Multimedia im Netz (Online Multimedia)

Wintersemester 2014/15

Übung 10 (Hauptfach)

Today‘s Agenda

- Quiz (Tutorial & Lecture)
- Discussion of assignment 08
- Media Streaming in the Browser
  - MediaStream API (aka getUserMedia)
  - WebRTC
  - Socket.IO
  - PeerChat
- Mash-Ups Presentation

Today‘s Agenda

- **Quiz (Tutorial & Lecture)**
- Discussion of assignment 08
- Media Streaming in the Browser
  - MediaStream API (aka getUserMedia)
  - WebRTC
  - Socket.IO
  - PeerChat
- Mash-Ups Presentation

Quiz – Part 1 - Tutorial

1. How do you add dependencies to a node web app?
2. Is JSON a human readable data format?
3. What are the advantages of JSON over XML?
4. Does every node web-app listen on its own port?
5. What does the body-parser middleware do?
6. What is serialization and why is it necessary?

Quiz – Part 2 – Lecture

1. Which is more recommendable regarding video streaming: TCP or UDP? Explain your answer.
2. What does „Jitter“ mean?
3. What does a Content Delivery Network (CDN) do?
4. Does video streaming on YouTube follow the Push or Pull model? Explain how it works in general.
5. What is the main idea behind adaptive video streaming?

Today‘s Agenda

- Quiz (Tutorial & Lecture)
- **Discussion of assignment 08**
- Media Streaming in the Browser
  - MediaStream API (aka getUserMedia)
  - WebRTC
  - Socket.IO
  - PeerChat
- Mash-Ups Presentation

Assignment 8: AngularJS vs. jQuery

- AngularJS core assets:
  → two-way databinding vs. callbacks
  → Model-View-Whatever pattern (≈ MVC)
  → Framework vs. Library
  → Templates
  → Modules and Dependency Injection
- All this makes AngularJS somewhat more dynamic and more flexible than jQuery
- Drawback: Higher abstraction level might be more difficult to adapt to

Assignment 8: MEAN vs. LAMP

- A client asks you to set up a company blog on the website. There are around 10 different authors, and the blog will be read by 2000 visitors per month.
  - not many visitors (2000/30 ≈ 70 per day) – LAMP still scales well to this point.
  - LAMP tools and templates readily available and easy to set up (e.g. WordPress)

  ➔ Stick with LAMP

Assignment 8: MEAN vs. LAMP

- You want to create a site that queries Twitter Tweets containing a certain hashtag and display those tweets immediately on the website.
  - „real time“ requirement is well supported by MEAN, because of its asynchronous nature.
  - simple and small AJAX calls natively supported by MEAN

  ➔ Use MEAN

Assignment 8: MEAN vs. LAMP

- You have an idea for a new portal that compares flight prices. More than 2 Million visitors are expected each month.
  - Heavy-load web app that requires scalable infrastructure
  - MySQL does not scale well to that point
  - asynchronous calls all the way (query many different flight operators' sites)
  - bidirectional communication very likely

  ➔ Use MEAN

Assignment 8: MEAN vs. LAMP

- You offer a service similar to Google Keep, where people can take notes and create simple to-do-lists.
  - Depending on the user count, this might be heavy duty.
  - Simple requirement → simple, well manageable architecture
  - No binary data in databases

➔ MEAN or LAMP both work fine. Slight tendency towards MEAN, because of the asynchronous aspect (easier collaboration, if necessary)

Assignment 8: MEAN vs. LAMP

- You want to create the next big social network that challenges Facebook
  - Better choose your technology well ;)
  - MEAN is still on the rise
  - LAMP might be too weak to handle a 'Facebook Killer' although much of Facebook's infrastructure relies on PHP.
  - MEAN needs to prove in the long run

➔ Better choose something that's up to speed with current developments instead of riding an old horse ;)

Today‘s Agenda

- Quiz (Tutorial & Lecture)
- Discussion of assignment 08
- **Media Streaming in the Browser**
  - MediaStream API (aka getUserMedia)
  - WebRTC
  - Socket.IO
  - PeerChat
- Mash-Ups Presentation

Video and Audio in the Browser

- Simple, unidirectional streaming possible with the `<video>` and `<audio>` tags
- Before HTML5, browser plugins were necessary to play multimedia content
- MP3/MP4 is supported in most browsers:

```
<audio src="./sound.m4a"></audio>
```

How do we achieve multimedia input?

