LABORATORIO DI REALTÀ AUMENTATA

Claudio Piciarelli

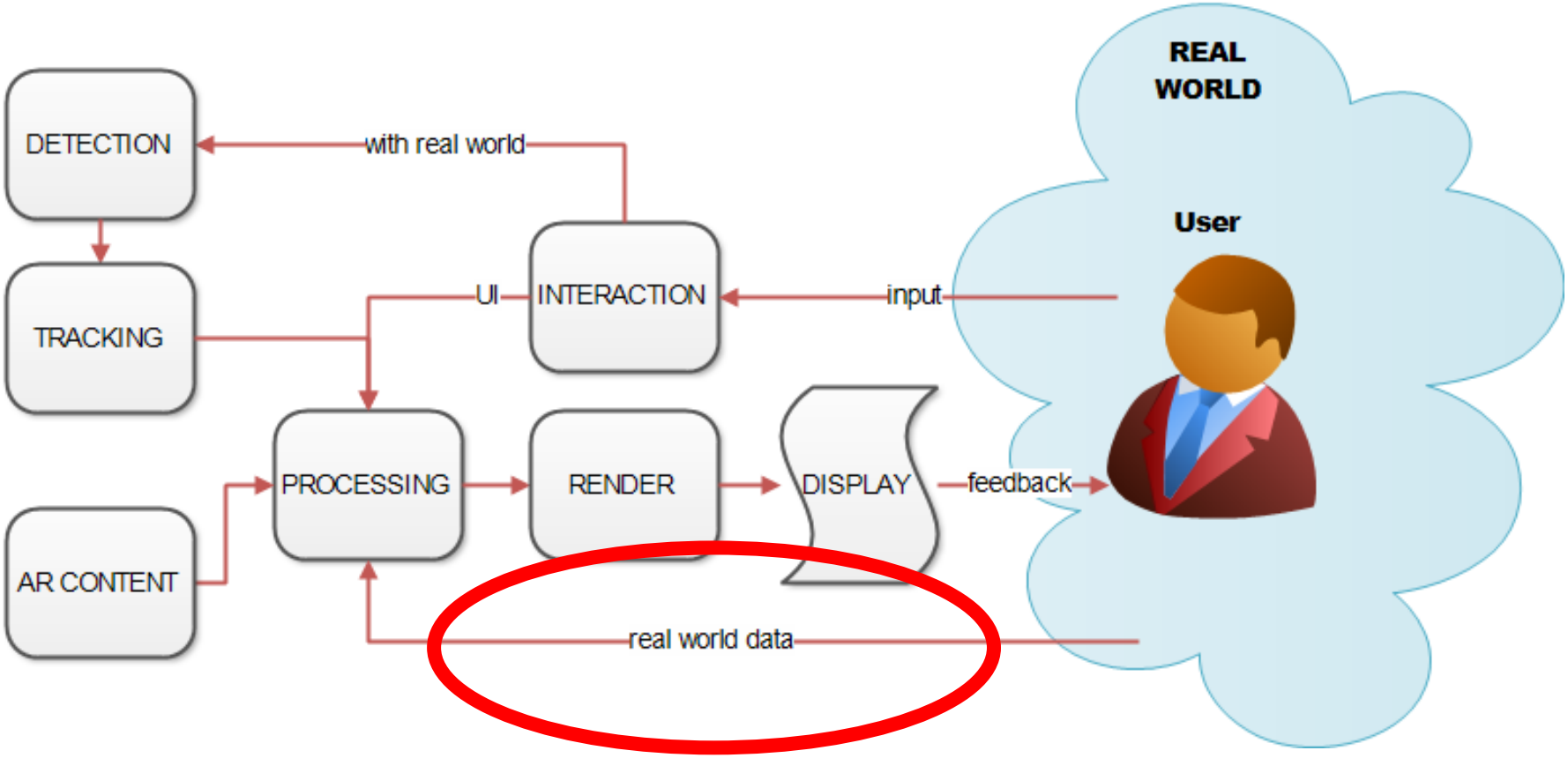
Università degli Studi di Udine
Corso di Laurea in Scienze e Tecnologie Multimediali

Project: video input

Prerequisites

- What do you need:
  - A WebRTC-enabled browser (I will use Firefox)
    - http://iswebrtcreadyyet.com/
  - Mac users: Safari support still limited
  - A text editor, e.g. Notepad++ (has syntax highlighting and a very basic javascript completion)
  - You can use your preferred editor, though

