
# WebRTC

## Communication temps-réel sans plugin

Frédéric LUART - Apizee
Pierre-Yves LAPERSONNE - Orange

Version 8 - 09/2014

Cette œuvre est mise à disposition selon
Licence Creative Commons Attribution 4.0 International

# PY LAPERSONNE

Software developer

pylapp.github.io

# Frédéric LUART

CTO Apizee

@Apizee_FR

Code d'Armor

# console.log(“Au menu ce soir...”)

- La naissance de *WebRTC*
- Les piliers de l’*API*
- L’infrastructure *WebRTC*
- *apiRTC* : un *framework* contre la migraine
- Démos !

Code d’Armor

# console.log(“La naissance de WebRTC”)

Raconte-moi une histoire...

Code d’Armor

# Au début furent les ténèbres...

Code d'Armor

- Pas de communication temps-réel “légères”

- Mais déjà des solutions avec les browsers web...
  - ...technos propriétaires
  - ...à installer
  - ...à mettre à jour
  - ...pas compatibles partout !


Code d'Armor

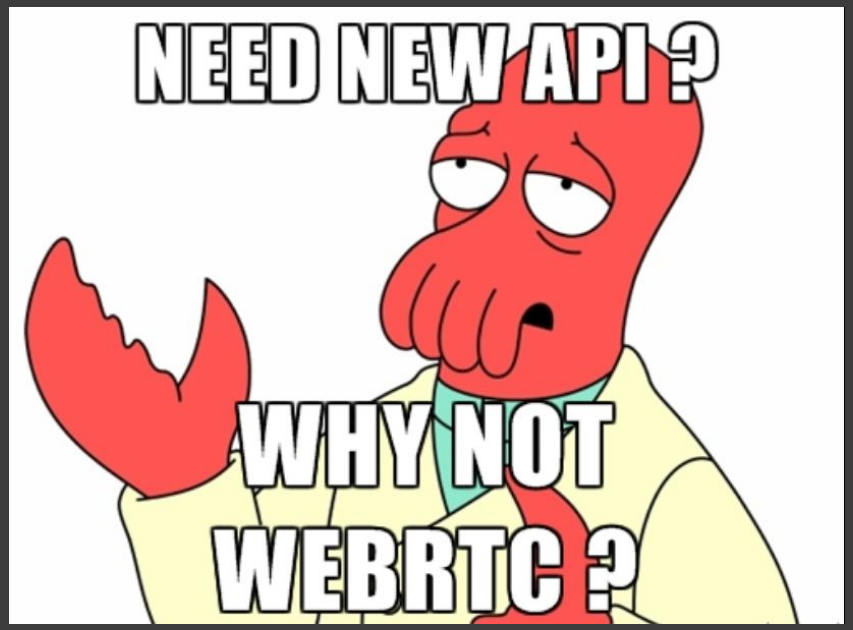


Code d'Armor

- Problèmes :
  - Rien de natif dans les browsers web
  - Dépendance totale envers les plugins
  - Sécurité, stabilité, mise à jour...
  - Plugins lourds et gourmands
  - **Aucune standardisation !**

Code d'Armor

# Et puis vint la lumière !




Code d'Armor

- Apparition de WebRTC :
  - **Normalisation de la chaine** de communication
  - **Intégration native** dans les browsers (lib C/C++)
  - **API full web simplifiée**
    - HTML5 : <audio>, <video>, ...
    - JavaScript, JavaScript, JavaScript
  - Environnement connu
    - Codecs G711 et Opus, VP8
    -   
  - **Multiplateforme** !
    - mobile natif & web, browsers “classiques”

Code d’Armor


verizon


skype™


Google


W3C®


orange™


ERICSSON


at&t


Code
d'Armor

- Petite question :


Code d'Armor

- Pour Microsoft, double jeu :
  - Implication par le biais de Skype
  - Travail sur sa solution : **CU-RTCWeb**
    - SDP non obligatoire pour la signalisation
    - Pas de PeerConnection
    - Codec propriétaire$ (H.264)
- Pour Apple, **silence radio total**...
  - Aucune contribution au projet
  - Minimum de compatibilité pour Safari

Code d'Armor

# console.log(“Les piliers de l’API”)

Place au code (un peu) !