HTML5, JavaScript and Media Capture

- Goal: access audio/video through HTML
- Capturing used to rely on plugins (e.g. Flash, Silverlight, Quicktime, Java Applets)
- HTML5 brings audio/video capturing functionality!
- Part of the WebRTC effort (more on this in a moment)

Media Stream API

- Synchronized streams of media, i.e. synchronized audio and video streams.
- Streams have inputs (e.g. camera) and an output
- Most important method:

```
navigator.getUserMedia(
    constraints,
    successCallback,
    errorCallback
)
```

- MediaStream Object returned to successCallback:

```
{   "onremovetrack" : null,
    "onaddtrack":null,
    "onended":null,
    "ended":false,
    "id":"3e7BpY1juXXwO2roIY9BgHBzTy6kMwCyyr4B",
    "label":"3e7BpY1juXXwO2roIY9BgHBzTy6kMwCyyr4B"
}
```

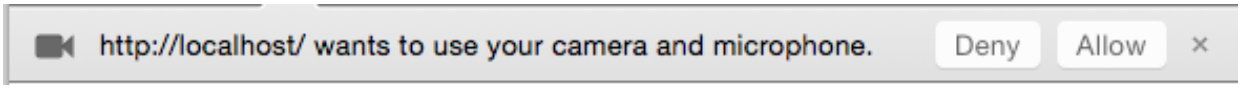
Media capture availability check

```
<script>
    function isUserMediaCapable() {
        return !!(
                navigator.getUserMedia ||
                navigator.webkitGetUserMedia ||
                navigator.mozGetUserMedia ||
                navigator.msGetUserMedia);
    }

    if (isUserMediaCapable()) {
        document.write("getUserMedia is supported");
    } else {
        alert('getUserMedia() not supported');
    }
</script>
```

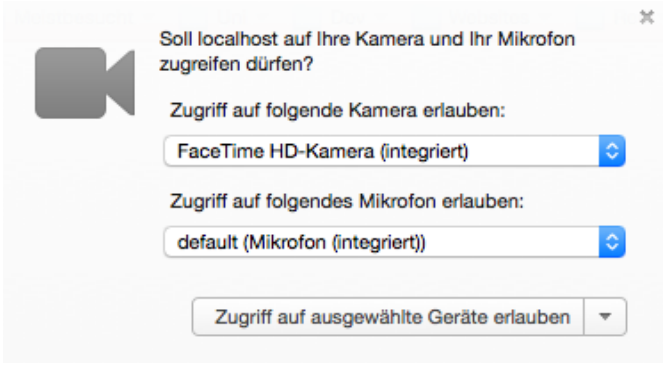
Permissions

- getUserMedia() usually generates a built-in browser dialog

Chrome:



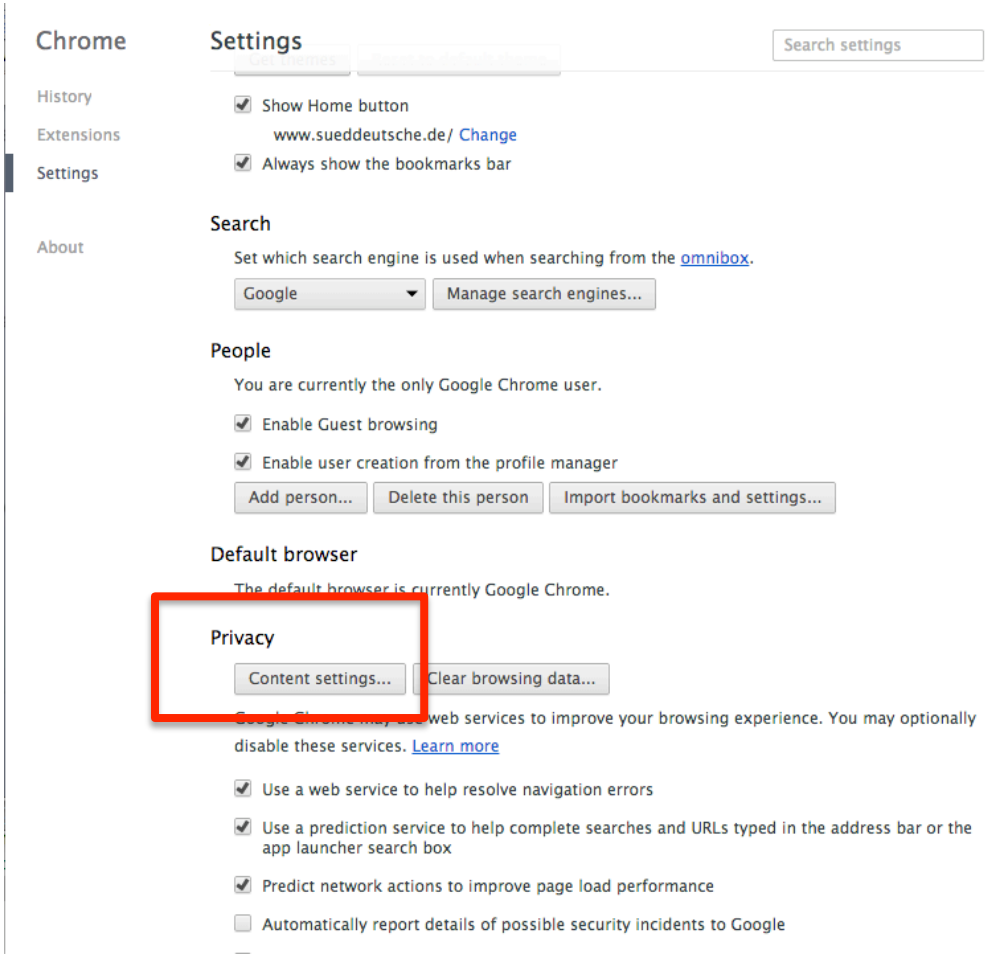
Firefox:



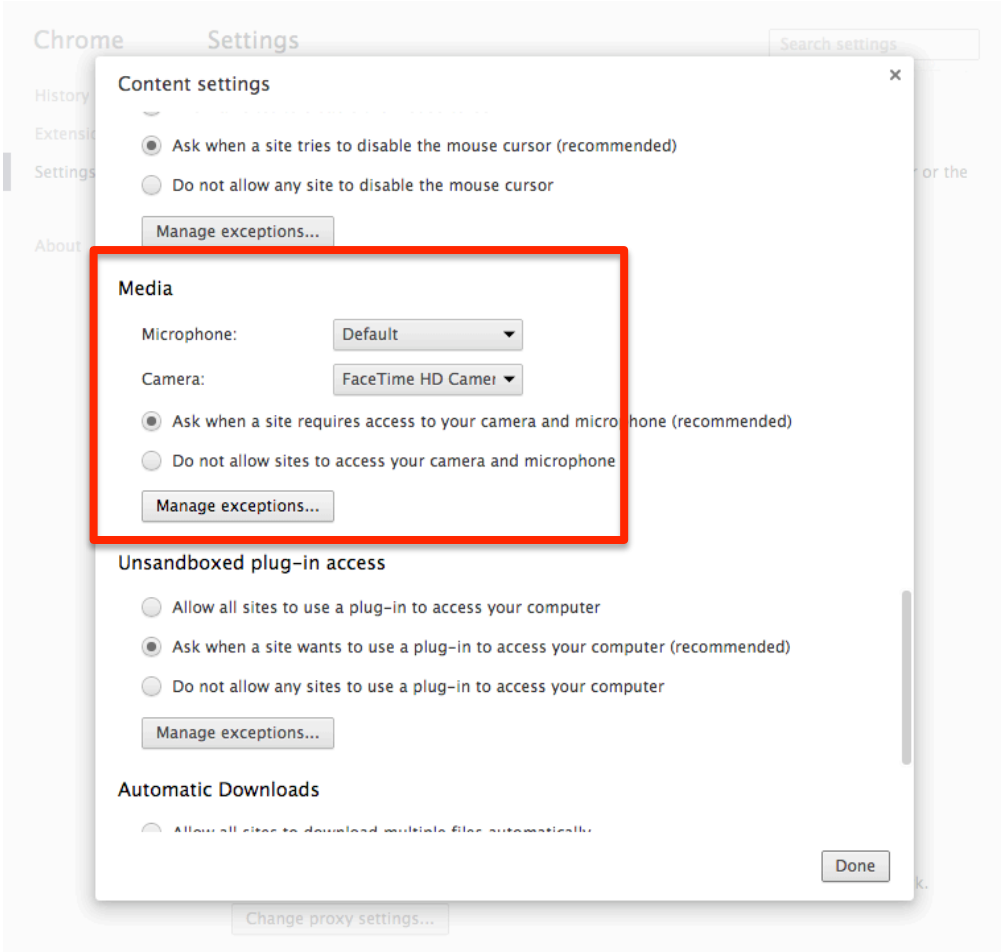
- Permissions are requested with a JSON Object as first parameter:

```
var requestedPermissions = {
    audio: true,
    video: true
};
```

