

controls.js - код для работы с конфигурацией клиента

Данный модуль содержит код для управления локальными медиа потоками и конфигурацией комнаты, включая создание новых дорожек, остановку существующих дорожек и подготовку конфигурации комнаты перед установкой соединения с сервером.

1. Обертка для кода

`createControls()` code

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

```
const createControls = function(config) {  
  ...  
}
```

2. Создание объекта controls

code

Создание объекта `controls`, содержащего все HTML элементы для настройки. Здесь же инициализируются таблицы для настройки публикации дорожек.

```
const controls = {  
  entrance: {  
    url: document.getElementById("url"),  
    roomName: document.getElementById("roomName"),  
    roomPin: document.getElementById("roomPin"),  
    nickName: document.getElementById("nickName"),  
    enter: document.getElementById("startButton")  
  },  
  addVideoTrack: {  
    source: document.getElementById("addVideoTrackSource"),  
    width: document.getElementById("addVideoTrackWidth"),  
    height: document.getElementById("addVideoTrackHeight"),  
    codec: document.getElementById("addVideoTrackCodec")  
  },  
  addAudioTrack: {  
    source: document.getElementById("addAudioTrackSource"),  
    channels: document.getElementById("addAudioTrackChannels")  
  },  
  addVideoEncoding: {
```

```

        rid: document.getElementById("addVideoTrackEncodingRid"),
        active: document.getElementById("addVideoTrackEncodingActive"),
        maxBitrate:
document.getElementById("addVideoTrackEncodingMaxBitrate"),
        resolutionScale:
document.getElementById("addVideoTrackEncodingResolutionScale")
    },
    tables: {
        video: $('#videoTracksTable').DataTable({
            "sDom": 't',
            "columns": [
                {
                    "className": 'details-control',
                    "orderable": false,
                    "data": null,
                    "defaultContent": ''
                },
                {"data": "source"},
                {"data": "width"},
                {"data": "height"},
                {"data": "codec"},
                {"data": "action"}
            ]
        }),
        audio: $('#audioTracksTable').DataTable({
            "sDom": 't',
            "columns": [
                {"data": "source"},
                {"data": "channels"},
                {"data": "action"}
            ]
        }),
        encodings: $('#videoTrackEncodingsTable').DataTable({
            "sDom": 't',
            "columns": [
                {"data": "rid"},
                {"data": "active"},
                {"data": "maxBitrate"},
                {"data": "resolutionScale"},
                {"data": "action"}
            ]
        })
    }
}

```

3. Заполнение полей модального окна входа

code

Заполнение полей модального окна входа в соответствии с текущей конфигурацией

```

//apply room config
controls.entrance.url.value = config.room.url;
controls.entrance.roomName.value = config.room.name;

```

```
controls.entrance.roomPin.value = config.room.pin;
controls.entrance.nickName.value = config.room.nickName;
```

4. Добавление новых аудио дорожек в объект controls

`addAudioTrackRow()` [code](#)

Добавление новых аудиодорожек и оповещение основного модуля

```
const addAudioTrackRow = function(track) {
  getMedia([track]).then(function(stream){
    let button = '<button id="' + stream.id + '-button" class="btn btn-
primary">Delete</button>';
    const row = controls.tables.audio.row.add({
      source: track.source,
      channels: track.channels,
      action: button,
      stream: stream
    }).node();
    controls.tables.audio.draw();

    $('#'+ stream.id + "-button").on('click', function(){
      //terminate stream
      console.log("terminate stream " + stream.id);
      let track = stream.getAudioTracks()[0];
      track.stop();
      track.dispatchEvent(new Event("ended"));
    });
    stream.getTracks()[0].onended = function() {
      controls.tables.audio.row(row).remove().draw();
    }
    trackCallback({
      stream: stream,
      encodings: track.encodings
    });
  });
};
```

Запрос локального медиа через WebRTC API

[code](#)

```
getMedia([track]).then(function(stream){
```

Добавление аудио дорожки в таблицу дорожек

[code](#)

```
let button = '<button id="' + stream.id + '-button" class="btn btn-
primary">Delete</button>';
const row = controls.tables.audio.row.add({
  source: track.source,
```

```
channels: track.channels,
action: button,
stream: stream
}).node();
controls.tables.audio.draw();
```