Architecture of an AR system



STM AR: video input

- Idea: acquire a video stream from a webcam using a standard browser
- If you don’t have a webcam, you can work on videos
- Alternative solution: use a webcam simulator software (e.g. manycam)

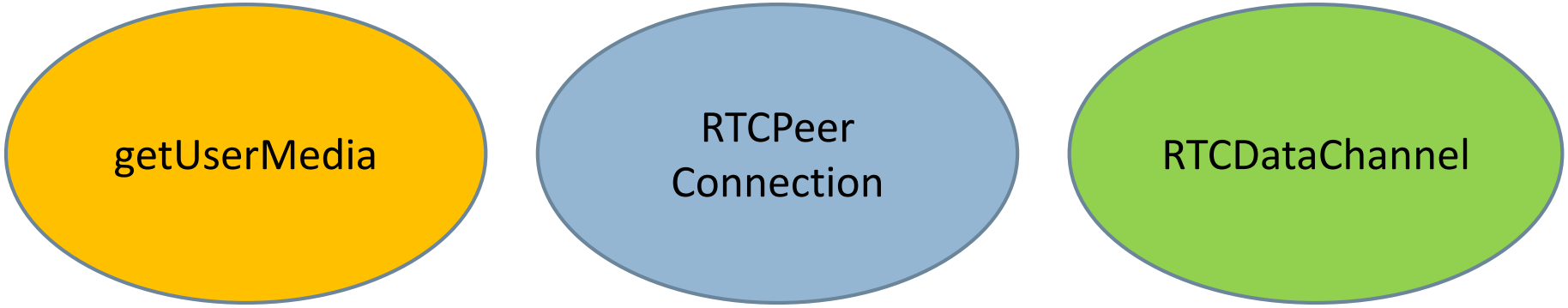
WebRTC

- **WebRTC is a free, open project** that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. The WebRTC components have been optimized to best serve this purpose.
- **Mission:** To enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices, and allow them all to communicate via a common set of protocols.

From: https://webrtc.org

WebRTC

- Studied for browser-to-browser real-time applications (voice calls, video chats, P2P file sharing…)
- Three main components



Main WebRTC components

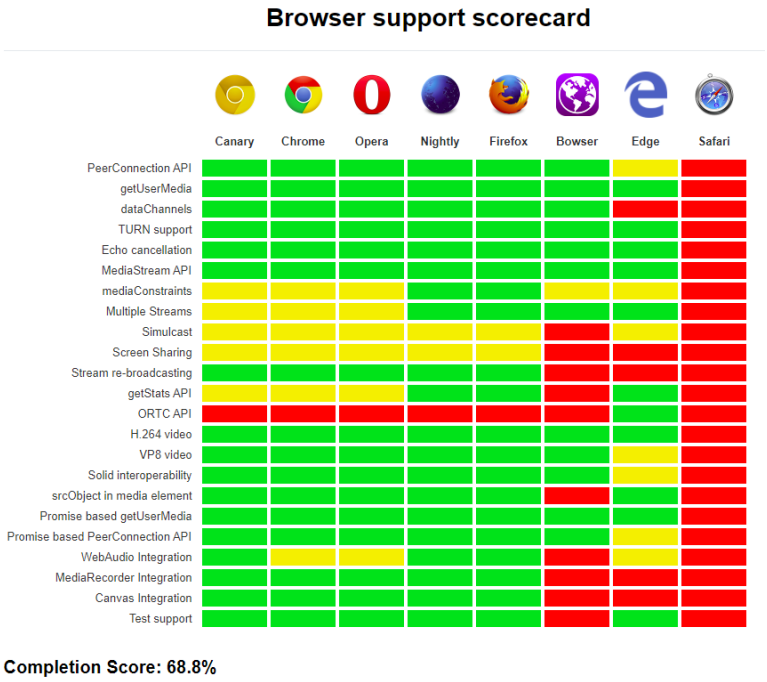
- **getUserMedia:** allows the browser to access local media (cameras and microphones)
- **RTCPeerConnection:** sets up browser-to-browser connections
- **RTCDataChannel:** allows browsers to share data/streams via peer-to-peer

- We will only use getUserMedia to acquire the webcam stream

Supported browsers

- Edge, Chrome, Firefox, Opera…
- Unfortunately, Safari still lacks native WebRTC support (but external plugins exist).