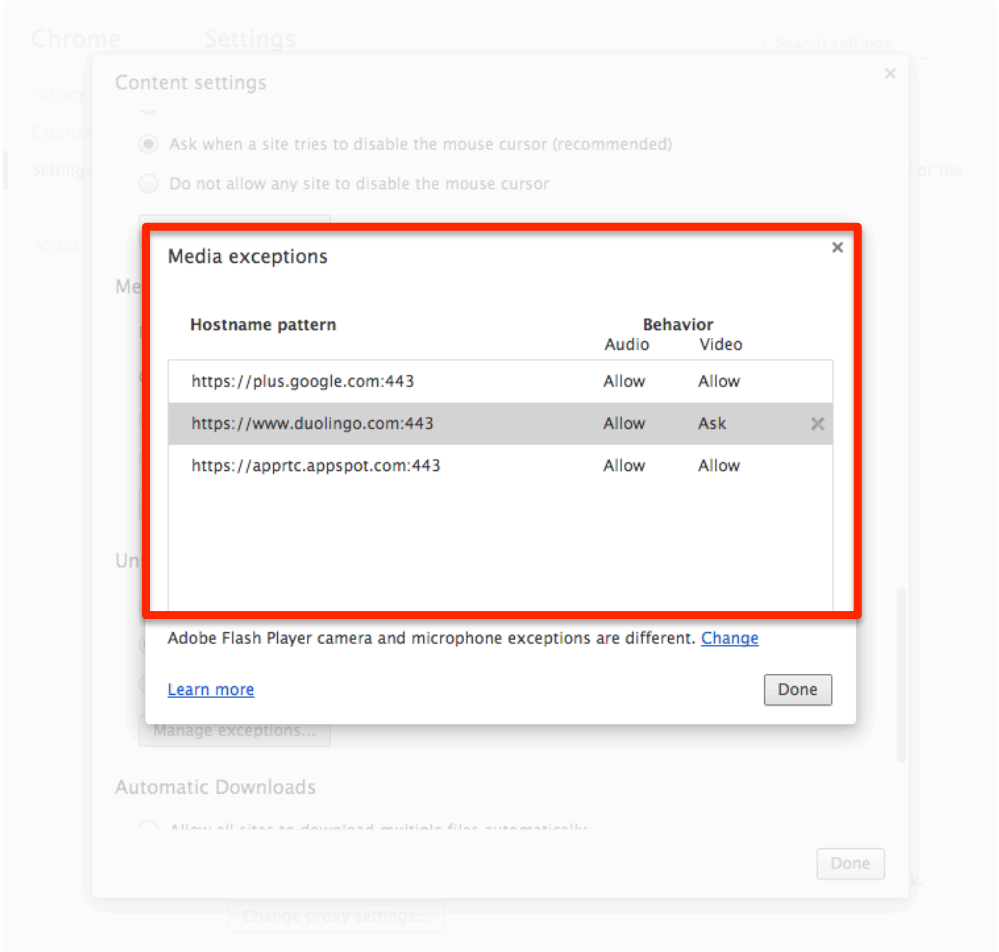
Reset Permissions in Chrome (1)



