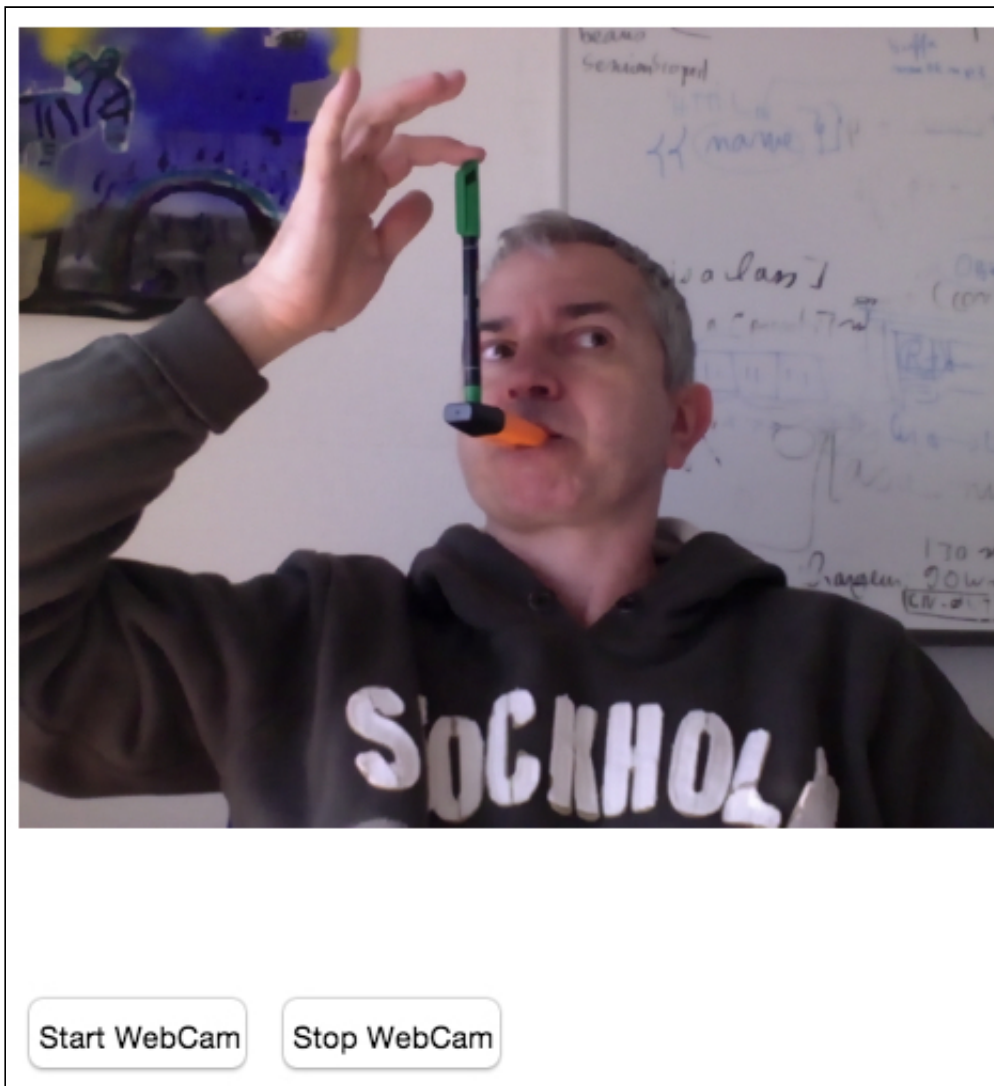


# More on getUserMedia

Let's see some more examples of what we can do with the getUserMedia API: start/stop the webcam, take a screenshot from the current video stream from the webcam, and apply CSS effects in real time. And at the end, we give links to some cool examples available on the Web.

## HOW TO STOP/RELEASE THE WEBCAM



[Online version at JS Bin](#)

In order to stop the webcam and make the hardware "unlock it", you need to call

the `stop()` method of the video stream.

