November 2017

WebRTC

Björn Helgeson
Prototyp Stockholm AB

Björn Helgeson
bjorn.helgeson@prototyp.se

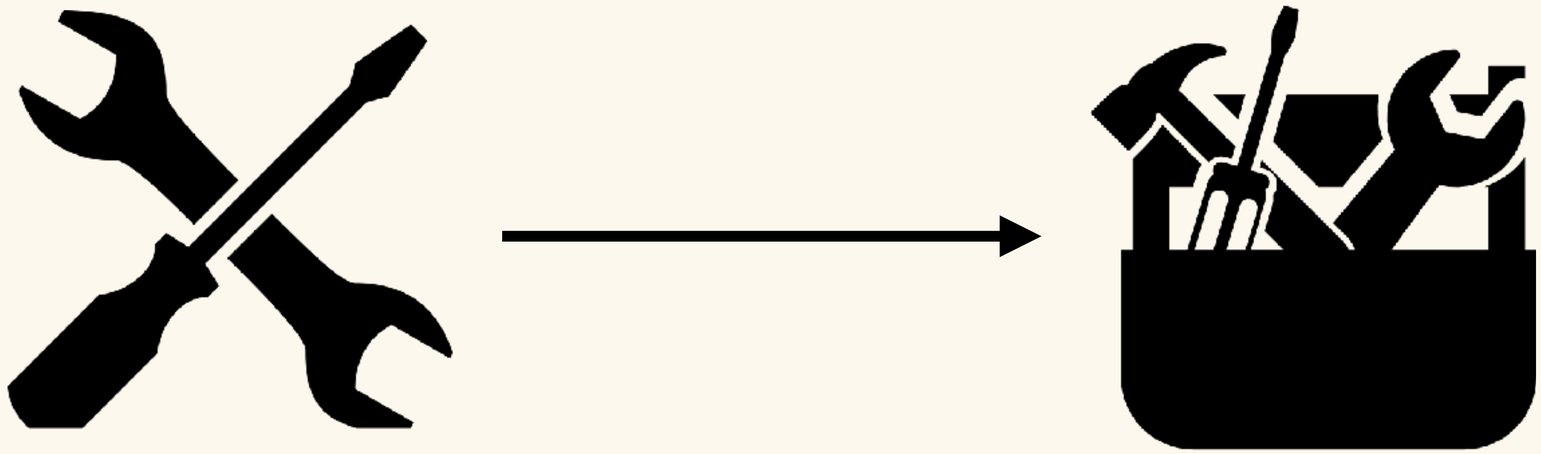
PROTOTYP

www.prototyp.se

Today 10-16

1. Tech Introduction
2. Tasks 1-4
3. Take a group photo!

This workshop

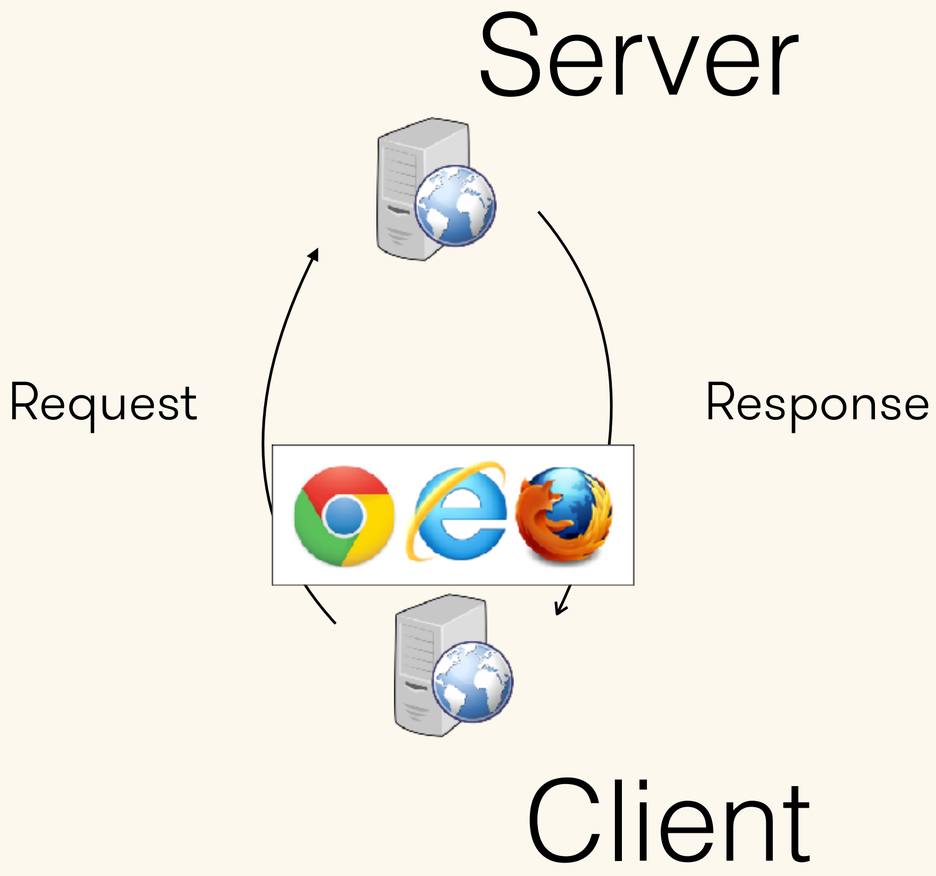


```html
<!DOCTYPE html>
<html>
<head>
<title>My home page</title>
</head>

<body>
<h1>Welcome!</h1>
<p>This is my website</p>
</body>

</html>
```

What's this?

```
var fruitSalad = ["Banana", "Apple", "Strawberry", "Mango"];
var len = fruitSalad.length;
var text = "This fruit salad contains: ";

for (var i = 0; i < len; i++) {
    text = text + fruitSalad[i];
}
```

Web browsers



The Web & HTML5

- Traditional web technologies
  – HTML, Javascript & CSS
- New web technologies
  – HTML5, WebRTC

Enabling powerful browser applications
without the need to install additional software

HTML5 Features

- New elements like <video>, <audio> and <canvas>
- Local Storage
- Better video & audio support
- Support for Geolocation, Offline Web Applications, WebSockets
- WebRTC

What is WebRTC?

