chat.js - code to handle chat in the room

This module encapsulates code which is responsible for room chat

1. Wrapper function

createChat() code

Encapsulates chat code inside the closure

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

2. Local variables

code

Define local variables - SFU constants and colours

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

3. Subscribe to room events

code

Subscribe to room events that are related to chat functionality

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

SFU_ROOM_EVENT.MESSAGE

Subscribe to new message event. Once received display it in local chat

code

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

SFU_ROOM_EVENT.JOINED

Subscribe to new participant event. Once received display a special notification inside the local chat

code

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

SFU_ROOM_EVENT.LEFT

Subscribe to participant left event. Once received display a special notification inside the local chat

code

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

4. Send message to other participants

sendMessage() code

Define the sendMessage function which will be responsible for parsing input, sending it to server and displaying it in the local chat as a message

Note that messages is sent using WebRTC data channels

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

Tie send button to sendMessage function

code

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

Give ability to send message by pressing Enter

code

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

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

5. Display chat messages locally

Define function that will take care of formatting and displaying messages in local chat

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 + ":";
    message.appendChild(nickDiv);
    let msgDiv = document.createElement('div');
    msgDiv.style.color = msgColour;
    msgDiv.innerText = msg.message;
    message.appendChild(msgDiv);
    scrollToBottom();
}
```

Scroll to bottom helper

scrollToBottom() code

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

Timestamp helper

getTimeStamp() code

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