```
// 测试
// window.onDouyinServer = function() {
//     new Barrage({ message: false })
// }

const Barrage = class {
    wsurl = "ws://127.0.0.1:9527"
    timer = null
    timeinterval = 10 * 1000 // 断线重连轮询间隔
    propsId = null
    chatDom = null
    roomJoinDom = null
    ws = null
    observer = null
    chatObserverrom = null
    option = {}
    event = {}
    eventRegirst = {}
    constructor(option = { message: true }) {
        this.option = option
        let { link, removePlay } = option
        if (link) {
            this.wsurl = link
        }
        if (removePlay) {
            document.querySelector('.basicPlayer').remove()
        }
        this.propsId = Object.keys(document.querySelector('.webcast-chatroom___list'))[1]
        this.chatDom = document.querySelector('.webcast-chatroom___items').children[0]
        this.roomJoinDom = document.querySelector('.webcast-chatroom___bottom-message')
        this.ws = new WebSocket(this.wsurl)
        this.ws.onclose = this.wsClose
        this.ws.onopen = () => {
            this.openWs()
        }
    }

    // 消息事件 , join, message
    on(e, cb) {
        this.eventRegirst[e] = true
        this.event[e] = cb
    }
    openWs() {
        console.log(`[${new Date().toLocaleTimeString()}]`, '服务已经连接成功!')
        clearInterval(this.timer)
        this.runServer()
    }
    wsClose() {
        console.log('服务器断开')
        if (this.timer !== null) {
            return
        }
        this.observer && this.observer.disconnect();
        this.chatObserverrom && this.chatObserverrom.disconnect();
        this.timer = setInterval(() => {
            console.log('正在等待服务器启动..')
            this.ws = new WebSocket(wsurl);
            console.log('状态 ->', this.ws.readyState)
            setTimeout(() => {
                if (this.ws.readyState === 1) {
                    openWs()
                }
            }, 2000)

        }, this.timeinterval)
    }
    runServer() {
        let _this = this
        if (this.option.join) {
            this.observer = new MutationObserver((mutationsList) => {
                for (let mutation of mutationsList) {
                    if (mutation.type === 'childList' && mutation.addedNodes.length) {
                        let dom = mutation.addedNodes[0]
                        let user = dom[this.propsId].children.props.message.payload.user
                        let msg = {
                            ...this.getUser(user),
                            ... { msg_content: `${user.nickname} 来了` }
                        }
                        if (this.eventRegirst.join) {
                            this.event['join'](msg)
                        }
                        this.ws.send(JSON.stringify({ action: 'join', message: msg }));
                    }
                }
            });
            this.observer.observe(this.roomJoinDom, { childList: true });

        }

        this.chatObserverrom = new MutationObserver((mutationsList, observer) => {
            for (let mutation of mutationsList) {
                if (mutation.type === 'childList' && mutation.addedNodes.length) {
                    let b = mutation.addedNodes[0]
                    if (b[this.propsId].children.props.message) {
                        let message = this.messageParse(b)
                        if (message) {
                            if (this.eventRegirst.message) {
                                this.event['join'](message)
                            }
                            if (_this.option.message === false && !message.isGift) {
                                return
                            }
                            this.ws.send(JSON.stringify({ action: 'message', message: message }));
                        }
                    }
                }
            }
        });
        this.chatObserverrom.observe(this.chatDom, { childList: true });
    }
    getUser(user) {
        if (!user) {
            return
        }
        let msg = {
            user_level: this.getLevel(user.badgeImageList, 1),
            user_fansLevel: this.getLevel(user.badgeImageList, 7),
            user_id: user.id,
            user_nickName: user.nickname,
            user_avatar: user.avatarThumb.urlList[0],
            user_gender: user.gender === 1 ? '男' : '女',
            user_isAdmin: user.userAttr.isAdmin,
            user_fansLightName: "",
            user_levelImage: ""
        }
        return msg
    }
    getLevel(arr, type) {
        if (!arr || arr.length === 0) {
            return 0
        }
        let item = arr.find(i => {
            return i.imageType === type
        })
        if (item) {
            return parseInt(item.content.level)
        } else {
            return 0
        }
    }
    messageParse(dom) {
        if (!dom[this.propsId].children.props.message) {
            return null
        }
        let msg = dom[this.propsId].children.props.message.payload
        let result = {
            repeatCount: null,
            gift_id: null,
            gift_name: null,
            gift_number: null,
            gift_image: null,
            gift_diamondCount: null,
            gift_describe: null,
        }

        result = Object.assign(result, this.getUser(msg.user))
        switch (msg.common.method) {
            case 'WebcastGiftMessage':
                console.log(msg)
                result = Object.assign(result, {
                    // repeatCount: parseInt(),
                    msg_content: msg.common.describe,
                    isGift: true,
                    gift_id: msg.gift.id,
                    gift_name: msg.gift.name,
                    // gift_number: parseInt(msg.comboCount),
                    gift_number: parseInt(msg.repeatCount),
                    gift_image: msg.gift.icon.urlListList[0],
                    gift_diamondCount: msg.gift.diamondCount,
                    gift_describe: msg.gift.describe,
                })
                break
            case 'WebcastChatMessage':
                result = Object.assign(result, {
                    isGift: false,
                    msg_content: msg.content
                })
                break
            default:
                result = Object.assign(result, {
                    isGift: false,
                    msg_content: msg.content
                })
                break
        }
        return result
    }
}

if (window.onDouyinServer) {
    window.onDouyinServer()
}

window.removeVideoLayer = function() {
    document.querySelector('.basicPlayer').remove()
    console.log('删除画面成功,不影响弹幕信息接收')
}
```