Curso livre de WebRTC

Se não sabes, aprendes; Se já sabes, ensinas.

+

Porque unidos somos mais fortes !

Denis Manzetti. Paizão, Empreendedor, fullstack developer

- https://github.com/manzettidenis
- https://www.linkedin.com/in/denis-manzetti-90b22876/
- https://www.facebook.com/denis.manzetti
- manzetti.denis@gmail.com

- Fundador da comunidade EmpreenDev, Full Stack Developer na SW Digital e evangelista de javascript, node.js e empreendedorismo.

- https://www.facebook.com/groups/304955876574030/
- https://www.facebook.com/empreendev
- https://www.facebook.com/javascriptbrasil/
- https://github.com/manzettidenis/empreendev
- http://swdigital.com.br/

Erik Xavier. jesus & coder

- https://github.com/erikxavier
- https://www.facebook.com/vidadebarbudo
- https://www.linkedin.com/in/erik-xavier-de-alvarenga-monteiro-383a84106/
- erik.xam@gmail.com

- Formado em Análise e Desenvolvimento de Sistemas, amante de javascript e apreciador de tecnologias open-source. Apaixonou-se por WebRTC à primeira vista, e desde então vem focando seus estudos na área de comunicação de dados real time na web.

Pauta do curso

- O que é o WebRTC
- As 3 principais APIs
- MediaStream (aka getUserMedia)
- RTCPeerConnection
- RTCDataChannel

Projeto Final

Web-app de chamada de video e audio.

O que é WebRTC ? Plugin-free realtime comunication

- Comunicação de video e audio de alta qualidade e baixo custo ..

O que é WebRTC ?

- Comunicação de video e audio de alta qualidade e baixo custo ..
- ... e todo tipo de "data", ponto a ponto (peer-to-peer)

O que foi o WebRTC para o chrome (e para a Web):

- Ele trouxe real time comunication para a web.
- Permitiu que developers pudessem criar um novo ecosistema de comunicação.
- Trouxe o poder do media stack para o **chrome**.

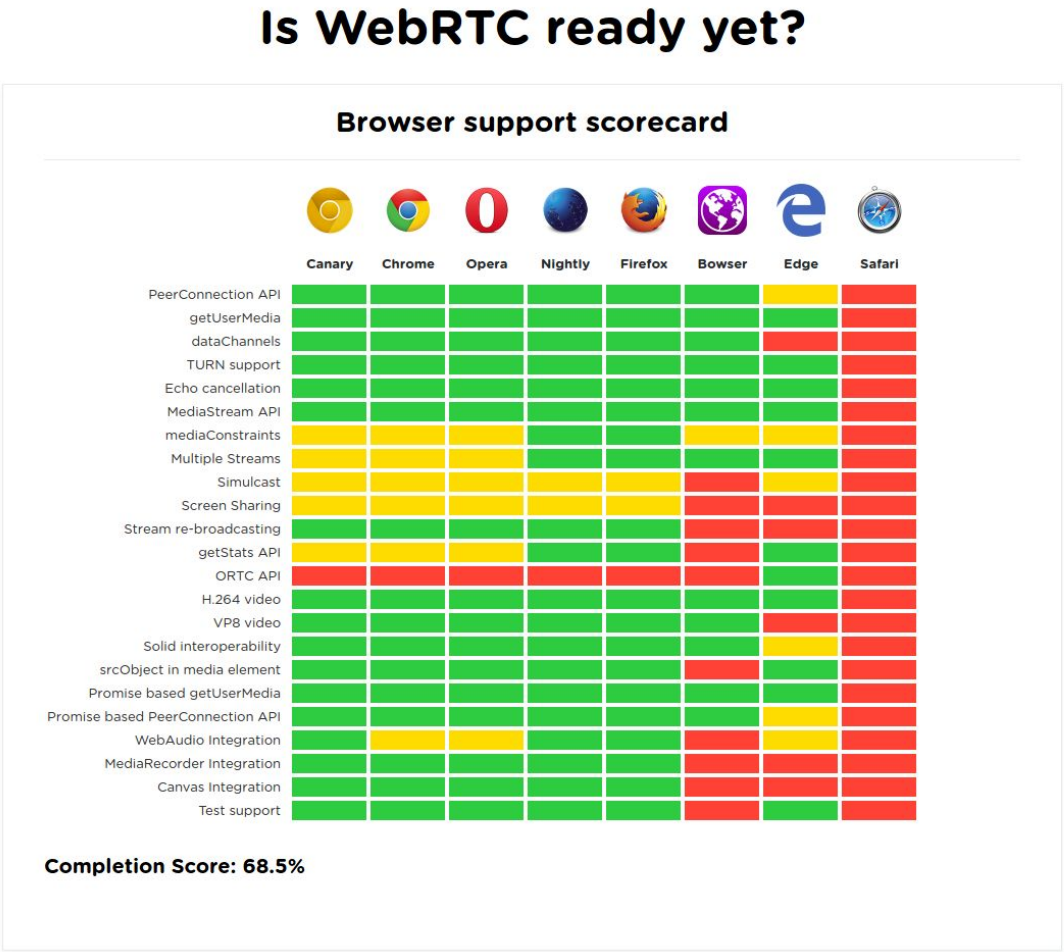
Em 2014, com o lançamento da versão 22 do firefox, a primeira com suporte a WebRTC fez com que mais de 1 bilhão de usuários usassem browsers com webrtc-enabled nativos.