Подписка на событие `click`. По нажатию кнопки `Delete` дорожка останавливается, генерируется событие `ended`

code

```
$('#' + stream.id + "-button").on('click', function(){
  //terminate stream
  console.log("terminate stream " + stream.id);
  let track = stream.getAudioTracks()[0];
  track.stop();
  track.dispatchEvent(new Event("ended"));
});
```

Подписка на событие `ended` и очистка таблицы при получении данного события

code

```
stream.getTracks()[0].onended = function() {
  controls.tables.audio.row(row).remove().draw();
}
```

Оповещение основного модуля о том, что добавлена новая дорожка

code

```
trackCallback({
  stream: stream,
  encodings: track.encodings
});
```

5. Добавление новых видео дорожек

`addVideoTrackRow()` code

Добавление новых видео дорожек и оповещение основного модуля, аналогично функции `addAudioTrackRow`.

```
const addVideoTrackRow = function(track) {
  getMedia([track]).then(function(stream){
    let button = '<button id="' + stream.id + '-button" class="btn btn-
primary">Delete</button>';
    const row = controls.tables.video.row.add({
      source: track.source,
      width: track.width,
```

```

        height: track.height,
        codec: track.codec,
        action: button,
        stream: stream,
        encodings: track.encodings
    }).node();
    controls.tables.video.draw();

    $('#' + stream.id + "-button").on('click', function(){
        //terminate stream
        console.log("terminate stream " + stream.id);
        let track = stream.getVideoTracks()[0];
        track.stop();
        track.dispatchEvent(new Event("ended"));
    });
    stream.getTracks()[0].addEventListener("ended", function() {
        controls.tables.video.row(row).remove().draw();
    });
    trackCallback({
        stream: stream,
        encodings: track.encodings
    });
});
}

```

6. Форматирование настроек кодирования видео

`format()` [code](#)

Вспомогательная функция форматирует настройки кодирования видео для их отображения в таблице

```

const format = function(d) {
    if (!d.encodings) {
        return;
    }
    let details = '<table cellpadding="5" cellspacing="0" border="0" style="padding-left:50px;">';
    d.encodings.forEach(function(encoding){
        details += '<tr>';
        for (const [key, value] of Object.entries(encoding)) {
            details += '<td>'+ key + '</td>'+
                '<td>'+ value + '</td>';
        }
        details += '</tr>';
    });
    details += '</table>';
    return details;
}

```

7. Добавление обработчика для отображения/скрытия данных дорожки

code

Добавление обработчика для того, чтобы показать или спрятать данные видео дорожки

```
// Add event listener for opening and closing details
$('#videoTracksTableBody').on('click', 'td.details-control', function () {
  let tr = $(this).closest('tr');
  let row = controls.tables.video.row(tr);
  if (row.child.isShown()) {
    // This row is already open - close it
    row.child.hide();
    tr.removeClass('shown');
  } else {
    // Open this row
    row.child(format(row.data())).show();
    tr.addClass('shown');
  }
});
```

8. Добавление аудио дорожек в таблицу

code

Добавление всех настроенных аудио дорожек в таблицу

```
config.media.audio.tracks.forEach(function(track){
  addAudioTrackRow(track);
})
```

9. Добавление видео дорожек в таблицу

code

Добавление всех настроенных видео дорожек в таблицу

```
config.media.video.tracks.forEach(function(track){
  addVideoTrackRow(track);
})
```

10. Отключение элементов формы

`muteForm()` code

Вспомогательная функция, отключающая все элементы указанной формы

```
const muteForm = function(form) {
  for (const [key, value] of Object.entries(form)) {
    value.disabled = true;
  }
}
```

11. Включение элементов формы

`unmuteForm()` [code](#)

Вспомогательная функция, включающая все элементы указанной формы

```
const unmuteForm = function(form) {
  for (const [key, value] of Object.entries(form)) {
    value.disabled = false;
  }
}
```

12. Отключение полей ввода модального окна входа

`muteInput()` [code](#)

Вспомогательная функция, отключающая поля ввода в модальном окне

```
const muteInput = function() {
  muteForm(controls.entrance);
}
```

13. Создание объекта конфигурации комнаты

`roomConfig()` [code](#)

Функция собирает объект конфигурации комнаты

```
const roomConfig = function() {
  return {
    url: controls.entrance.url.value,
    roomName: controls.entrance.roomName.value,
    pin: controls.entrance.roomPin.value,
    nickname: controls.entrance.nickName.value
  }
}
```