Modified version of the previous example:

```
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
  <script>
    navigator.getUserMedia = (navigator.getUserMedia ||
    navigator.webkitGetUserMedia ||
    navigator.mozGetUserMedia ||
    navigator.msGetUserMedia);

    var webcamStream;

12.   function startWebCam() {
        if (navigator.getUserMedia) {
            navigator.getUserMedia (
                // constraints
                {
                    video: true,
                    audio: false
                },

                // successCallback
22.         function(localMediaStream) {
                    var video =document.querySelector('video');
                    video.src =window.URL.createObjectURL(localMediaStream);
                    webcamStream = localmediaStream;

                    },

                    // errorCallback
                    function(err) {
                        console.log("The following error occurred:
32.         " + err);
                    }
                );
            }
        }
    }
  </script>
</head>
</html>
```

```

    } else {
        console.log("getUserMedia not supported");
    }
}
function stopWebcam() {
    webcamStream.stop();
}
</script>
</head>
<body >
    <video width=200 height=200id="video" controls autoplay>
</video>
    <button onclick="startWebcam();">Start Webcam</button>
    <button onclick="stopWebcam();">Stop Webcam</button>
</body>
</html>

```

## Explanation:

**Lines 6-9:** set the `navigator.getUserMedia` method with the name that works on the current browser. As some browsers only have experimental implementations, this method needs to be prefixed by `-webkit` or `-moz` etc. Then in the rest of the code, we can just use `navigator.getUserMedia` without worrying about prefixes. If the browser does not support this API at all, it will receive `null`.

**Line 13:** we test if `navigator.getUserMedia` is not `null` (aka supported by the current browser).

**Lines 14-32:** we call `navigator.getUserMedia`. We defined here (line 21 and 28) the callback functions directly between the parenthesis of the method call. This is possible with JavaScript, and might confuse beginners:

- **Lines 15-19** define the first parameter of the call: a JavaScript object that defines the "constraints", here we only want to capture the video.
- **Lines 21-26** define the function called in case of success, instead of having

just the function name here like in previous examples, we directly put the code of the function. Lines 23-25 are executed when the webcam stream could be opened. Line 25 we store the stream in a global variable so that we can use from another function (for stopping/starting the webcam...)

- **Lines 28-31** define a function called in case of error (the webcam cannot be accessed).

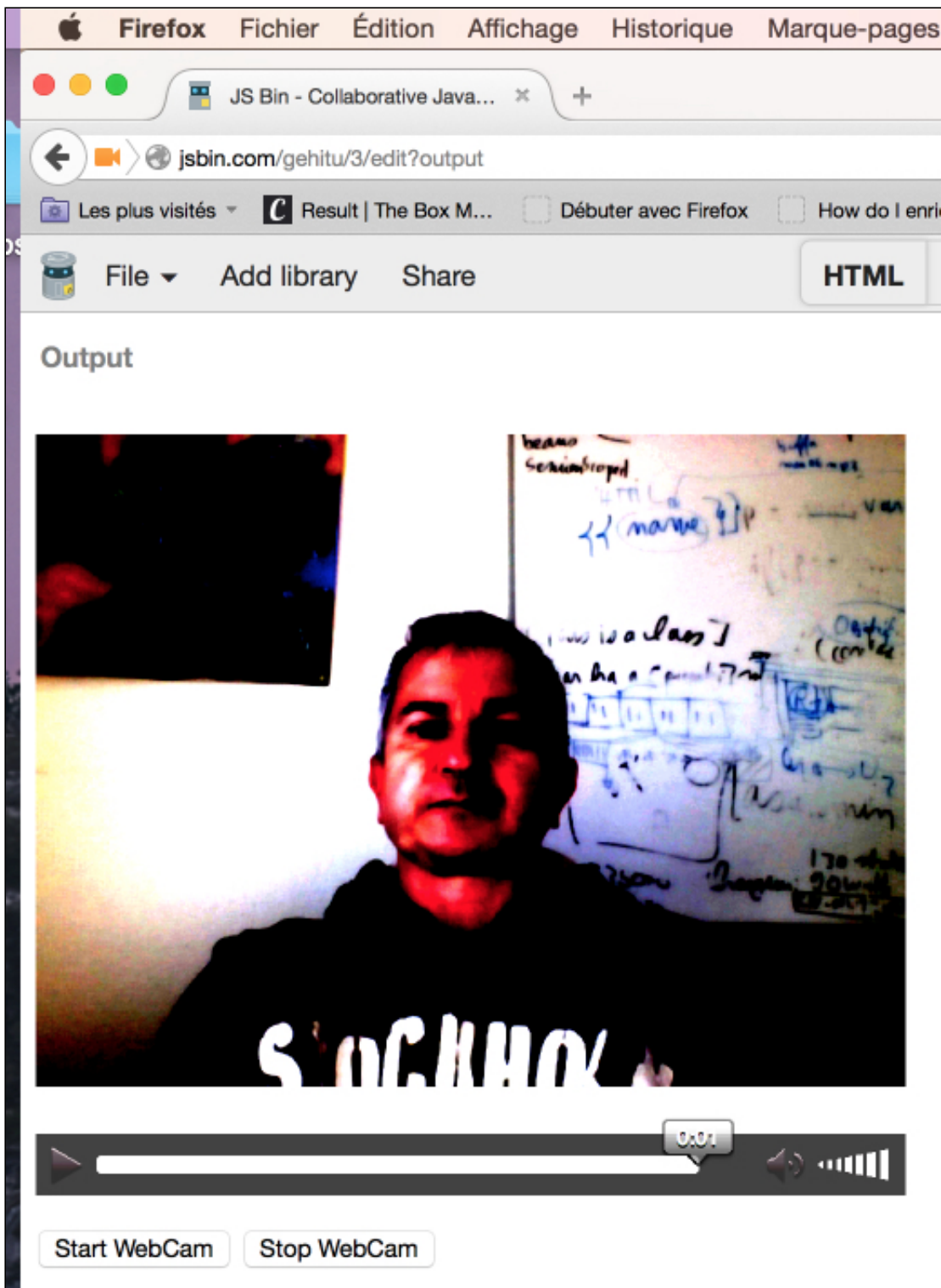
Line 32: closing parenthesis of `navigator.getUserMedia(...)` method call.

