

util.js

`util.js` module contains a various helper functions

Analyzing the source code

To analyze the source code take the `util.js` module version available [here](#)

1. Exclude a video codecs from SDP

`stripCodecs()` [code](#)

```
const stripCodecs = function(sdp, codecs) {
  if (!codecs.length) return sdp;
  var sdpArray = sdp.split("\n");
  var codecsArray = codecs.split(",");

  //search and delete codecs line
  var pt = [];
  var i;
  for (var p = 0; p < codecsArray.length; p++) {
    console.log("Searching for codec " + codecsArray[p]);
    for (i = 0; i < sdpArray.length; i++) {
      if (sdpArray[i].search(new RegExp(codecsArray[p], 'i')) !== -1 &&
sdpArray[i].indexOf("a=rtpmap") === 0) {
        console.log(codecsArray[p] + " detected");
        pt.push(sdpArray[i].match(/[0-9]+/)[0]);
        sdpArray[i] = "";
      }
    }
  }
  if (pt.length) {
    //searching for fmp
    for (p = 0; p < pt.length; p++) {
      for (i = 0; i < sdpArray.length; i++) {
        if (sdpArray[i].search("a=fmp:" + pt[p]) !== -1 ||
sdpArray[i].search("a=rtcp-fb:" + pt[p]) !== -1) {
          sdpArray[i] = "";
        }
      }
    }
  }

  //delete entries from m= line
  for (i = 0; i < sdpArray.length; i++) {
    if (sdpArray[i].search("m=audio") !== -1 ||
sdpArray[i].search("m=video") !== -1) {
      var mLineSplitted = sdpArray[i].split(" ");
      var newMLine = "";
    }
  }
}
```

```

        for (var m = 0; m < mLineSplitted.length; m++) {
            if (pt.indexOf(mLineSplitted[m].trim()) === -1 || m <= 2)
            {
                newMLine += mLineSplitted[m];
                if (m < mLineSplitted.length - 1) {
                    newMLine = newMLine + " ";
                }
            }
            sdpArray[i] = newMLine;
        }
    }
}

//normalize sdp after modifications
var result = "";
for (i = 0; i < sdpArray.length; i++) {
    if (sdpArray[i] !== "") {
        result += sdpArray[i] + "\n";
    }
}
return result;
}

```

2. Exclude all the video codecs from SDP but a certain codec

`stripVideoCodecsExcept()` [code](#)

```

const stripVideoCodecsExcept = function (sdp, codec) {
    let actualStripCodec = "rtx";
    if (codec === "VP8") {
        actualStripCodec += ",H264";
    } else if (codec === "H264") {
        actualStripCodec += ",VP8";
    } else {
        return sdp;
    }
    return stripCodecs(sdp, actualStripCodec);
}

```

3. Construct a server websocket URL

`setURL()` [code](#)

```

const setURL = function () {
    var proto;
    var url;
    var port;
    if (window.location.protocol == "http:") {
        proto = "ws://";
        port = "8080";
    } else {
        proto = "wss://";
    }
}

```

```

        port = "8443";
    }

    url = proto + window.location.hostname + ":" + port;
    return url;
}

```

4. Get a parameter from URL

`getUrlParam()` [code](#)

```

const getUrlParam = function (name) {
    var url = window.location.href;
    name = name.replace(/\[\]\]/g, "\\$&");
    var regex = new RegExp("[?&]" + name + "([^\&#]*)|&|#|$", "");
    results = regex.exec(url);
    if (!results) return null;
    if (!results[2]) return '';
    return decodeURIComponent(results[2].replace(/\+/g, " "));
}

```

5. Generate an UUID

`createUUID()` [code](#)

```

const createUUID = function (length) {
    var s = [];
    var hexDigits = "0123456789abcdef";
    for (var i = 0; i < 36; i++) {
        s[i] = hexDigits.substr(Math.floor(Math.random() * 0x10), 1);
    }
    s[14] = "4";
    s[19] = hexDigits.substr((s[19] & 0x3) | 0x8, 1);
    s[8] = s[13] = s[18] = s[23] = "-";

    var uuid = s.join("");

    return uuid.substring(0, length);
}

```

6. Browser detection object

`Browser()` [code](#)

```

const Browser = function() {
    const isIE = function () {
        return /*@cc_on!@*/false || !!document.documentMode;
    }

    const isFirefox = function () {
        return typeof InstallTrigger !== 'undefined';
    }
}

```

```

    }

    const isChrome = function () {
        return !!window.chrome && /Chrome/.test(navigator.userAgent) &&
        /Google Inc/.test(navigator.vendor) && !/OPR/.test(navigator.userAgent);
    }

    const isEdge = function () {
        return !this.isIE() && !!window.StyleMedia;
    }

    const isOpera = function () {
        return (!!window.opr && !!opr.addons) || !!window.opera ||
        navigator.userAgent.indexOf(' OPR/') >= 0;
    }

    const isiOS = function () {
        return /iPad|iPhone|iPod/.test(navigator.userAgent) &&
        !window.MSStream;
    }

    const isSafari = function () {
        let userAgent = navigator.userAgent.toLowerCase();
        return /(safari|applewebkit)/i.test(userAgent) &&
        !userAgent.includes("chrome") && !userAgent.includes("android");
    }

    const isAndroid = function () {
        return navigator.userAgent.toLowerCase().indexOf("android") > -1;
    }

    const isSafariWebRTC = function () {
        return navigator.mediaDevices && this.isSafari();
    }

    const isSamsungBrowser = function () {
        return /SamsungBrowser/i.test(navigator.userAgent);
    }

    const isAndroidFirefox = function () {
        return this.isAndroid() && /Firefox/i.test(navigator.userAgent);
    }

    const isMobile = function () {
        return this.isAndroid() || this.isiOS();
    }

    return {
        isIE: isIE,
        isFirefox: isFirefox,
        isChrome: isChrome,
        isEdge: isEdge,
        isOpera: isOpera,
        isSafari: isSafari,
        isSafariWebRTC: isSafariWebRTC,
        isSamsungBrowser: isSamsungBrowser,
        isAndroid: isAndroid,
        isiOS: isiOS,
    }

```

```
isAndroidFirefox: isAndroidFirefox,  
isMobile: isMobile  
}  
}
```

7. Workaround to play an audio in Safari 12 using a preloader

`playFirstSound()` [code](#)

```
const playFirstSound = function (parent, preloader) {  
  return new Promise(function (resolve, reject) {  
    let audio = document.createElement("audio");  
    audio.controls = "controls";  
    audio.muted = true;  
    audio.hidden = true;  
    audio.preload = "auto";  
    audio.type="audio/mpeg";  
    if (preloader) {  
      audio.src = preloader;  
      parent.appendChild(audio);  
      audio.play().then(function() {  
        audio.remove();  
        resolve();  
      }).catch(function (e) {  
        console.error("Can't play preloader: " + e);  
        reject();  
      });  
      return;  
    }  
    resolve();  
  });  
}
```

8. Get a short meeting room participant Id

`getShortUserId()` [code](#)

```
const getShortUserId = function (userId) {  
  return userId.substring(0, 4);  
}
```