Code d’Armor

- Les ingrédients de base :

  ✓ Un navigateur Web **récent**

  ✓ Du code **JavaScript**
  - ➢ utilisant l'API
  - ➢ faisant la signalisation
  - ➢ gérant les éléments des pages Web

  ✓ **WebSocket**
  - ➢ optionnel mais très pratique :)
  - ➢ connexion avec le serveur

Code d'Armor

```
var maWS = null;

function faireConnection()
{
	maWS = new WebSocket(“ws://super.url:port/truc”);
	maWS.open = onWsOpen;
	maWS.onmessage = onWsMessage;
	maWS.onerror = onWsError;
	maWS.onclose = onWsClose;
}

function onWsMessage( msg )
{
	// Le code qui va bien
}

// Etc...
```

Code d'Armor

- **Pilier #1 : les users medias / media streams**
  - Concernent tous les flux **temps-réel**
    - microphone, caméra, webcam
  - **Demandés** par le navigateur
    - l'utilisateur doit accepter
    - le périphérique est choisi à ce moment
  - **Média Stream**
    - 1 entrée et 1 sortie
    - local
      - via périphérique de la machine
    - non-local
      - <video>, <audio>, peer connection

Code d'Armor

- Media Stream :
  - Flux de données audio et vidéo
  - Représenté par une **URL**
  - Composé de plusieurs **MediaStreamTrack**
- Media Stream Track :
  - Comporte une **piste** audio ou vidéo
  - Composé de un ou plusieurs **canaux**
    - ➢ signal HP gauche, signal HP droite, ...

Code d'Armor

```
navigator.getUserMedia =
        navigator.getUserMedia
        || navigator.webkitGetUserMedia
        || navigator.mozGetUserMedia
        || navigator.msGetUserMedia;

if ( navigator.getUserMedia ){

        navigator.getUserMedia(
                {video:true, audio:true},
                onCaptureMediaSuccess,
                onCaptureMediaFailure
        );

} else {
        // Kaboum !
        // Fonction non supportée dans ce navigateur
}
```

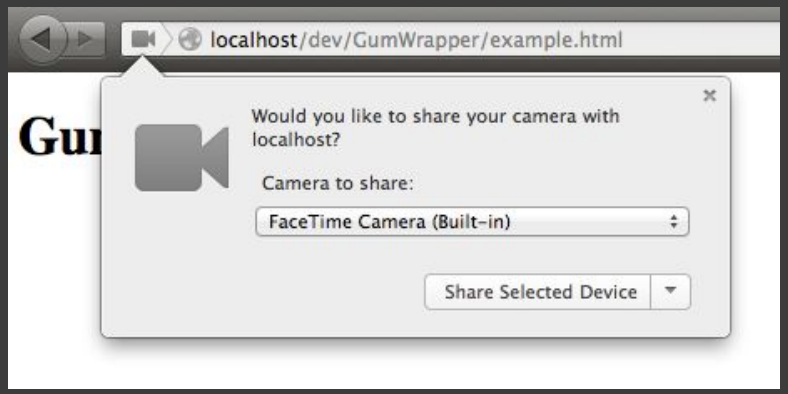


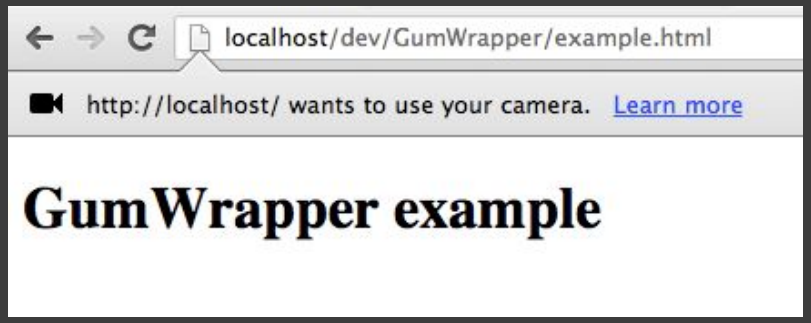
Code d'Armor

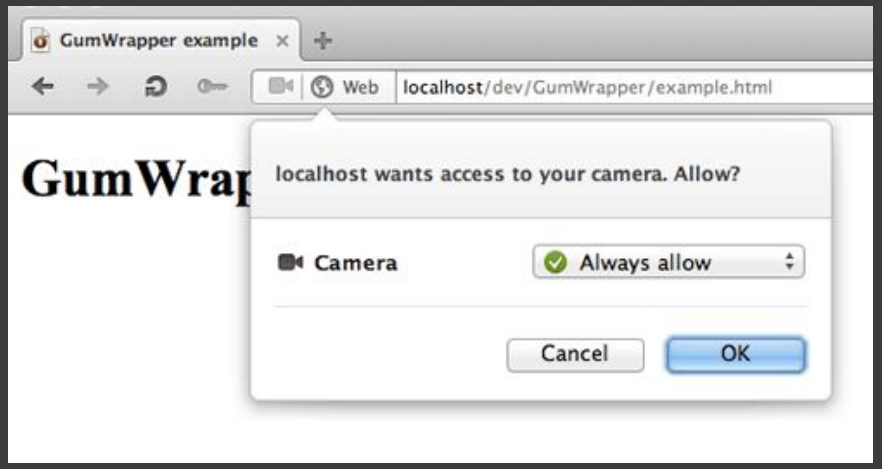
```
// Callback en cas de succès :-)
var onCaptureMediaSuccess = function ( stream ){
        // Mise à jour d'un élément HTML
        maBaliseVideo.src = window.URL.createObjectURL(stream);
        // Ajout du flux média à la peer connection
        maPeerConnection.addStream(stream);
        // Génération offre SDP, ...
};


// Callback d'échec :-(
var onCaptureMediaFailure = function ( kaboum ){
        console.error("Boum!");
}
```