**Lines 37-39:** a function for stopping the webcam. We use the global variable `webcamStream` here, that has been initialized when we started using the webcam, line 25.

OTHER EXAMPLES THAT MIX IN WHAT WE'VE SEEN IN PREVIOUS CHAPTERS, BUT THIS TIME WITH A LIVE VIDEO STREAM

## Applying CSS effects on a video element with a live webcam

Try this example that shows how to use the `getUserMedia` API - note the CSS effects (click on the video to cycle from one effect to another). Works in Chrome/Firefox/Opera: [Online version at JS Bin](#)



## Taking a snapshot from the live webcam stream

The trick is to copy and paste the current image from the video stream into a `<canvas>` element:

[Online version at JS Bin](#)

We will examine this example in greater detail next week when we look at

the `<canvas>` element. For the time being, just play with the example.

Firefox
Fichier
Édition
Affichage
Historique
Marque-pages
Outils
Fenêtre
Aide

JS Bin
JS Bin - Collaborative Java...

jsbin.com/gehitu/6/edit?js,output
Rechercher

Les plus visités
Result | The Box M...
Débuter avec Firefox
How do I enrich m...
New Super Mario ...
Re: RESTful API fo...

File
Add library
Share
HTML
CSS
JavaScript
Console
Output
Account
Blog

```

    audio: false
  },

  // successCallback
  function(localMediaStream) {
    video = document.querySelector('video');
    video.src =
window.URL.createObjectURL(localMediaStream);
    webcamStream = localMediaStream;
  },

  // errorCallback
  function(err) {
    console.log("The following error occurred: " +
err);
  }
};
} else {
  console.log("getUserMedia not supported");
}
}

function stopWebcam() {
  webcamStream.stop();
}

//-----
// TAKE A SNAPSHOT CODE
//-----
var canvas, ctx;

function init() {
  // Get the canvas and obtain a context for
  // drawing in it
  canvas = document.getElementById("myCanvas");
  ctx = canvas.getContext('2d');
}

function snapshot() {
  // Draws current image from the video element into
  the canvas
  ctx.drawImage(video, 0,0, canvas.width,
canvas.height);
}

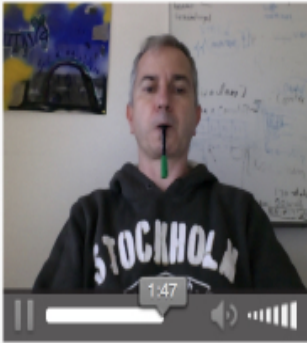
```

Output
Run with JS
Auto-run JS


## Take a snapshot of the current video stream

See CSS and JavaScript tabs. Click on the Start WebCam button.

Start WebCam
Stop WebCam
Take Snapshot



Screenshots :



Bin
just

## IMPRESSIVE DEMONSTRATIONS AVAILABLE ON THE WEB

- Exploding video: look at yourself and click on the video to make it explode! [Working demonstration](#) and [explanations here](#).
- [Live CSS filter Photo booth](#)
- A MUST TRY: [Paul Neave's WebGL Camera Effects](#)
- [Paint in real time with webcam output](#), choose webcam in the right menu, then click and drag rectangles. Share the URL and you have a multi-participant paint program with webcam painting! By Michel Buffa.