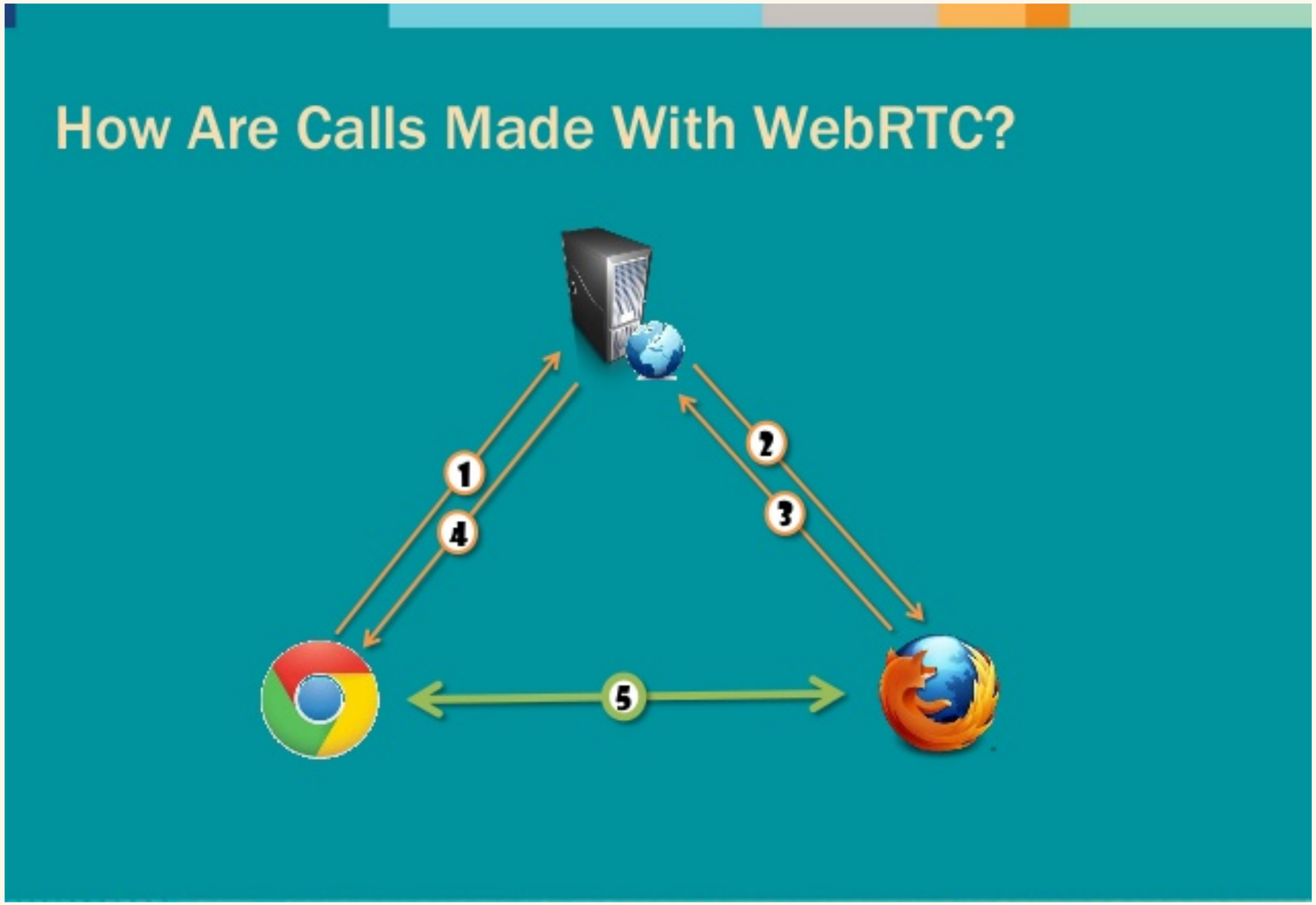
- Real Time Communication
  - Voice Calling
  - Video Chat
  - P2P file sharing

- It's not a service, it's a technology

- We will be using JS-API

Is webrtc ready yet?

WebRTC - What’s the thing?

