Today’s Agenda

- Quicktest 😊
- Media Input without Browser Plugins
- WebSocket
- Quiz
Quick Test #03
Media Input without Browser Plugins
Video and Audio in 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:
  ```html
  <audio src="/sound.m4a"></audio>
  <video controls width="250">
    <source src="/video.webm" type="video/webm">
    <source src="/video.mp4" type="video/mp4">
  </video>
  Video element is not supported.
  ```

For more information, visit: [https://caniuse.com/#feat=video](https://caniuse.com/#feat=video)
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 future tutorials)
Media Devices API

- Synchronized media streams, i.e. synchronized audio & video streams in the Browser.
- Streams have inputs (e.g. camera) and an output
- Most important method:
  ```javascript
  navigator.mediaDevices
  .getUserMedia(constraints).then(successCallback).catch(errorCallback)
  ```
- MediaStream object returned to successCallback:
  ```javascript
  {
    active: true,
    id: "Cp7wyEAYPZCFTzFkUDjN9R2xxzRqo4fzaueW",
    onactive: null,
    ...
  }
  ```
MediaDevices API Browser Support

https://caniuse.com/#search=getusermedia

<table>
<thead>
<tr>
<th>Current aligned</th>
<th>Usage relative</th>
<th>Date relative</th>
<th>Apply filters</th>
<th>Show all</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>Edge</td>
<td>Firefox</td>
<td>Chrome</td>
<td>Safari</td>
</tr>
<tr>
<td>2-16</td>
<td>17-35</td>
<td>4-20</td>
<td>3.1-10.1</td>
<td>10-11.5</td>
</tr>
<tr>
<td>17-35</td>
<td>36-41</td>
<td>42-70</td>
<td>21-52</td>
<td>15-17</td>
</tr>
<tr>
<td>36-41</td>
<td>53-78</td>
<td>79</td>
<td>11-12.1</td>
<td>40-63</td>
</tr>
<tr>
<td>53-78</td>
<td>80-82</td>
<td>TP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

https://caniuse.com/#search=getusermedia
Availability Check

```html
<script>
    function isMediaDevicesCapable() {
        return navigator.mediaDevices &&
            navigator.mediaDevices.getUserMedia;
    }
    if (isMediaDevicesCapable()) {
        document.write('mediaDevices.getUserMedia supported');
    } else {
        document.write('sorry, mediaDevices.getUserMedia unsupported');
    }
</script>
```
Permissions

- `getUserMedia()` usually generates a built-in browser dialog
- Permissions are requested with a JSON Object as first parameter:

```javascript
let requestedPermissions = {
    audio: true,
    video: true
};
```
Simple A/V Capture Script

```html
<script>
    let video = document.querySelector('video');
    if (navigator.mediaDevices.getUserMedia) {
        navigator.mediaDevices.getUserMedia({ audio: true, video: true })
            .then(function(stream) {
                video.srcObject = stream;
            }).catch(function(error) {
                document.write(error);
            });
</script>
```
Taking Video Snapshots

```html
<video autoplay></video>

<script>
let video = document.querySelector('video'), canvas;

navigator.mediaDevices.getUserMedia({video: true}).then(function(stream) {
    video.src = stream;
    video.addEventListener('click', () => {
        let img = document.querySelector('img') || document.createElement('img');
        let width = video.offsetWidth, height = video.offsetHeight, context;
        canvas = document.createElement('canvas');
        canvas.width = width; canvas.height = height;
        context = canvas.getContext('2d');
        context.drawImage(video, 0, 0, width, height);
        img.src = canvas.toDataURL('image/png');
        document.body.appendChild(img);
    });
});
</script>
```

03-snapshots.html
Breakout

• 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/
WebSocket
WebSocket

- Basic ideas
  - TCP based, bidirectional, full-duplex messaging
  - Establish connection (single TCP connection)
    - **Bidirectional**: Send messages in both direction
    - **Full Duplex**: Send message each other independently
- Allow you stream data to and from web browsers.
- The socket starts out as a HTTP connection and then "Upgrades" to a TCP socket after a HTTP handshake. After the handshake, either side can send data.
WebSocket

- Communication protocol used to send/receive data on the Internet
- Like HTTP, but on steroids… WebSockets are wa more efficient
- Persistent 2-way connection between <-- server
- Easy to build real-time applications:
  - Chat & Conferencing
  - Notifications
  - Online games
  - Financial trading
  - Live maps
  - ...

WEBSOCKETS

WEBSOCKETS EVERYWHERE
Drawbacks

- Poor browser supports
- More complex handling both on client and server side, e.g. handling reconnection
- Eat massive memory to maintain a lots of connections
  - Well-designed client side polling sometimes costs lower pressure on the server depending on the actual business
Socket.IO

- NodeJS module for event-based, bidirectional communication
- Encapsulates WebSocket and eliminates low-level details
- (near) real-time communication
- Cross-Platform support, automatically select the best approach
- **Allow binary streaming (important for video/audio)**
- Two components:
  - Server side module (NodeJS)
  - Client side script (JavaScript)
Socket.IO (server)

```javascript
var app = require('express')();
var http = require('http').createServer(app);
var io = require('socket.io')(http);
app.get('/', function(req, res) {
  res.sendFile(__dirname + '/index.html');
});
io.on('connection', function(socket) {
  socket.on('from-client', function(msg) {
    io.emit('from-server', msg + ', again');
  });
});
https.listen(3000);
```
Socket.IO (client)

```javascript
<script>
let socket = io()
socket.emit('from-client', 'hello')
socket.on('from-server', (data) => {
    console.log(data) // response 'hello, again'
})
</script>
```
Code Alone: Twitch MIMUC over WS

- Use MediaDevices API and Socket.IO to implement a naive Twitch server :)

- Basic idea:
  - **Anchor** sends its video (a series of images) and audio via WebSocket to the server
  - **Server** replicates the video and audio then broadcasts (**push**) to **viewers**
Issue: The Connection is Not Private
TIP: Resolving HTTPS locally

- Media Devices API is only allowed in file:// or HTTPS environment, which means you must setup your express server to support HTTPS, this is usually done by server side proxy, e.g. NGINX. A quick solution:
- **Step 1:** Generate a self-signed HTTPS certificate, or use our provided certificate (see Github repo)

  ```
  openssl req -nodes -new -x509 -keyout server.key -out server.cert
  ```

- **Step 2:** Add .cert file to your system as a trusted certificate
- **Step 3:** In your express server, use https:
  ```
  var https = require('https').createServer({
    key: fs.readFileSync('server.key'),
    cert: fs.readFileSync('server.cert')
  }, app);
  var io = require('socket.io')(https);
  ...
  https.listen(port, function(){
    console.log('listening on *:' + port);
  });
  ```

macOS as example
Anchor:

Streaming: ON

Viewer:
Reminder: Register Exam

● Exam
  ○ Date: February 11th 2020, 12:00-2:00 PM
  ○ Location: LMU main building, Geschwister-Scholl-Platz 1, Room M218

● Register yourself via Uni2Work before 2020-02-05 23:59:00
Thanks!

What are your questions?