# Photo booth

In this project, you will create a simple photo booth application. You will use the local video camera on your computer and to take snapshots. You can then apply visual filters to your snapshots.

## Core concepts

* Capturing video from a local camera
* Visual filters

The ability to capture and manipulate video and audio from Javascript is part of a new standard called WebRTC. For more information, try some of these links:

* http://www.html5rocks.com/en/tutorials/getusermedia/intro/
* http://www.webrtc.org/

## Files

As usual, create a new folder for this project and add the following files: manifest.json, photobooth.html, photobooth.css, photobooth.js

*manifest.json*

{  
 "name": "Video Camera",  
 "description": "This app lets you record video.",  
 "version": "1.0",  
 "manifest\_version": 2,  
  
 "app": {  
 "launch": {  
 "local\_path": "photobooth.html"  
 }  
 }  
}

photobooth.html

<!DOCTYPE html>  
<html>  
<head>  
 <script src="photobooth.js" type="text/javascript"></script>  
 <link rel="stylesheet" type="text/css" href="photobooth.css">  
</head>  
<body>  
 <div id="effects">  
 <button data-effect="sepia(1)">Sepia</button>  
 <button data-effect="invert(1)">Invert</button>  
 <button data-effect="blur(3px)">Blur</button>  
 <button data-effect="grayscale(1)">Grayscale</button>  
 <button data-effect="">Normal</button>  
 </div>  
  
 <div>  
 <button id="photo-button">Snap photo!</button>  
 </div>  
  
 <video id="my-webcam"></video>  
  
 <div id="scroller">  
 <div id="my-photos">  
 </div>  
 </div>  
</body>  
</html>

photobooth.css

#my-photos {  
 overflow-x: auto;  
 white-space: nowrap;  
}  
  
#my-photos canvas {  
 display: inline-block;  
 height: 150px;  
 margin-right: 2px;  
}

*photobooth.js*

The parts of the code between /\* and \*/ are "comments". You do not need to type in the comments.

/\*  
 \* This will cause the browser to ask the user's permission to use the video  
 \* camera. If the user grant permission, then the `accessGranted` function  
 \* will be called. Otherwise, the `accessDenied` function will be called.  
 \*/  
navigator.webkitGetUserMedia({video: true}, accessGranted, accessDenied);  
  
function accessGranted(localMediaStream) {  
 /\*  
 \* Get a reference to the <video> element in the HTML file  
 \*/  
 var video = document.getElementById('my-webcam');  
  
 /\*  
 \* Hook the video element up to the stream from the video camera  
 \*/  
 video.src = window.URL.createObjectURL(localMediaStream);  
  
 video.play();  
}  
  
function accessDenied() {  
 console.log('access denied');  
}  
  
function snapPhoto() {  
 /\*  
 \* Create a new canvas element. A canvas element is a special kind of HTML  
 \* element that can be drawn on using Javascript.  
 \*/  
 var canvas = document.createElement('canvas');  
  
 /\*  
 \* Set the size of the canvas to the dimensions of the video  
 \*/  
 var video = document.getElementById('my-webcam');  
 canvas.width = video.clientWidth;  
 canvas.height = video.clientHeight;  
  
 /\*  
 \* You can draw shapes, images, and text onto a canvas element. In this case  
 \* we are going to grab an image from one frame from the camera element and  
 \* draw it onto the canvas  
 \*/  
 var context = canvas.getContext('2d');  
 context.drawImage(video, 0, 0);  
  
 /\*  
 \* If the user has applied a filter to their video, we will want to apply  
 \* that to the canvas as well.  
 \*/  
 canvas.style.webkitFilter = video.style.webkitFilter;  
  
 /\*  
 \* Now that the canvas is ready, add it to HTML document. It will be added  
 \* as a child of the 'my-photos' element.  
 \*/  
 var list = document.getElementById('my-photos');  
 list.appendChild(canvas);  
  
 /\*  
 \* Scroll the list all the way to the right, so that the user can see the  
 \* new canvas element \*/  
 list.scrollLeft = list.scrollWidth;  
}  
  
/\*  
 \* This tells the browser to wait until the DOMContentLoaded event occurs before  
 \* continuing. This is a common pattern found in Javascript applications.  
 \* Google for `DOMContentLoaded` for more information.  
 \*/  
document.addEventListener('DOMContentLoaded', function() {  
 /\*  
 \* This tells the browser that when the use clicks the 'photo-button', the  
 \* snapPhoto function should be called  
 \*/  
 var button = document.getElementById('photo-button');  
 button.onclick = snapPhoto;  
  
 /\*  
 \* This tells the browser that when any of the effect buttons are pushed,  
 \* we should read the filter name off of the 'data-effect' attribute and  
 \* add it to the video element.  
 \*/  
 var video = document.getElementById('my-webcam');  
 var effects = document.getElementById('effects');  
 effects.onclick = function(event) {  
 video.style.webkitFilter = event.target.getAttribute('data-effect');  
 };  
});

## Running the application

Once you have created your Chrome app, open a new tab tab. There should be an icon called "Photo booth".

## Improvement ideas

* Add a way to save your files to your local disk. Or tweet them!
* Mash this project up with the Chat Room project to let people upload snapshots to a chat room.
* WebRTC allows people to connect their video streams to other people over the internet. You can use this functionality to create a "Skype"-like program.