- http://iswebrtcreadyyet.com/



Security issues

- Many browsers are enforcing security policies to enhance web security
- getUserMedia() could be disabled on insecure (`http://`) sources

- Workaround:
  - Use `https://` web sites
  - Localhost is considered a secure source, thus your local http server should work

Preparing the web page

- To start, we just need an HTML5 web page with an empty video element

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Webcam access with WebRTC</title>
</head>
<body>
    <video id="myvideo"></video>
</body>
</html>
```

Preparing the script part

- Good practice: execute the scripts only when the web page has fully loaded:

```
<!doctype html>
<html>
<head>
    <title>Webcam access with WebRTC</title>
    <script>
    window.onload = function(){
        // put your code here!
    }
    </script>
</head>
<body>
    <video id="myvideo"></video>
</body>
</html>
```

Where is `getUserMedia()`?

- Standard implementation:
  - `navigator.mediaDevices.getUserMedia()`

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox Android	Opera Android	Safari on iOS	Samsung Internet
`getUserMedia`	53 ★ ▼	12	36 ★ ▼	No	40 ★ ▼	11	53	53 ★ ▼	36 ★ ▼	41 ★ ▼	11	6.0
Secure context required	53	79	68	No	40	?	53	53	68	41	?	6.0

Image from: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia

- Legacy:
  - `navigator.getUserMedia()`

Which `getUserMedia()` should we use?

- We will rely on the new standard implementation
- Supported by many recent browsers (Firefox >36, current is 86; Chrome >53, current is 89)

How to use `getUserMedia()`

- Syntax, using javascript Promises:

```
navigator.mediaDevices.getUserMedia(constraints)
.then(
        /* we need a function here */
).catch(
        /* we need a function here */
);
```

How to use `getUserMedia()`

- Syntax:

```
navigator.mediaDevices.getUserMedia(constraints)
.then(function(stream) {
  /* use the stream */
}).catch(function(err) {
  /* handle the error */
});
```

- Alternative syntax, with arrow functions:

```
navigator.mediaDevices.getUserMedia(constraints)
.then(stream => {
  /* use the stream */
}).catch(err => {
  /* handle the error */
});
```

Constraints

- Constraints are javascript Objects
- Basic constraints: enable / disable audio and video

```
var constraints = { audio: true, video: true }
```

- Video size hints (not mandatory)

```
{
  audio: true,
  video: { width: 1280, height: 720 }
}
```

Constraints

- Video size: mixing hints and mandatory requests

```
{
  audio: true,
  video: {
    width: { min: 1024, ideal: 1280, max: 1920 },
    height: { min: 700, ideal: 720, max: 1080 }
  }
}
```

- Video size: forcing a specific resolution

```
{
  audio: true,
  video: {
        width: { exact: 1280 },
        height: { exact: 720 }
  }
}
```

General rule:
Only **`min`**, **`max`** and **`exact`** are mandatory requests

Constraints

- Prefer frontal camera (if available) on mobile devices

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

- Force rear camera:

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

Error handling

- In case of errors notify the user, either using `alert()` or `console.log()` and fallback to video file

```
var video = document.getElementById("myvideo");

navigator.mediaDevices.getUserMedia(constraints)
.then(function(stream) {
  /* use the stream */
}).catch(function(err) {
  alert(err.name + ": " + err.message);
  video.src = "marker.webm";
});
```

Using the video stream

```
var video = document.getElementById("myvideo");

