Capturing image from camera using getUserMedia() and saving it in a document using XPages

November 5, 2013

HTML5 introduces a new API *navigator.getUserMedia()* which allows web applications to access a user's camera and microphone. This article on HTML5 Rocks and this question on StackOverflow describe how to use it in your web applications.

In this article I will describe how to use *navigator.getUserMedia()* in your XPage and save the captured image in your Notes document. As of now this demo is supported only on Google Chrome & Mozilla Firefox on desktop and mobile (I have tested it on Andorid only).

HTML5 introduces <*video*> tag which allow developers access to user's camera.

Next we need to add <*canvas*> tag and an <*image*> tag. The image captured from user's camera is drawn on to <*canvas*>. We then get the Base64 encoded image from <*canvas*> put into our <*image*> (JavaScript code to do that comes later in article).

Notice that the <*canvas*> is hidden.

I am also going to add a text area to my XPage where I will store the Base64 encoded text of the captured image which later would be converted into image using Java (code for that comes later in article). While developing applications you can hide this text area using CSS.

And finally we need a button to take snapshot from camera.

In our JavaScript code we initialize all the variables we would be using.

```
var video = document.getElementById("video"); // Video
var canvas = document.getElementById("canvas"); // Canvas
var ctx = canvas.getContext("2d"); // Used to draw image from video stream on to canvas
var localMediaStream = null;
var capturedImage = document.getElementById("#{id:capturedImage}"); // Image
```

To get the video stream from camera we use *navigator.getUserMedia*. As of now *navigator.getUserMedia* is vendor prefixed so we test which one is supported.

```
// Get the video from the webcam, this is currently prefixed for different browsers, so we need to test which one is
navigator.getMedia = (navigator.getUserMedia || navigator.webkitGetUserMedia
                                        || navigator.mozGetUserMedia || navigator.msGetUserMedia);
```

Now we need to turn on the camera, get the video stream and show it in <*video*> tag. The code to get the video stream slightly differs from browser to browser.

```
navigator.getMedia({
        video : true,
        audio : false // We do NOT require audio
}, function(stream) {
```

```
        if (navigator.mozGetUserMedia) { // For Firefox
                video.mozSrcObject = stream;
        } else { // For Chrome
                var vendorURL = window.webkitURL || window.URL;
                video.src = vendorURL.createObjectURL(stream);
        }
        video.play(); // Start the video in webcam
        localMediaStream = stream; // Store the video stream
        button.textContent = "Say Cheese!";
}, function(e) {
        alert("An error occured! " + (e.name || e));
});
```

Once we have the video stream we can draw it on our <*canvas*>. While drawing we need to make sure that the height and width of our <*video*> and <*canvas*> are set to same dimensions otherwise the drawn image gets cropped. To get the Base64 encoded text for image we use *canvas.toDataURL*.

```
// Draw on canvas
canvas.width = video.videoWidth;
canvas.height = video.videoHeight;

// Draw on image
capturedImage.height = video.videoHeight;
capturedImage.width = video.videoWidth;
ctx.drawImage(video, 0, 0);// , width, height);
var imageData = canvas.toDataURL("image/png"); // Gets the Base64 encoded text for image
capturedImage.src = imageData;

// Store the Base64 of captured image
document.getElementById("#{id:taImageBase64}").value = imageData;
```

So our final code for button looks something like this:

```
var button = document.getElementById("#{id:btnCapture}");
button.addEventListener("click", function(e) {
        // Check if browser is supported
        if (!navigator.getMedia) {
                alert("Your browser does not support video stream.");
                return;
        }

        // If the video stream is working
        if (localMediaStream) {
                // Draw on canvas
                canvas.width = video.videoWidth;
                canvas.height = video.videoHeight;

                // Draw on image
                capturedImage.height = video.videoHeight;
                capturedImage.width = video.videoWidth;
                ctx.drawImage(video, 0, 0);
```

```
                var imageData = canvas.toDataURL("image/png"); // Gets the Base64 encoded text for image
                capturedImage.src = imageData;

                // Store the Base64 of captured image
                document.getElementById("#{id:taImageBase64}").value = imageData;
                return;
        }

        // If video stream is NOT working
        navigator.getMedia({
                video : true,
                audio : false // We do NOT require audio
        }, function(stream) {
                if (navigator.mozGetUserMedia) {
                        video.mozSrcObject = stream;
                } else {
                        var vendorURL = window.webkitURL || window.URL;
                        video.src = vendorURL.createObjectURL(stream);
                }
                video.play(); // Start the video in webcam
                localMediaStream = stream; // Store the video stream
                button.textContent = "Say Cheese!";
        }, function(e) {
                alert("An error occured! " + (e.name || e));
        });
}, false);
```

Now we need to convert the Base64 text to image and attach it to our document. For converting Base64 to image we create a Java class and use *javax.xml.bind.DatatypeConverter.parseBase64Binary* to convert it to binary and then we can attach it to a rich text field in document.