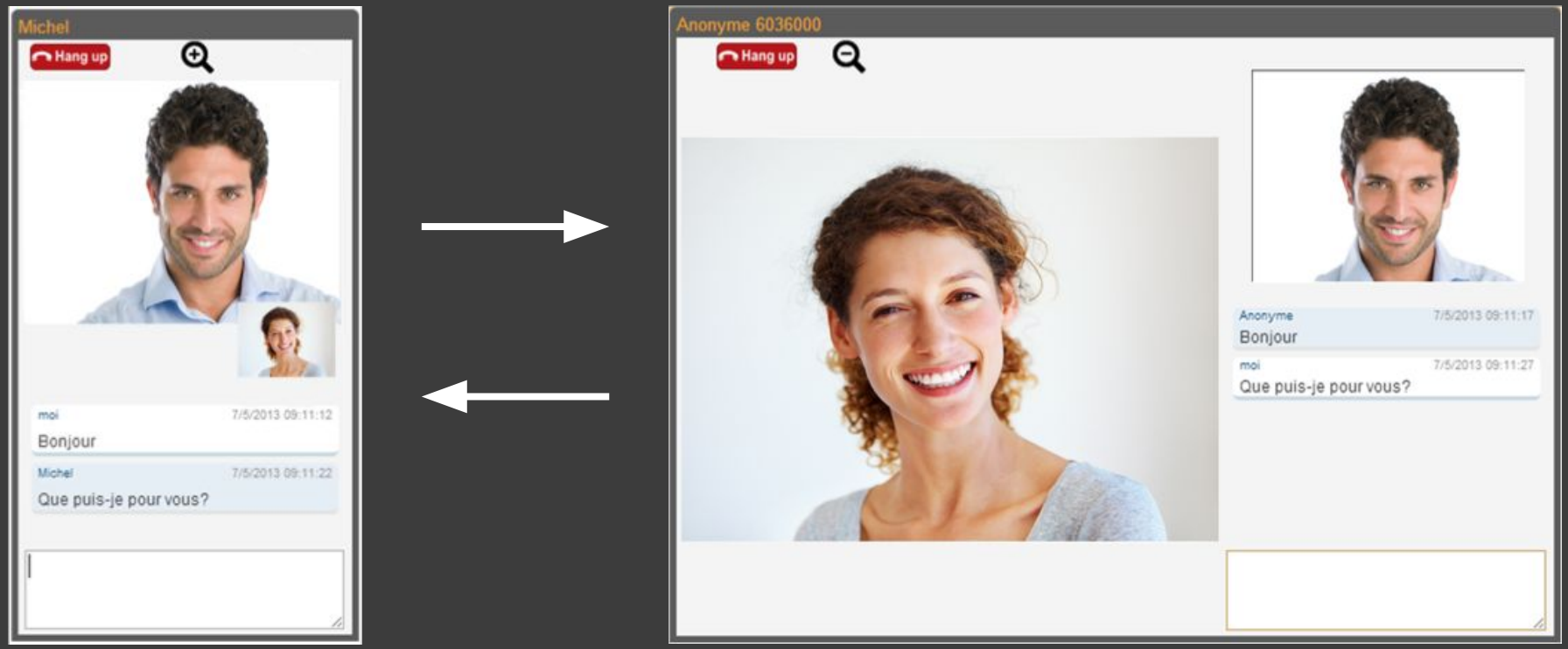
Code d'Armor


localhost/dev/GumWrapper/example.html
Would you like to share your camera with localhost?
Camera to share:
FaceTime Camera (Built-in)
Share Selected Device


localhost/dev/GumWrapper/example.html
http://localhost/ wants to use your camera. Learn more
GumWrapper example


GumWrapper example
Web
localhost/dev/GumWrapper/example.html
localhost wants access to your camera. Allow?
Camera
Always allow
Cancel
OK

