединения

code

```
//connected to server
const chatDiv = document.getElementById('messages');
const chatInput = document.getElementById('localMessage');
const chatButton = document.getElementById('sendMessage');
//create and bind chat to the new room
createChat(room, chatDiv, chatInput, chatButton);
```

Подписка на сообщения об ошибках комнаты

code

```
room.on(constants.SFU_ROOM_EVENT.FAILED, function(e) {
  const errField = document.getElementById("errorMsg");
  errField.style.color = "red";
  errField.innerText = e;
}).on(constants.SFU_ROOM_EVENT.OPERATION_FAILED, function (e) {
  const errField = document.getElementById("errorMsg");
  errField.style.color = "red";
  errField.innerText = e.operation + " failed: " + e.error;
})
```

Инициализация объекта для отображения потоков от других участников

code

```
//setup remote display for showing remote audio/video tracks
const remoteDisplay = document.getElementById("display");
initRemoteDisplay(room, remoteDisplay, pc);
```

Получение настроек публикации локального медиа

code

```
//get configured local video streams
let streams = cControls.getVideoStreams();
//combine local video streams with audio streams
streams.push.apply(streams, cControls.getAudioStreams());
```

Добавление каждого потока в объект `localDisplay` для отображения и в объект `PeerConnection` для публикации

code

```
//add our local streams to the room (to PeerConnection)
streams.forEach(function (s) {
  //add local stream to local display
  localDisplay.add(s.stream.id, "local", s.stream);
  //add each track to PeerConnection
  s.stream.getTracks().forEach((track) => {
    addTrackToPeerConnection(pc, s.stream, track, s.encodings);
    subscribeTrackToEndedEvent(room, track, pc);
  });
});
```

Добавление слушателя, чтобы определить, когда клиент добавляет новые потоки для публикации. Получив новый поток, необходимо добавить его в `localDisplay` для отображения, добавить в `PeerConnection` для публикации и обновить состояние комнаты

code

```
//add callback for the new local stream to the local controls
cControls.onTrack(function (s) {
  //add local stream to local display
  localDisplay.add(s.stream.id, "local", s.stream);
  //add each track to PeerConnection
  s.stream.getTracks().forEach((track) => {
    addTrackToPeerConnection(pc, s.stream, track, s.encodings);
    subscribeTrackToEndedEvent(room, track, pc);
  });
  //kickoff renegotiation
  room.updateState();
});
```

Вход пользователя в комнату

code

```
//join room
room.join();
```

## 5. Завершение публикации потока

`subscribeTrackToEndedEvent()` [code](#)

Вспомогательная функция, которая подписывается на событие `ended` для локального потока. При получении события поток удаляется из `PeerConnection`, и состояние комнаты обновляется.

```
const subscribeTrackToEndedEvent = function(room, track, pc) {
  track.addEventListener("ended", function() {
    //track ended, see if we need to cleanup
    let negotiate = false;
    for (const sender of pc.getSenders()) {
      if (sender.track === track) {
        pc.removeTrack(sender);
        //track found, set renegotiation flag
        negotiate = true;
        break;
      }
    }
    if (negotiate) {
      //kickoff renegotiation
      room.updateState();
    }
  });
};
```

## 7. Добавление новой дорожки в `PeerConnection`

`addTrackToPeerConnection()` [code](#)

Вспомогательная функция, которая добавляет новую дорожку в `PeerConnection` для публикации

```
const addTrackToPeerConnection = function(pc, stream, track, encodings) {  
  pc.addTransceiver(track, {  
    direction: "sendonly",  
    streams: [stream],  
    sendEncodings: encodings ? encodings : [] //passing encoding types  
    for video simulcast tracks  
  });  
}
```