Reset Permissions in Chrome (2)



Reset Permissions in Chrome (3)



Simple A/V Capture Script

```
<video autoplay></video>
<script>
    // Source : http://goo.gl/Bc8JN
    // allows us to use .getUserMedia across browsers.
    navigator.getUserMedia = navigator.getUserMedia ||
    navigator.webkitGetUserMedia ||
    navigator.mozGetUserMedia ||
    navigator.msGetUserMedia;

    // get the first video element in the DOM
    var video = document.querySelector('video');
    if (navigator.getUserMedia) {
        navigator.getUserMedia({
                audio: true,
                video: true
            },
            function (stream) {
                video.src = window.URL.createObjectURL(stream)
            });
    }
</script>
```

Constraints

- Instead of simply requesting a permission, we can request even more settings, e.g. the video resolution:

```
var hdConstraints = {
    video: {
        mandatory: {
            minWidth: 1920,
            minHeight: 1080
        }
    }
};

navigator.getUserMedia(hdConstraints, ...)
```

- All supported constraints: https://w3c.github.io/mediacapture-main/getusermedia.html#idl-def-MediaTrackConstraints

Taking Video Snapshots

```
var video = document.querySelector('video');
var canvas = document.querySelector('canvas');
var ctx = canvas.getContext('2d');


function snapshot() {
    ctx.drawImage(video, 0, 0);
    document.querySelector('img').src =
        canvas.toDataURL('image/webp');
}

navigator.getUserMedia(hdConstraints, function(stream) {
    video.src = window.URL.createObjectURL(stream);
    video.addEventListener('click', snapshot, false);
});
```

Interactivity

- Play a „camera shutter“ sound when the user takes a snapshot!
- A sound file is included in the material for this tutorial
- Sound file source (Creative Commons): https://www.freesound.org/people/xef6/sounds/61059/

WebRTC

- Goal: **r**eal **t**ime **c**ommunication without browser plugins (like Adobe Flash, Apple QuickTime, Facebook, Skype, Google Talk ...)
- Audio, Video or simply ***data***.
- Tutorial: http://www.html5rocks.com/en/tutorials/webrtc/basics/
- Currently supported Browsers (Jan 2015):

- Mobile OS:

WebRTC Challenges

- Stream audio, video and other data
- Get network info (IP adresses, ports) to other clients through firewalls
- Coordinate sessions
  - signal
  - initiate
  - close
- And more...
- WebRTC implements the APIs to take the challenges:
  - MediaStream
  - RTCPeerConnection
  - RTCDataChannel