- **Pilier #2 : la RTCPeerConnection**



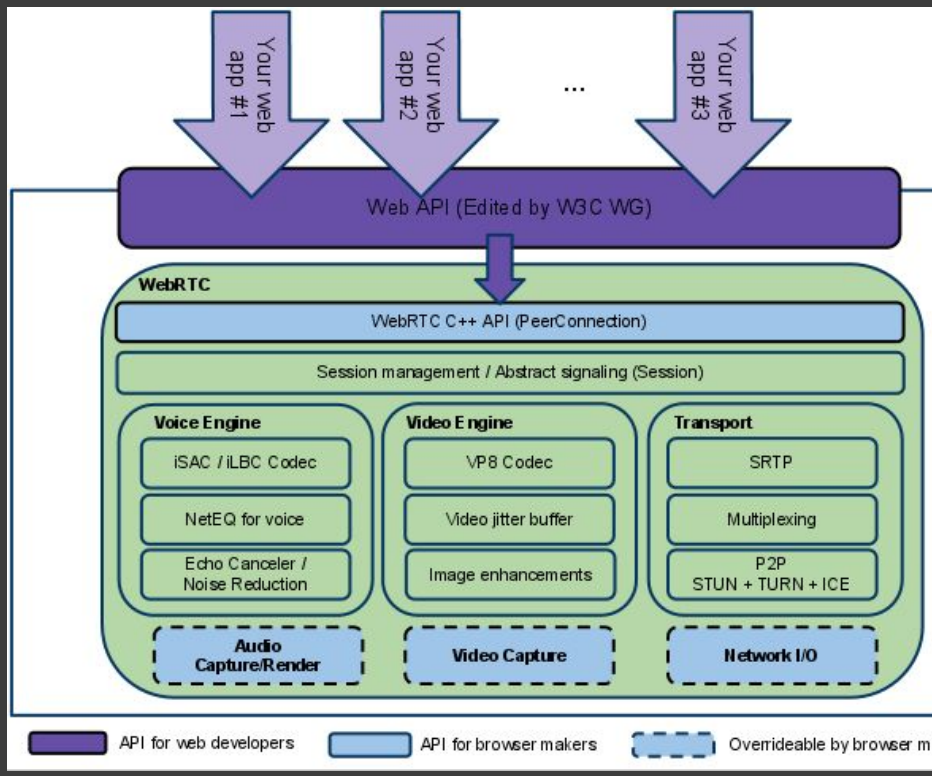

API simple : getUserMedia + peerConnection -> remote MediaStream

Code d'Armor

- **Pilier #2 : la RTCPeerConnection**

**Process Audio & Video**

- suppression bruit
- compression (codecs)
- gestion du routage
- media (NATs & firewalls)
- encryption (SRTP)
- gestion de la bande passante




**API**

- création d'une PeerConnection
- ajout d'un MediaStream
- et quelques methodes pour établir l'appel

Code d'Armor

- **RTCPeerConnection : initialisation**
  - Vérifie les conditions de médias locaux : résolution, capacités, codecs
  - Récupère les adresses réseaux de l'application : candidates (host, STUN, TURN)
  - Enfin échanges des infos via le canal de signalisation

Exemple dans la norme :
w3.org/TR/webrtc/#simple-peer-to-peer-example

Code d'Armor

- **Pilier #2 : la RTCPeerConnection**

CreatePeerConnection()
CreateOffer()
SetLocalDescription()

offer

Start Ice process and connectivity checking

CreatePeerConnection()
SetRemoteDescription()
CreateAnswer()
SetLocalDescription()

answer

SetRemoteDescription()

candidate

Start Ice process and connectivity checking

onIceCandidate()

candidate

onIceCandidate()

**Audio, video**



Code d'Armor

- **Pilier #3 : le RTC Data Channel**
  - API équivalente à WebSocket
  - Supporte String + type binaire JavaScript : Blob, ArrayBuffer, ArrayBufferView
  - Sécurisé avec DTLS (dérivé de SSL) - obligatoire

| | TCP | UDP | SCTP |
|---|---|---|---|
| **Reliability** | reliable | unreliable | configurable |
| **Delivery** | ordered | unordered | configurable |
| **Transmission** | byte-oriented | message-oriented | message-oriented |
| **Flow control** | yes | no | yes |
| **Congestion control** | yes | no | yes |

Code d'Armor

```
var dCOptions = {
  ordered: false, // do not guarantee order
  maxRetransmitTime: 3000, // in milliseconds
};

var peerConnection = new RTCPeerConnection();

var dataChannel = peerConnection.createDataChannel("myLabel", dCOptions);
dataChannel.onerror = function(error){console.log("Data Channel Error:", error);};
dataChannel.onmessage = function(event){console.log("Got Data Channel
Message:", event.data);};

dataChannel.onopen = function () {
  dataChannel.send("Hello World!");
};
dataChannel.onclose = function (){console.log("The Data Channel is Closed");};
```