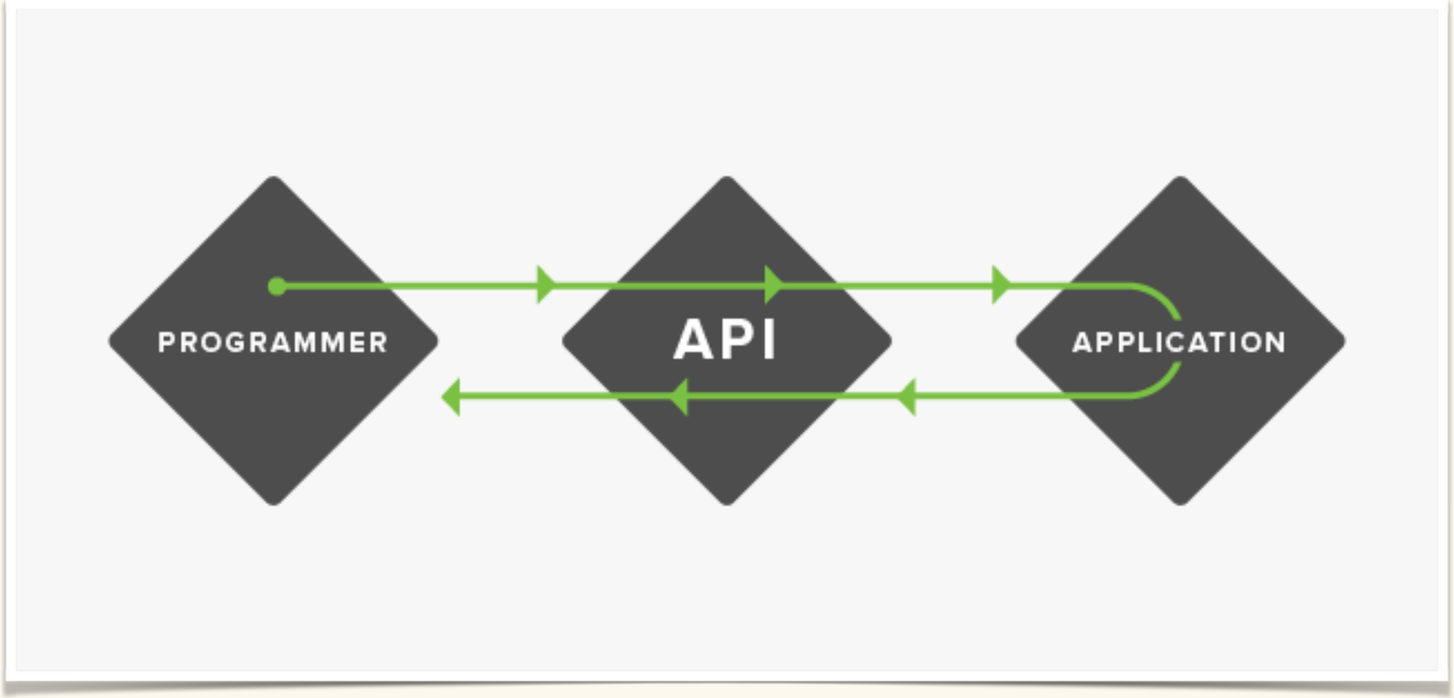


WebRTC - Examples

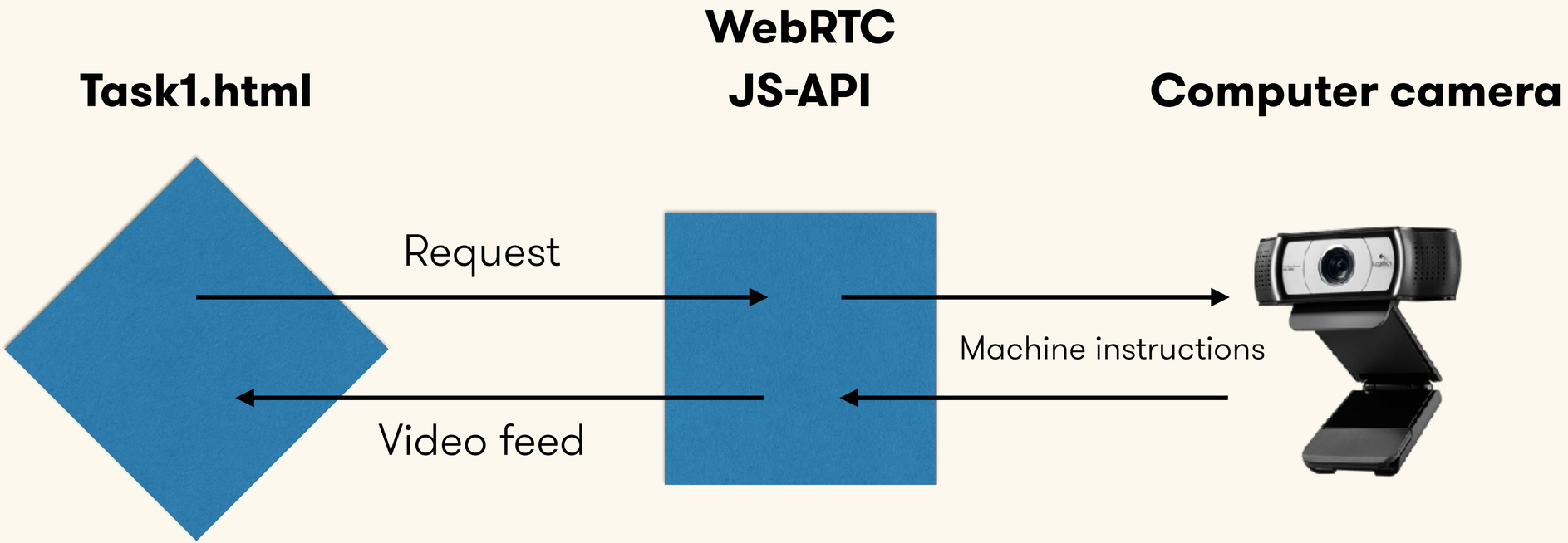
1. http://webcamtoy.com/
2. https://appear.in/
3. https://live.pics.io/
4. https://www.sharefest.me/
5. https://github.com/brianchirls/Seriously.js/

What’s an API?