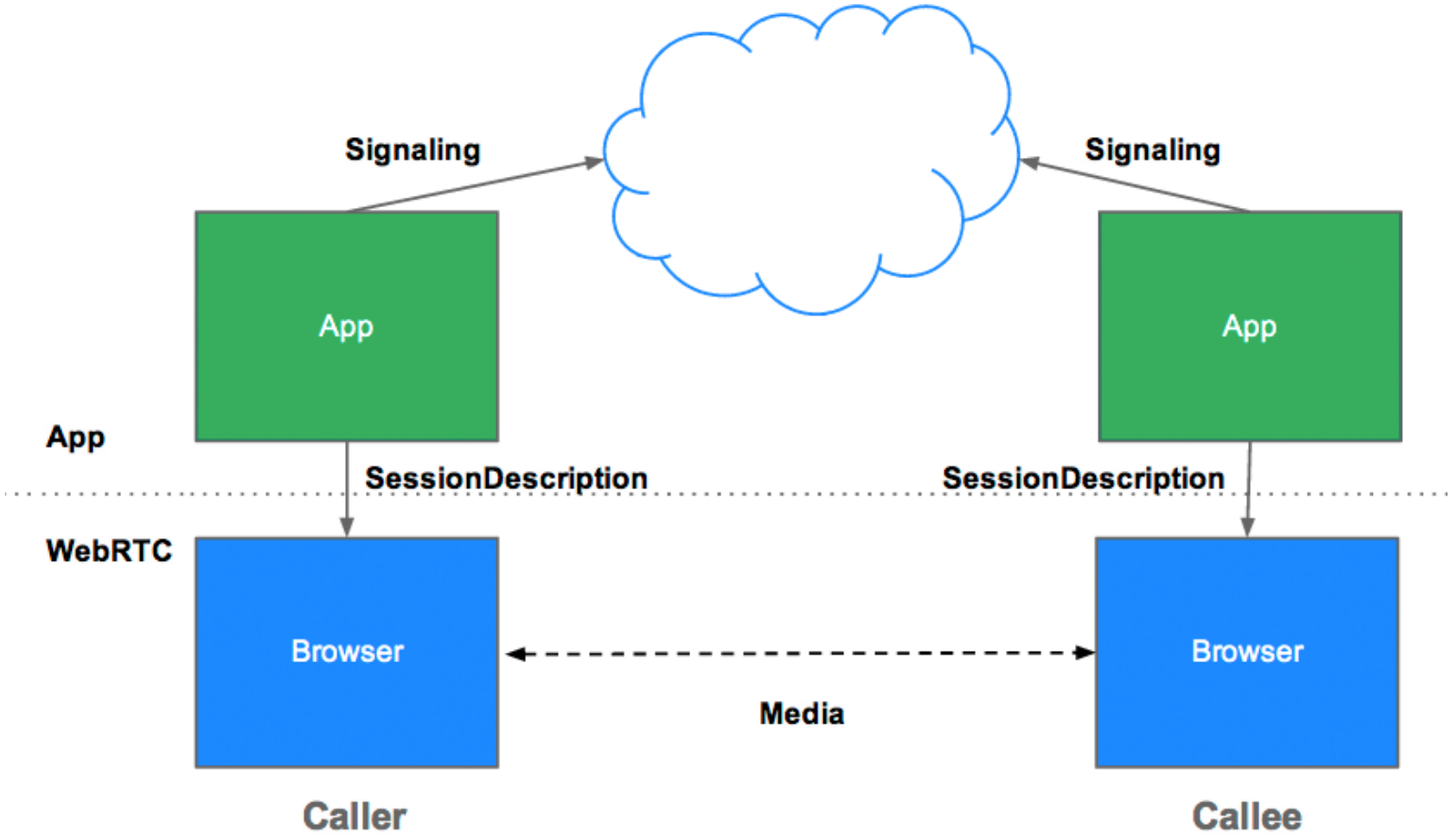
WebRTC Challenges: Signaling

- Signaling:
  - session control messages, network coordination, media capabilities, meta-data, error messages
  - Choose your own messaging protocol, e.g. SDP, SIP, XMPP and bi-directional communication channel, e.g. Socket.io

- Example: http://www.simpl.info/rtcpeerconnection/

- Signaling isn‘t trivial. Studying examples is recommended!

Signaling



http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/?redirect_from_locale=de#what-is-signaling

RTCPeerConnection API

- Does the heavy lifting for peer-to-peer connections
- Stable and efficient communication of streaming data
- Responsibilites:
  - packet loss concealment
  - echo cancellation
  - bandwidth adaptivity
  - dynamic jitter buffering
  - automatic gain control
  - noise reduction and suppression
  - image 'cleaning'.

http://www.html5rocks.com/en/tutorials/webrtc/basics/#toc-rtcpeerconnection

RTCPeerConnection Methods

- obtain RTCPeerConnection object: `new RTCPeerConnection();`
  - currently, browser prefixes are required.
    `new webkitRTCPeerConnection(server,constraints)`
  - Attach event handlers to this object, e.g. onaddstream

- Call:
  `pc.createOffer(successHandler,errorHandler,constraints)`
- Answer:
  `pc.createAnswer(successHandler,errorHandler,constraints)`