Principais funções WebRTC

- Capturar audio e video
- Transmitir audio e video para outro ponto webrtc via internet.
- Transmitir data para outro ponto webrtc via internet.

Suporte ao WebRTC

http://iswebrtcreadyyet.com



Os 3 componentes da API

95.6% dos usuários usam browsers com suporte WebRTC nativo.

APIs Javascript

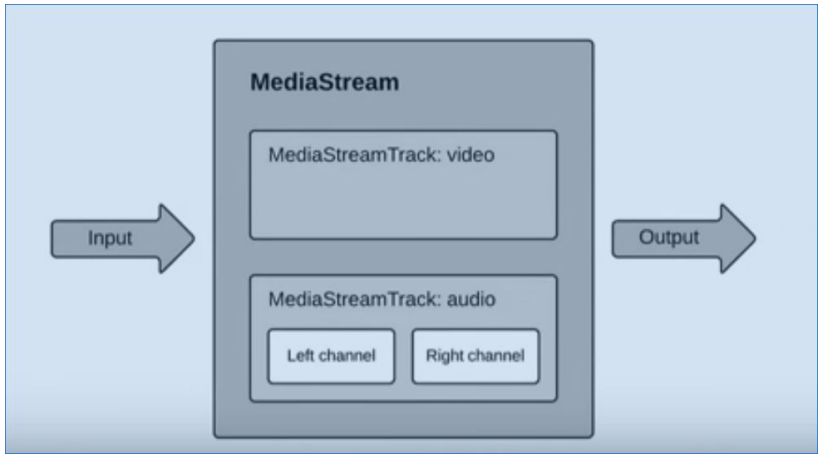
- MediaStream (GetUserMedia)
- RTCPeerConection
- RTCDataChannel

MediaStream

Aka (GetUserMedia)

Media Stream (aka getUserMedia)

- Representa um stream de audio e/ou video
- Pode conter múltiplas faixas (tracks)
- Obtém o MediaStream através do navigator.mediaDevices.getUserMedia



Media Stream

Propriedades

- **MediaStream.active (read only)** - returns true if MediaStream is active, or false otherwise.

- **MediaStream.id (read only)** - A unique identifier for the object.

Media Stream

Event handlers

- **MediaStream.onactive** - A handler for an active event that is fired when a MediaStream object becomes active.
- **MediaStream.onaddtrack** - A handler for an addtrack event that is fired when a new MediaStreamTrack object is added.
- **MediaStream.oninactive** - A handler for an inactive event that is fired when a MediaStream object becomes inactive.
- **MediaStream.onremovetrack** - A handler for a removetrack event that is fired when a MediaStreamTrack object is removed from it.ifier for the object.

Media Stream

Methods

- **MediaStream.addTrack()** - Adds the MediaStreamTrack object given as argument to the MediaStream. If the track has already been added, nothing happens.
- **MediaStream.clone()** - Returns a clone of the MediaStream object with a new ID.
- **MediaStream.getAudioTracks()** - Returns a list of the audio MediaStreamTrack objects from the MediaStream object.
- **MediaStream.getTrackById()** - Returns the track by ID. If the argument is empty or the ID is not found, it returns null. If several tracks have the same ID, it returns the first one.
- **MediaStream.getTracks()** - Returns a list of all MediaStreamTrack objects from the MediaStream object.
- **MediaStream.removeTrack()** - Removes the MediaStreamTrack object given as argument from the MediaStream. If the track has already been removed, nothing happens.

Usando Callbacks

```
const constraints = {video:true};

const succesCallback = (stream) => {
    const video = document.querySelector("video");
    video.src = window.URL.createObjectURL(stream);
}

const errorCallback = (error) => {
    console.log("navigator.getUserMedia error:", error);
}

navigator.getUserMedia(constraints, succesCallback, errorCallback);
```

Usando Callbacks

```
const constraints = {video:true};

const succesCallback = (stream) => {
    const video = document.querySelector("video");
    video.src = window.URL.createObjectURL(stream);
}

const errorCallback = (error) => {
    console.log("navigator.getUserMedia error:", error);
}

navigator.getUserMedia(constraints, succesCallback, errorCallback);
```

navigator.getUserMedia() está deprecated.