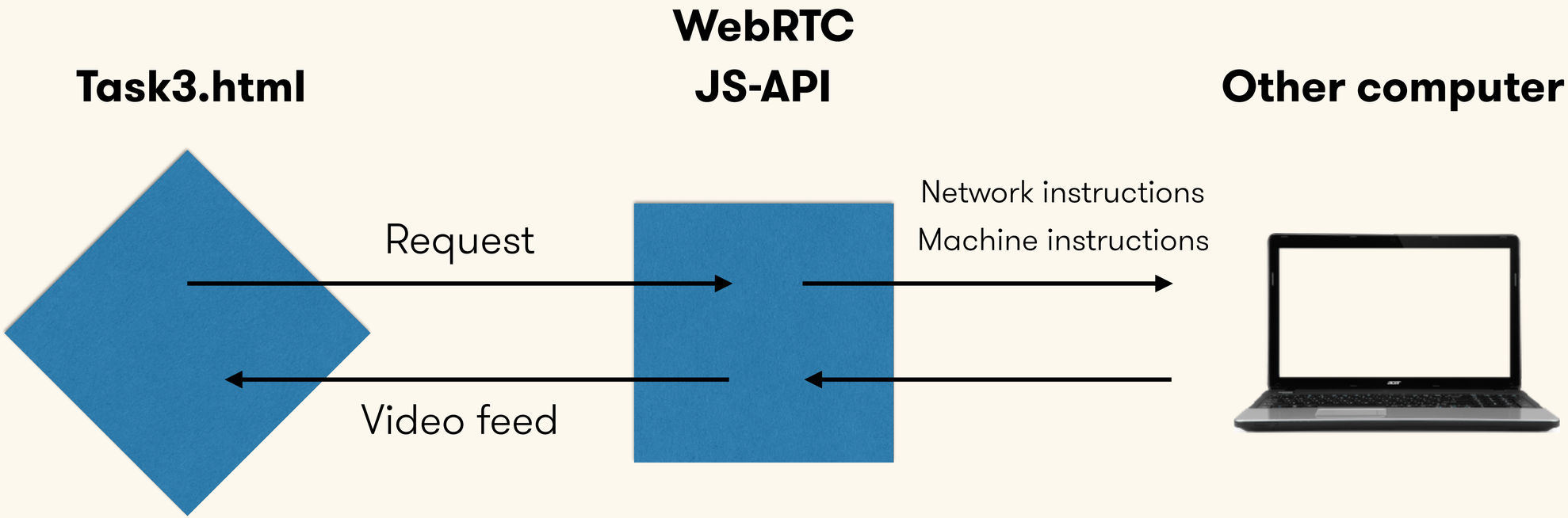
- Brief intro to API's
- For a programmer:



What’s an API


Task1.html
WebRTC
JS-API
Computer camera
Request
Video feed
Machine instructions

What’s an API


Task3.html
WebRTC
JS-API
Other computer
Request
Video feed
Network instructions
Machine instructions

WebRTC consists of

1. An implementation of SRTP with an SDP control mechanism on top
2. A media engine with G.711, Opus and VP8 Codecs
3. A VoIP implementation using STUN, TURN and ICE for NAT traversal

WebRTC API - JS Major Components

1. `getUserMedia` – grabs your computers camera and/or microphone after being granted permission, and captures the streams.
2. `PeerConnection` – sets up audio/video calls between two parts
3. `DataChannels` - allow browsers to share files or other data directly via peer-to-peer

WebRTC - getUsermedia

```
navigator.getUserMedia(
   constraints, success, error
)
```

See task1.html

WebRTC - Important components

- Session control messages - to initialize or close the communication
- ICE Candidate - tells you who you are to the outside world (your computer's IP address and port)
- Session description - tells you what codecs and resolutions can be handled by your browser and the browser it wants to communicate with