navigator.mediaDevices.getUserMedia(constraints)
.then(function(stream){
        video.srcObject = stream;
})
.catch(function(err){
        alert(err.name + ": " + err.message);
        video.src = "marker.webm";
});
```

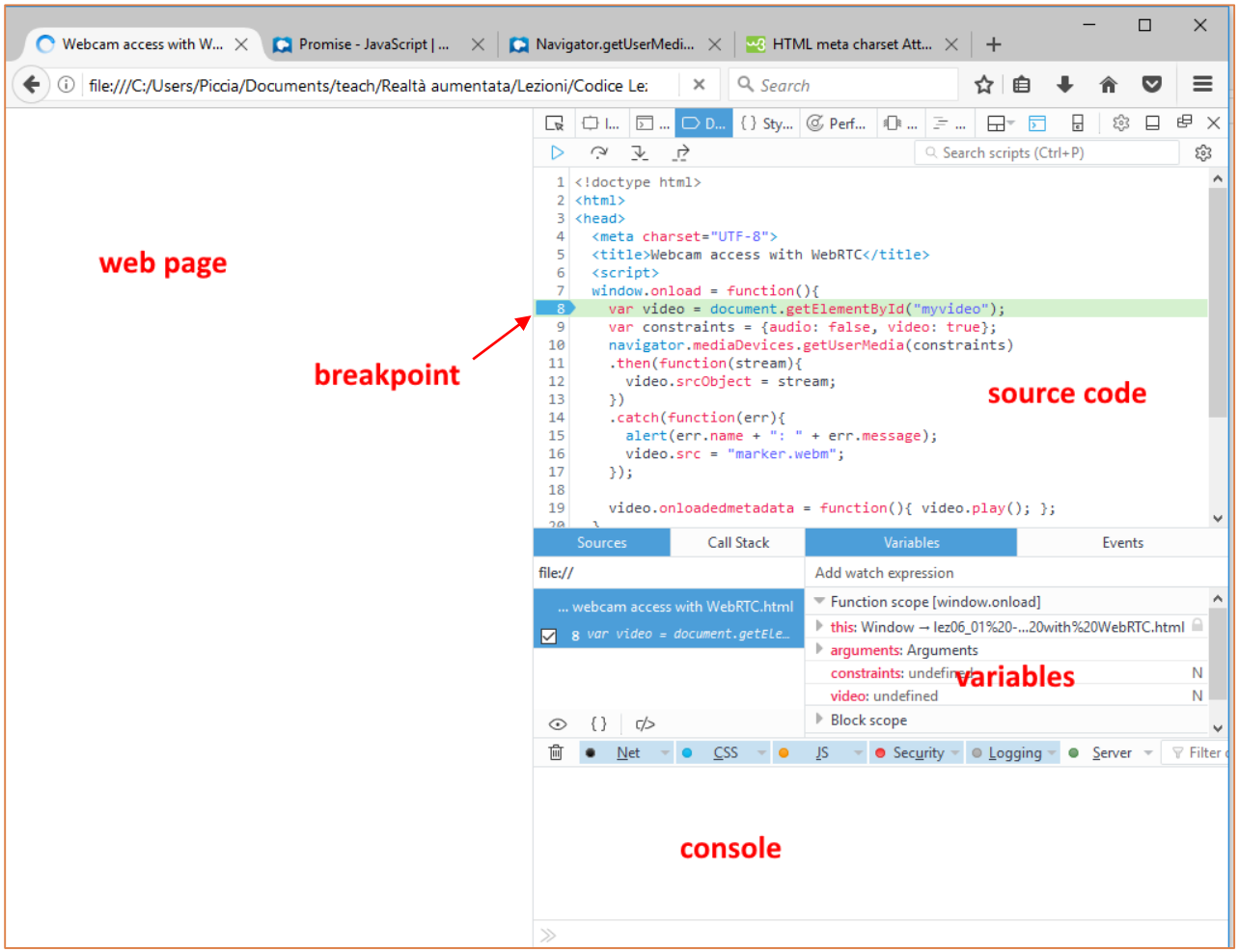
Final code:

```
<!doctype html>
<html>
<head>
        <meta charset="UTF-8">
        <title>Webcam access with WebRTC</title>
        <script>
        window.onload = function(){
                var video = document.getElementById("myvideo");
                var constraints = {audio: false, video: true};
                navigator.mediaDevices.getUserMedia(constraints)
                .then(function(stream){
                        video.srcObject = stream;
                })
                .catch(function(err){
                        alert(err.name + ": " + err.message);
                        video.src = "marker.webm";
                });

                video.onloadedmetadata = function(){ video.play(); };
        }
        </script>
</head>
<body>
        <video id="myvideo"></video>
</body>
</html>
```

Something not working?

- Right-click on the page, then choose “inspect element”



Using the video

- The `<video>` obtained by the webcam acts as any other video. For example, you can try acquiring snapshots or playing with CSS filters as in the previous lesson

Additional info

- https://webrtc.org/getting-started/media-devices?hl=en
- https://webrtc.org/getting-started/media-capture-and-constraints?hl=en