- ICE Framework methods, e.g.
  `pc.addIceCandidate(candidate);`

https://developer.mozilla.org/en-US/docs/Web/API/RTCPeerConnection#Basic_Usage

More WebRTC

- Further links:
  - http://www.webrtc.org/ (Official Webpage)
  - https://github.com/GoogleChrome/webrtc (various WebRTC Samples)
  - https://bitbucket.org/webrtc/codelab (Video-chat code example with NodeJS and Socket.io)
  - https://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment (ICE Protocol)
  - http://simplewebrtc.com/ (WebRTC library)
  - https://talky.io/ (Video chat service powered by WebRTC)

Socket.IO

socket.io

- NodeJS module for event-based, bidirectional communication
- (near) real-time communication
- Cross-Platform support
- Allows binary streaming (important for video/audio!)
- Two components:
  - Server side module (NodeJS)
  - Client side script (JavaScript)

Socket.io (server)

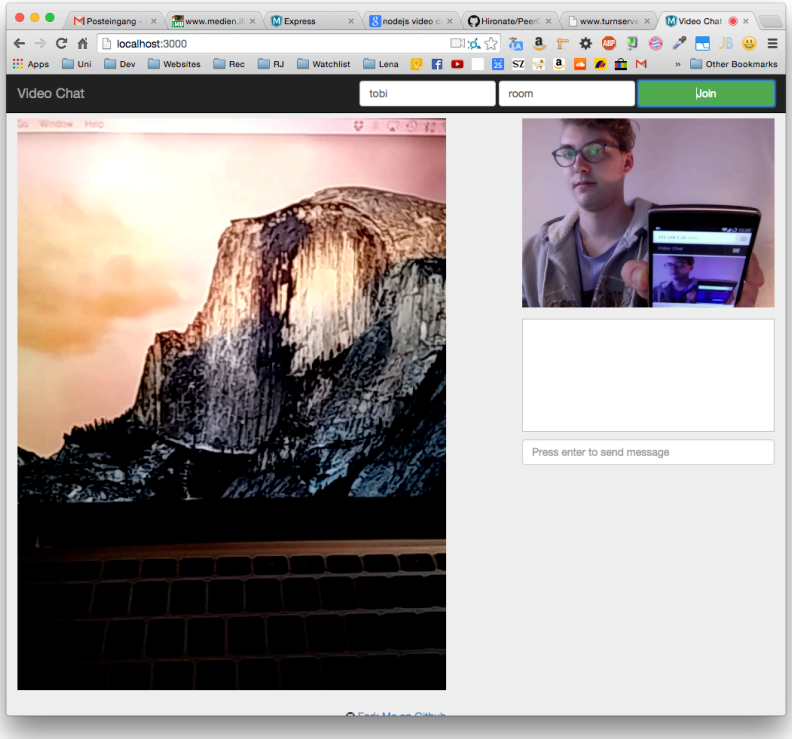
```
var app = require('express')();
var server = require('http').Server(app);
var io = require('socket.io')(server);

server.listen(3000);

app.get('/', function (req, res) {
    res.sendFile(__dirname + '/index.html');
});

io.on('connection', function (socket) {
    socket.emit('news', { hi: 'welcome to the chat!' });
    socket.on('clientMessage', function (data) {
        console.log(data);
        io.emit('news',data);
    });
});
```

Socket.io (client)

```
<script>

    var socket = io.connect();

    socket.on('news', function (data) {
        console.log(data);
    });

    socket.emit('clientMessage','test message');
</script>
```

PeerChat by H. Kavad

- NodeJS implementation of a WebRTC compliant video chat
- Relies on NodeJS and Socket.io for signaling
- Clone: https://github.com/Hironate/PeerChat
- The API Key mentioned in the Readme file is not necessary if you do not deploy the app to a production environment



Assignment 10

- **Topic**: Browser PhotoBooth, PeerChat modification
- **Due in**: 1 Week
- **Due date**: 19.01.2015

Announcements

- Remaining tutorial dates / topics
  - 19.01.2015: Repetition
  - 26.01.2015: Q&A
- There will be another assignment next week
- Exam:
  - 16.02.2015 14-16h, Rooms A140 & A240 (Main Building)
  - register until: 09.02.2015
  - de-register until: 13.02.2015
  - If you do not de-register, the attempt will be treated as “failed”

Thanks!

What are your questions?