Web Security, SSL & Certificates

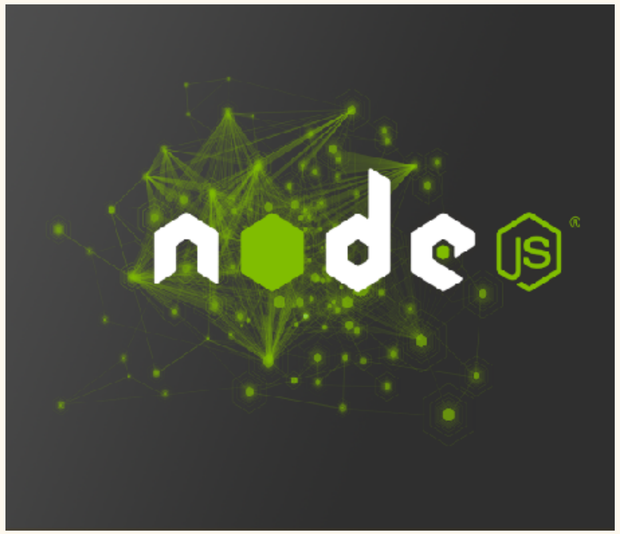
- Some parts of the WebRTC API requires data to be transferred encrypted.
- Therefore we need https traffic enabled on our server
- Therefore we need a certificate
- Self-signed certificates
- https://www.youtube.com/watch?v=SJJmoDZ3il8

Other Tools

Node.js

Platform to easily build fast network applications

Host a server easily on your own computer



Other Tools

Seriously.js

A real-time, node-based video effects compositor for the web built with HTML5 & Javascript

Preparation

1. Go to http://nodejs.org/
2. Download node.js for your system
3. Make sure you have a text editor or download for instance Sublime Text 3

First Task

1. Make sure you completed the previous steps described in the Preparation slide.
2. Download the material from the course web.
3. Starting the server.
   a) Open up a command terminal on your computer.
   b) Navigate to the folder with the downloaded course materials, and to the "http" folder.
   c) In your command line or terminal - Run: `node http_server.js`
4. Test in your browser at: http://localhost:8080/task1.html

   If everything is working your browser should ask you for your camera and after accepting you should see yourself.
5. Enable the audio so you can hear yourself (beware of acoustic feedback)

Second Task

1. Go to: http://localhost:8080/task2.html
2. Play with the filters!
3. Replace the image with your webcam video
4. Apply cool CSS filters to your video chat

Third Task

1. As we get into multi-peer video, things get a bit trickier. We will need a server with encrypted traffic (https), and therefore a certificate. For this example we will use a self-signed certificate, which is easy to set up but will give a client browser a warning when accessing your server. The code in task 3 and 4 is put together mostly from https://shanetully.com/2014/09/a-dead-simple-webrtc-example/, so be sure to read the tutorial there before you proceed.
2. Close down your server from the previous task and navigate to the “https” folder and run `node https_server.js`
3. Try out the two-party WebRTC video chat: https://localhost:8443/task3.html
4. See if you can connect two clients. Use ifconfig (Mac) or ipconfig (Windows) to find the IP address of the computer you are running the server on. Then use two clients (for example a browser on your host computer and a phone) to connect to your server and instantiate a video call.

Fourth Task

1. More advanced filters:
   https://github.com/brianchirls/Seriously.js/wiki/Tutorial Start out with the file task4.html.
2. Try and use Seriously.js filters (instead of CSS like in the previous example) in your video chat.
3. Create your own green-screened video stream by using the Chroma Key filter to replace the background in your video chat.
4. Hints:
   hint 1: check out https://localhost:8443/seriouslyexample.html to see how to apply effects to video.
   hint 2: use the 'chroma' effect on the video, then 'blend' with an image

Links

getUserMedia docs:
https://developer.mozilla.org/en/docs/Web/API/Navigator/getUserMedia

Seriously.js docs: https://github.com/brianchirls/Seriously.js/wiki/Tutorial

WebRTC with https: https://shanetully.com/2014/09/a-dead-simple-webrtc-example/,


PROTOTYP