# chat.js

Данный модуль содержит код для работы с чатом в комнате

## 1. Обертка для кода

`createChat()` code

Функция-обертка для вызова из основной логики, ограничивает область видимости

```
const createChat = function(room, messages, input, sendButton) {
    ...
}
```

## 2. Локальные переменные

code

Объявление локальных переменных: константы и цвета

```
const constants = SFU.constants;
const chatSelfColour = "green";
const chatTextColour = "black";
const chatOtherColour = "red";
const chatEventColour = "navy";
```

## 3. Подписка на события комнаты

code

Подписка на события комнаты, имеющие отношение к функциям чата

```
room.on(constants.SFU_ROOM_EVENT.MESSAGE, function(e) {
    appendMessage({
        userId: getUserId(e.message),
        nickName: getNickName(e.message),
        message: getMessage(e.message)
    }, chatOtherColour, chatTextColour);
}).on(constants.SFU_ROOM_EVENT.JOINED, function(e) {
    appendMessage({
        userId: getShortUserId(e.userId),
        nickName: e.name,
        message: e.type
```

```
    }, chatOtherColour, chatEventColour);
}).on(constants.SFU_ROOM_EVENT.LEFT, function(e) {
    appendMessage({
        userId: getShortUserId(e.userId),
        nickName: e.name,
        message: e.type
    }, chatOtherColour, chatEventColour);
});
```

## SFU_ROOM_EVENT.MESSAGE

Подписка на событие `SFU_ROOM_EVENT.MESSAGE`. При получении события, сообщение отображается в локальном чате

code

```
room.on(constants.SFU_ROOM_EVENT.MESSAGE, function(e) {
    appendMessage({
        userId: getUserId(e.message),
        nickName: getNickName(e.message),
        message: getMessage(e.message)
    }, chatOtherColour, chatTextColour);
    ...
});
```

## SFU_ROOM_EVENT.JOINED

Подписка на событие `SFU_ROOM_EVENT.JOINED`. При получении, сообщение о входе участника отображается в локальном чате

code

```
room.on(constants.SFU_ROOM_EVENT.MESSAGE, function(e) {
    ...
}).on(constants.SFU_ROOM_EVENT.JOINED, function(e) {
    appendMessage({
        userId: getShortUserId(e.userId),
        nickName: e.name,
        message: e.type
    }, chatOtherColour, chatEventColour);
    ...
});
```

## SFU_ROOM_EVENT.LEFT

Подписка на событие `SFU_ROOM_EVENT.LEFT`. При получении, сообщение о выходе участника отображается в локальном чате

code

```
room.on(constants.SFU_ROOM_EVENT.MESSAGE, function(e) {
    ...
}).on(constants.SFU_ROOM_EVENT.LEFT, function(e) {
    appendMessage({
        userId: getShortUserId(e.userId),
        nickName: e.name,
        message: e.type
    }, chatOtherColour, chatEventColour);
});
```

## 4. Отправка сообщения другим участникам

`Room.sendMessage()` code

Функция `sendMessage` реализует разбор пользовательского ввода, отправку сообщения серверу и отображение сообщения в локальном чате

Отметим, что сообщения отправляются по WebRTC data channels

```
const sendMessage = async function() {
    let message = input.value;
    input.value = "";
    await room.sendMessage(message);
    appendMessage({
        userId: getShortUserId(room.userId()),
        nickName: nickName.value,
        message: message
    }, chatSelfColour, chatTextColour);
}
```

Привязка кнопки `Send` к функции `sendMessage`

code

```
sendButton.addEventListener("click", sendMessage);
```

Отправка сообщения по нажатию `Enter`

code

```
input.onkeyup = function(e) {
    if (e.keyCode === 13) {
        if (e.shiftKey) {

        } else {
            sendMessage();
        }
        return false;
    }
}
```

## 5. Локальное отображение сообщений чата

Функция реализует форматирование и отображение сообщений в локальном чате

`appendMessage()` code

```
const appendMessage = function(msg, nickColour, msgColour) {
    let message = document.createElement('div');
    message.setAttribute("class","message");
    messages.appendChild(message);
    let nickDiv = document.createElement('div');
    nickDiv.style.color = nickColour;
    nickDiv.innerText = getChatTimestamp() + " " + msg.nickName + "#" +
msg.userId + ":";
    message.appendChild(nickDiv);
    let msgDiv = document.createElement('div');
    msgDiv.style.color = msgColour;
    msgDiv.innerText = msg.message;
    message.appendChild(msgDiv);
    scrollToBottom();
}
```

Вспомогательная функция для прокрутки списка сообщений вниз

`scrollToBottom()` code

```
const scrollToBottom = function() {
    messages.scrollTop = messages.scrollHeight;
}
```

Вспомогательная функция для вывода времени

`getTimeStamp()` code

```
const getChatTimestamp = function() {
    let currentdate = new Date();
    return currentdate.getHours() + ":" + currentdate.getMinutes() + ":" +
currentdate.getSeconds();
}
```

## 6. Получение идентификатора пользователя из сообщения

`getUserId()` code

```
const getUserId = function(msgData) {
    let userId = "unknown";
    if (msgData.userId) {
        userId = msgData.userId;
    } else if (msgData.message.userId) {
        userId = msgData.message.userId;
    }
```

```
    return getShortUserId(userId);
}
```

## 7. Получение имени пользователя из сообщения

`getNickName()` code

```
const getNickName = function(msgData) {
    let nickName = "unknown";
    if (msgData.nickName) {
        nickName = msgData.nickName;
    } else if (msgData.message.nickName) {
        nickName = msgData.message.nickName;
    }
    return nickName;
}
```

## 8. Получение содержимого сообщения

`getMessage()` code

```
const getMessage = function(msgData) {
    let message = "";
    if (msgData.message) {
        message = JSON.parse(msgData.message).payload;
    }
    return message;
}
```