Code d'Armor

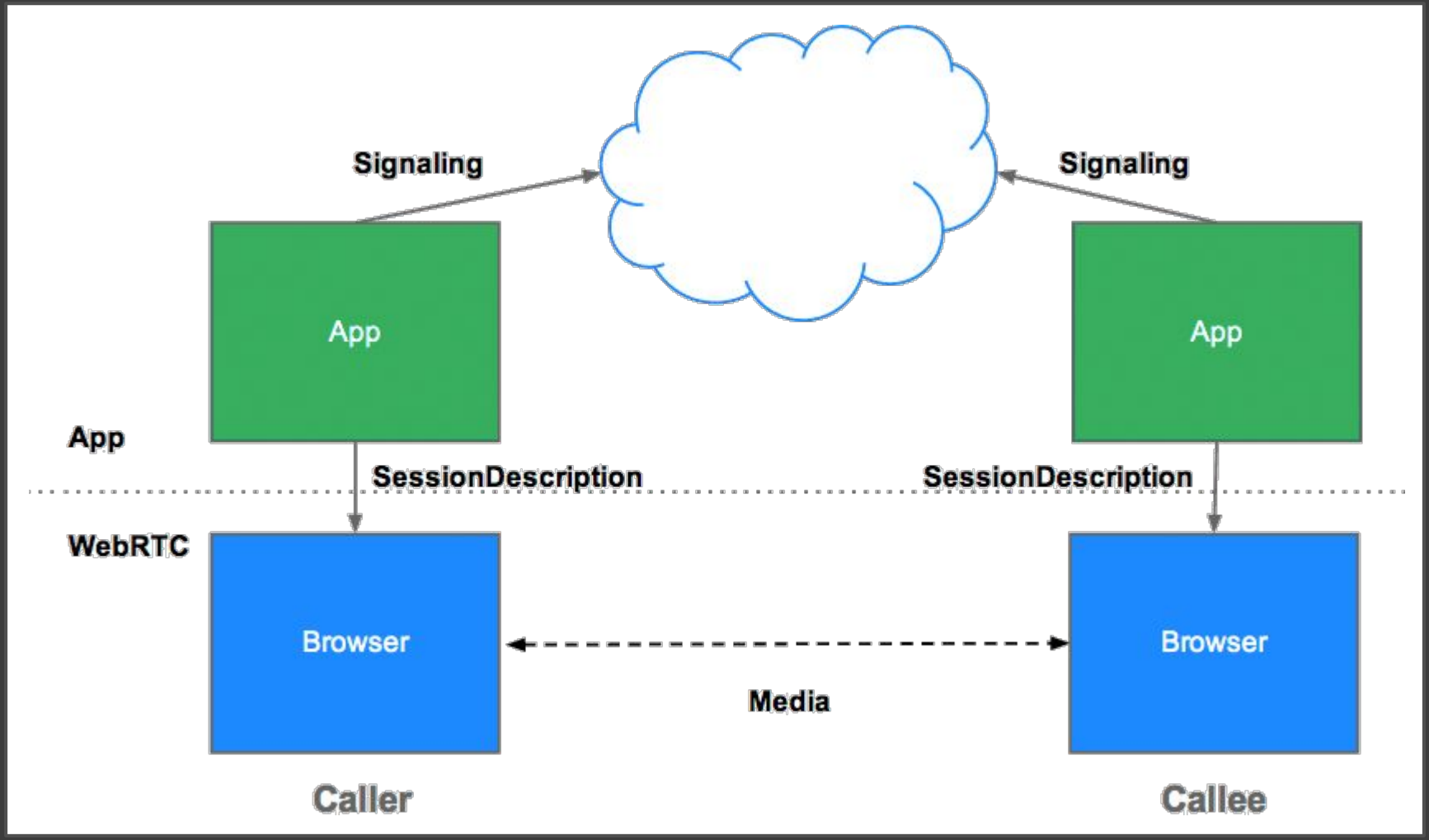
| WebRTC | Web Socket | RTC PeerConnection | Media Stream | RTC DataChannel |
|---|---|---|---|---|
| **Chrome** | 16+ | 20+ | 18+ | 26+ |
| **Firefox** | 11+ | 24+ | 20+ | 24+ |
| **Opera** (presto) | 12 | - | 12 | - |
| **Opera** (webkit) | 15+<br>16+ mobile | 18+<br>16+ mobile | 18+<br>16+ mobile | 18+<br>16+ mobile |
| **Safari** | 6+ | - | - | - |
| **Internet Explorer** | 10+ | - | - | - |