14. Получение локальных видео дорожек

`getVideoStreams()` [code](#)

Функция возвращает все локальные видео дорожки

```
const getVideoStreams = function() {
  let streams = [];
  controls.tables.video.rows().every(function(rowIdx, tableLoop, rowLoop) {
    let data = this.data();
    streams.push({
      stream: data.stream,
      encodings: data.encodings
    });
  });
  return streams;
}
```

15. Получение локальных аудио дорожек

`getAudioStreams()` [code](#)

Функция возвращает все локальные аудио дорожки

```
const getAudioStreams = function() {
  let streams = [];
  controls.tables.audio.rows().every(function(rowIdx, tableLoop, rowLoop) {
    let data = this.data();
    streams.push({
      stream: data.stream,
      encodings: []
    });
  });
  return streams;
}
```

16. Обработчик для добавления видео дорожки в таблицу

[code](#)

Подписка на событие `click` кнопки добавления дорожки. Функция добавляет новую дорожку в таблицу

```
document.getElementById("addVideoTrack").addEventListener("click",
function(e){
  let encodings = [];
  controls.tables.encodings.rows().every(function() {
    let encoding = this.data();
    encodings.push({
      rid: encoding.rid,
```



```

        active: encoding.active,
        maxBitrate: encoding.maxBitrate,
        scaleResolutionDownBy: encoding.resolutionScale
    })
});
let track = {
    source: controls.addVideoTrack.source.value,
    width: controls.addVideoTrack.width.value,
    height: controls.addVideoTrack.height.value,
    codec: controls.addVideoTrack.codec.value,
    encodings: encodings
}
addVideoTrackRow(track);
});

```

17. Обработчик для удаления параметров кодирования видео из таблицы

code

Удаление параметров кодирования видео из таблицы

```

$("#videoTrackEncodingsTable").on("click", ".remove", function(){
    controls.tables.encodings.row($(this).parents('tr')).remove().draw();
});

```

18. Обработчик для добавления параметров кодирования видео в таблицу

code

Добавление параметров кодирования видео в таблицу

```

document.getElementById("addVideoTrackEncoding").addEventListener("click",
function(){
    let button = '<button class="btn btn-primary remove">Delete</button>';
    controls.tables.encodings.row.add({
        rid: controls.addVideoEncoding.rid.value,
        active: controls.addVideoEncoding.active.value,
        maxBitrate: controls.addVideoEncoding.maxBitrate.value,
        resolutionScale: controls.addVideoEncoding.resolutionScale.value,
        action: button
    }).draw();
});

```

19. Обработчик для добавления видео дорожки в таблицу

code

Подписка на событие `click` кнопки добавления дорожки. Функция добавляет новую дорожку в таблицу

```
document.getElementById("addAudioTrack").addEventListener("click",
function(e){
  let encodings = [];
  let track = {
    source: controls.addAudioTrack.source.value,
    channels: controls.addAudioTrack.channels.value,
    encodings: encodings
  }
  addAudioTrackRow(track);
});
```

20. Передача callback функции новым дорожкам

code

Функция передает указанную callback функцию новым дорожкам

```
const onTrack = function(callback) {
  trackCallback = callback;
}
```

21. Экспорт функций

code

Экспорт функций для использования в основном модуле

```
return {
  muteInput: muteInput,
  roomConfig: roomConfig,
  getAudioStreams: getAudioStreams,
  getVideoStreams: getVideoStreams,
  onTrack: onTrack
}
```

22. Получение медиа потоков из WebRTC API

`getMedia()` code

Запрос списка локальных медиа потоков от WebRTC API

```
const getMedia = async function(tracks) {
  //convert to constraints
  let screen = false;
  const constraints= {};
  tracks.forEach(function(track){
    if (track.source === "mic") {
      //audio
      constraints.audio = {};
      constraints.audio.stereo = track.channels !== 1
    } else if (track.source === "camera") {
      constraints.video = {
        width: track.width,
        height: track.height
      };
    } else if (track.source === "screen") {
      constraints.video = {
        width: track.width,
        height: track.height
      };
      screen = true;
    }
  });

  //get access to a/v
  let stream;
  if (screen) {
    stream = await navigator.mediaDevices.getDisplayMedia(constraints);
  } else {
    stream = await navigator.mediaDevices.getUserMedia(constraints);
  }
  return stream;
}
```