Agora usaremos navigator.mediaDevices.getUserMedia().

Usando Callbacks

```
const constraints = {video:true};

const succesCallback = (stream) => {
    const video = document.querySelector("video");
    video.src = window.URL.createObjectURL(stream);
}

const errorCallback = (error) => {
    console.log("navigator.getUserMedia error:", error);
}

navigator.mediaDevices.getUserMedia(constraints, succesCallback, errorCallback);
```

Usando Promises

```
navigator.mediaDevices.getUserMedia({video: true, audio: false})
   .then(stream => video.srcObject = stream)
   .catch(e => log(e.name + ": "+ e.message))

let stop = mediaType => video.srcObject.getTracks().forEach(track => track.kind == mediaType &&
track.stop());
```

Constraints

```
const constraints  = {
  audio: true,
  video: {
    width: { min: 1024, ideal: 1280, max: 1920 },
    height: { min: 776, ideal: 720, max: 1080 }
  }
}
```

```
{ audio: true, video: { facingMode: "user" } }
```

```
{ audio: true, video: { facingMode: { exact: "environment" } } }
```

Constraints

Properties of video tracks

`aspectRatio` - A `ConstrainDouble` specifying the video aspect ratio or range of aspect ratios which are acceptable and/or required.

`facingMode` - A `ConstrainDOMString` object specifying a facing or an array of facings which are acceptable and/or required.

`frameRate -` A `ConstrainDouble` specifying the frame rate or range of frame rates which are acceptable and/or required.

`height` - A `ConstrainLong` specifying the video height or range of heights which are acceptable and/or required.

`width` - A `ConstrainLong` specifying the video width or range of widths which are acceptable and/or

Constraints

Properties of audio tracks

`channelCount` -A `ConstrainLong` specifying the channel count or range of channel counts which are acceptable and/or required.

`echoCancellation` - A `ConstrainBoolean` object specifying whether or not echo cancellation is required or preferred and/or required.

`latency` - A `ConstrainDouble` specifying the latency or range of latencies which are acceptable and/or required.

`sampleRate` - A `ConstrainLong` specifying the sample rate or range of sample rates which are acceptable and/or required.

`sampleSize -` A `ConstrainLong` specifying the sample size or range of sample sizes which are acceptable and/or required.

`volume -` A `ConstrainDouble` specifying the volume or range of volumes which are acceptable and/or required.

Retrocompatibilidade

```
// Older browsers might not implement mediaDevices at all, so we set an empty object first
if (navigator.mediaDevices === undefined) {
  navigator.mediaDevices = {};
}

// Some browsers partially implement mediaDevices. We can't just assign an object
// with getUserMedia as it would overwrite existing properties.
// Here, we will just add the getUserMedia property if it's missing.
if (navigator.mediaDevices.getUserMedia === undefined) {
  navigator.mediaDevices.getUserMedia = function(constraints) {

    // First get ahold of the legacy getUserMedia, if present
    var getUserMedia = navigator.webkitGetUserMedia || navigator.mozGetUserMedia;

    // Some browsers just don't implement it - return a rejected promise with an error
    // to keep a consistent interface
    if (!getUserMedia) {
      return Promise.reject(new Error('getUserMedia is not implemented in this browser'));
    }

    // Otherwise, wrap the call to the old navigator.getUserMedia with a Promise
    return new Promise(function(resolve, reject) {
      getUserMedia.call(navigator, constraints, resolve, reject);
    });
  }
}
```

```
navigator.mediaDevices.getUserMedia({ audio: true, video: true })
.then(function(stream) {
  var video = document.querySelector('video');
  // Older browsers may not have srcObject
  if ("srcObject" in video) {
    video.srcObject = stream;
  } else {
    // Avoid using this in new browsers, as it is going away.
    video.src = window.URL.createObjectURL(stream);
  }
  video.onloadedmetadata = function(e) {
    video.play();
  };
})
.catch(function(err) {
  console.log(err.name + ": " + err.message);
});
```

Exemplo

Caputarando Mídia

https://manzettidenis.github.io/exemplos/aula01/exemplo-01-getusermedia-promises.html

Exemplo

Snapchating com webrtc e canvas

https://manzettidenis.github.io/exemplos/aula01/exemplo-02-snapchating-video-with-canvas

Exemplo

Brincando com filtros css

https://manzettidenis.github.io/exemplos/aula01/exemplo-03-filter-css

EXEMPLO BÔNUS

FACE CAPTURE COM WEBRTC

https://webrtc.github.io/samples/src/content/getusermedia/face/

RTCPeerConnection

Na próxima aula aprenderemos sobre essa api e como criar conexões peer-to-peer com WebRTC

Por hoje é só pessoal...

Abraços, e até a próxima aula, domingo que vem.