# console.log(“L’infrastructure WebRTC”)

Peer to peer : but we need server >@%&$%&

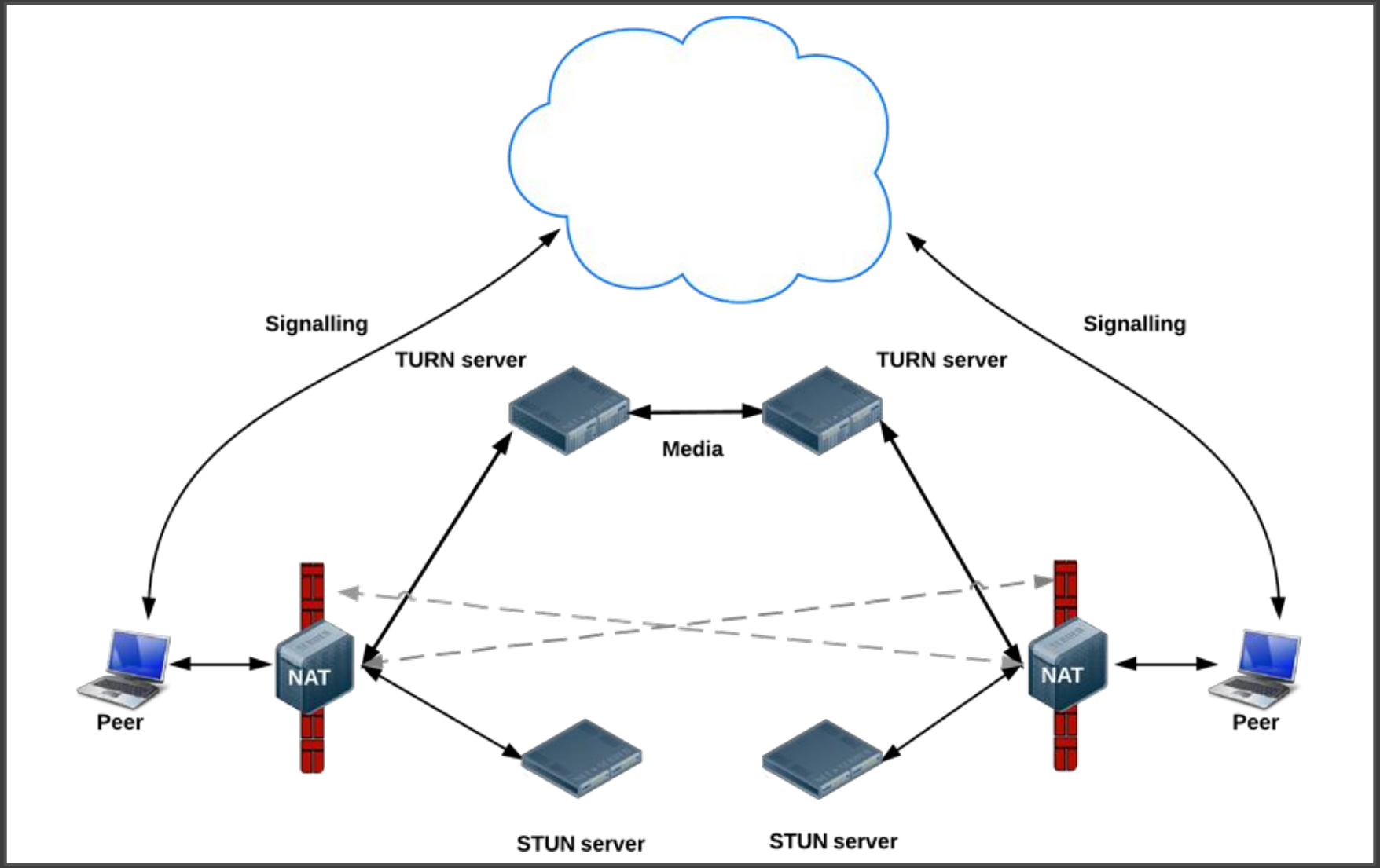
Code d’Armor

- **JSEP Architecture**



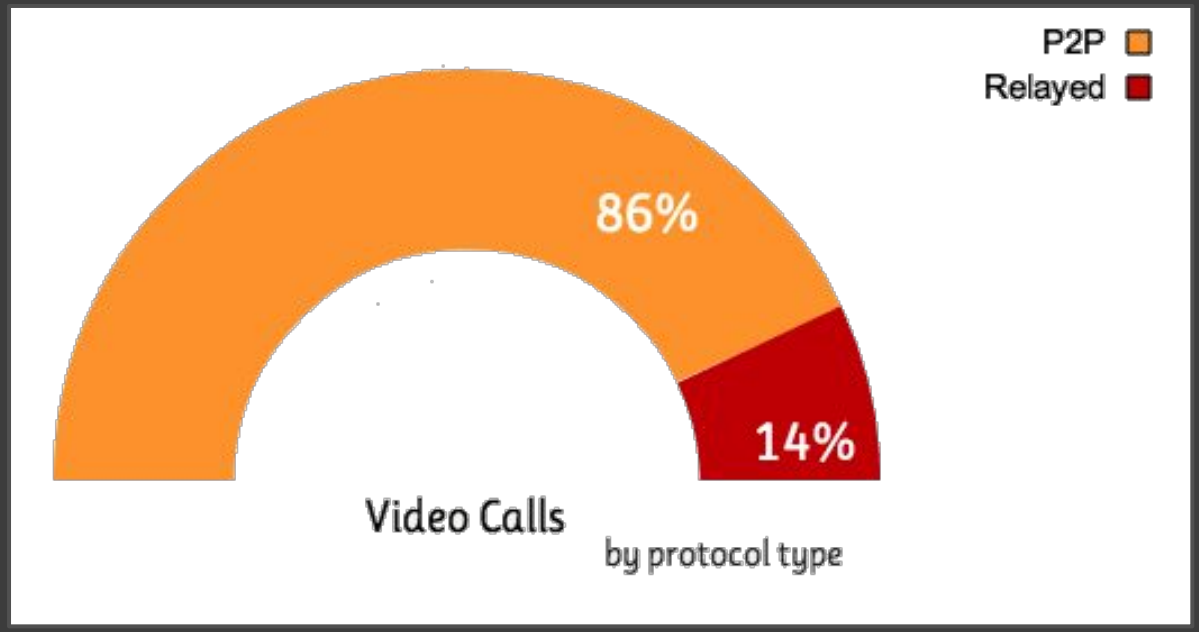

Code d'Armor

- **STUN, TURN**




Code d'Armor

- **ICE**
  - permet d’établir la connection
  - détermine le meilleur chemin pour chaque appel
  - la majorité des appels utilise STUN




Code d’Armor

- **RTCSessionDescription**

Use of SDP

```
v=0
o=- 6225574337129023010 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS 7ggGWSPAbwihaka6SUr59h9PqX4x5CGvM7VH
m=audio 1 RTP/SAVPF 111 103 104 0 8 106 105 13 126
c=IN IP4 0.0.0.0
a=rtcp:1 IN IP4 0.0.0.0
a=ice-ufrag:k4o07seRUgvxGDO1
a=ice-pwd:NyOZCJbWUprps1Kuxmka3D9O
a=ice-options:google-ice
a=fingerprint:sha-256
67:E1:C7:C3:88:80:4B:05:4E:B1:9F:F7:75:CA:04:2B:7E:BC:D4:11:76:DF:E8:3D:97:20:1B:5F:01:3C:C8:8
C
...
a=sendrecv
a=rtcp-mux
a=rtpmap:111 opus/48000/2
….
```

Code d'Armor

- **Candidate**

{"type":"candidate","label":0,"id":"audio","candidate":"a=candidate:1862263974 1 udp 2122260223 192.168.1.73 52046 typ host generation 0\r\n"}

{"type":"candidate","label":0,"id":"audio","candidate":"a=candidate:2565840242 1 udp 1686052607 2.16.43.105 52046 typ srflx raddr 192.168.1.73 rport 52046 generation 0\r\n"}

{"type":"candidate","label":0,"id":"audio","candidate":"a=candidate:2634085855 1 udp 41885439 46.145.45.125 52397 typ relay raddr 2.16.43.105 rport 63754 generation 0\r\n"}

Code d'Armor

- **Mal au crâne ?**

Code d'Armor

# console.log(“apiRTC”)

Qu’est-ce que *apiRTC* ?

Code d’Armor

- **apiRTC :** framework et plateforme de communication

Co-Browsing
Page Navigation, form ...

Whiteboard
On Photo, color, shape picker ..

Instant Messaging Client
Real time message exchange, Group Chat

WebRTC Client
Call : One way, 1 to 1, N to 1, N to N, Mute audio, video, AutoAnswer, Media routing mode selection, take snapshot, screenSharing …

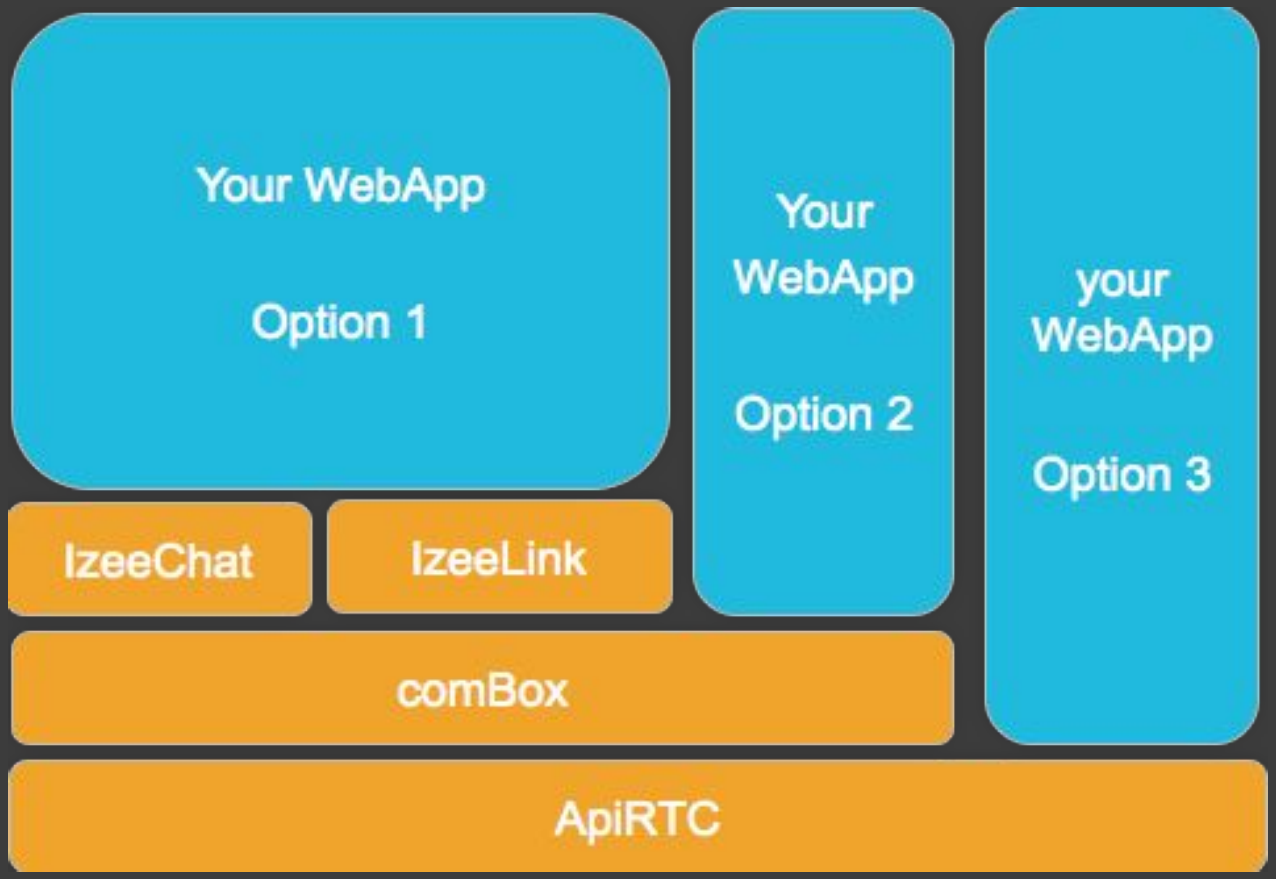
Session
Connection / Disconnection : Long polling, WebSocket,
Authentication, Presence
Conversation History (CDR, messages …)

Available on the App Store

ANDROID APP ON Google play

Code d'Armor

- Different niveau d’abstraction des APIS :



- apiRTC example

```
<script>
  apiRTC.init({
      apiKey : "myDemoApiKey",
      //apiCCId : "1234",
      onReady : function () {
        console.log('Session created with sessionId :' + apiRTC.session.apiCCId);
        var webRTCClient = apiRTC.session.createWebRTCClient({
            localVideo : "myLocalVideo",
            minilocalVideo : "myMiniVideo",
            remoteVideo : "myRemoteVideo",
            status : "status",
            command : "command"
        });
      }
  });
</script>
```

Code d'Armor

# console.log(“Démos !”)


Code d'Armor

- Share Fest
  - https://www.sharefest.me/
- Facekat
  - http://shinydemos.com/facekat/
- Frisb
  - http://www.frisb.com/
- Izeeroom
  - https://izeeroom.apizee.com/
- Magic Xylophone
  - http://soundstep.com/blog/experiments/jsdetection/
- *apiRTC*
  - IzeeChat - Web, iOS, android
  - IzeeLink in social network
  - IzeeDiag

Code d'Armor


Merci !
About being nice to your neighbors
SPEED LIMIT 80
Et merci à Sam Dutton pour son aide :-)
Code d'Armor


Code d'Armor

- Quelques démos / API supplémentaires
  - Spacegoo
    - ➢ http://www.spacegoo.com/chess/
  - Browser Quest
    - ➢ http:/browserquest.mozilla.org
  - Banana Bread
    - ➢ https://developer.mozilla.org/fr/demos/detail/bananabread
  - Cubeslam
    - ➢ https://www.cubeslam.com/xlkipm
  - ScretlyMeet.me
    - ➢ https://secretlymeet.me/
  - Peer5
    - ➢ https://peer5.com/
  - Twilio
    - ➢ http://www.twilio.com/

Code d'Armor

- Quelques démos / API supplémentaires
  - TenHands
    - ➢ https://tenhands.net/Home.htm
  - This Is Drum
    - ➢ http://http://thisisdrum.com/

- Quelques liens
  - Exemple tiré du W3C
    - ➢ http://www.w3.org/TR/webrtc/#simple-example
  - Architecture WebRTC côté navigateur
    - ➢ http://www.webrtc.org/reference/architecture
  - Lycode
    - ➢ http://lynckia.com/licode/
  - Le code C/C++ à la base de tout
    - ➢ http://www.webrtc.org/webrtc-native-code-package
  - Supports des technos selon le navigateur
    - ➢ http://caniuse.com/#feat=rtcpeerconnection
  - Tutoriel sur HTML5ROCKS
    - ➢ http://www.html5rocks.com/en/tutorials/webrtc/basics/
  - Les aspects STUN / TURN
    - ➢ http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

Code d'Armor


About being nice to your neighbors
SPEED LIMIT 80
Code d'Armor