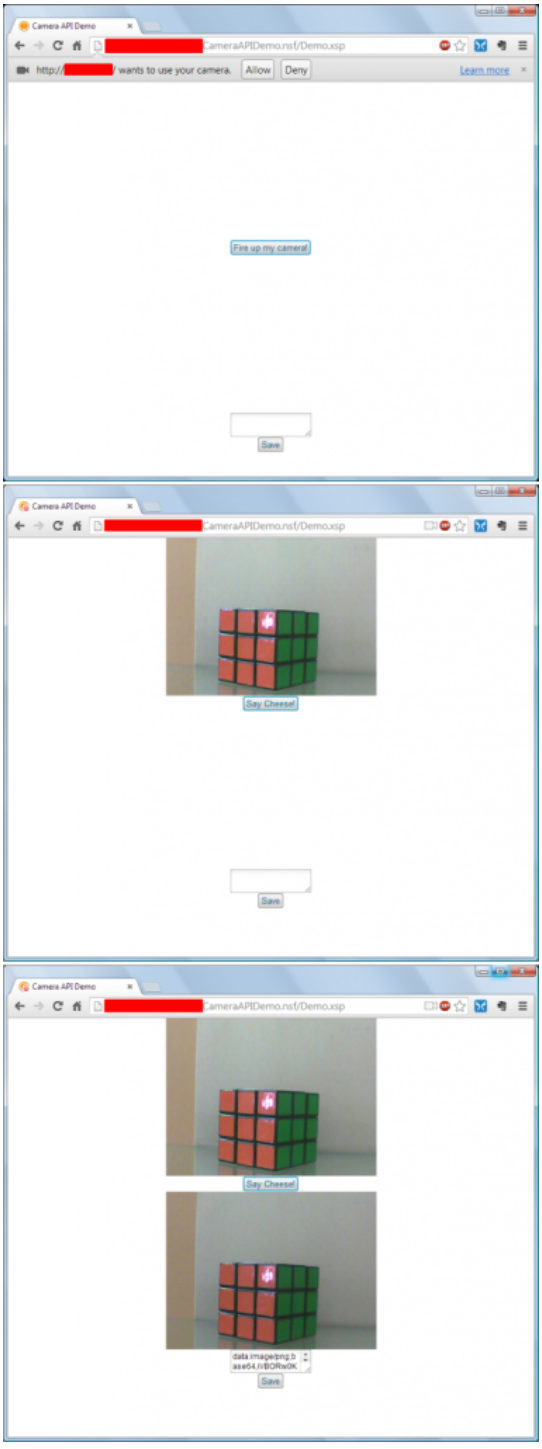
```
package uk.co.pipalia;

import java.io.ByteArrayInputStream;
import javax.faces.context.FacesContext;
import javax.faces.el.ValueBinding;
import javax.xml.bind.DatatypeConverter;
import lotus.domino.Document;
import lotus.domino.MIMEEntity;
import lotus.domino.MIMEHeader;
import lotus.domino.Session;
import lotus.domino.Stream;
import com.ibm.xsp.model.domino.DominoUtils;
import com.ibm.xsp.page.compiled.ExpressionEvaluatorImpl;

public class SaveBase64Image {
        Document targetDoc;
        String b64String;

        public SaveBase64Image(Document targetDoc) {
                // Get the value stored in text area
                String valueExpr = "#{javascript: getComponent(\"taImageBase64\").getValue();}";
                FacesContext facesContext = FacesContext.getCurrentInstance();
                ExpressionEvaluatorImpl evaluator = new ExpressionEvaluatorImpl(facesContext);
```

```
                ValueBinding vb = evaluator.createValueBinding(facesContext.getViewRoot(), valueExpr, null, null);
                b64String = (String) vb.getValue(facesContext);

                this.targetDoc = targetDoc;
                b64String = b64String.replace("", "");
        }

        public void process() {
                if (b64String.trim().equals("")) {
                        return;
                }

                try {
                        // Convert the Base64 string to binary
                        byte[] imgBytes = DatatypeConverter.parseBase64Binary(b64String);

                        Session s = DominoUtils.getCurrentSession();
                        s.setConvertMime(false);

                        // Create MIME to store the image
                        MIMEEntity richtext = targetDoc.createMIMEEntity("CapturedImage");
                        MIMEHeader header = richtext.createHeader("Content-Type");
                        header.setHeaderVal("multipart/mixed");

                        MIMEEntity child = richtext.createChildEntity();
                        header = child.createHeader("Content-Disposition");
                        header.setHeaderValAndParams("attachment; filename=camerasnap.png");

                        // Add the image to MIME via stream
                        Stream stream = s.createStream();
                        stream.setContents(new ByteArrayInputStream(imgBytes));
                        child.setContentFromBytes(stream, "image/png", MIMEEntity.ENC_IDENTITY_BINARY);

                        stream.close();
                        s.setConvertMime(true);
                } catch (Exception e) {
                        System.out.println("Error in uk.co.pipalia.SaveBase64Image.process()");
                        e.printStackTrace();
                }
        }
}
```

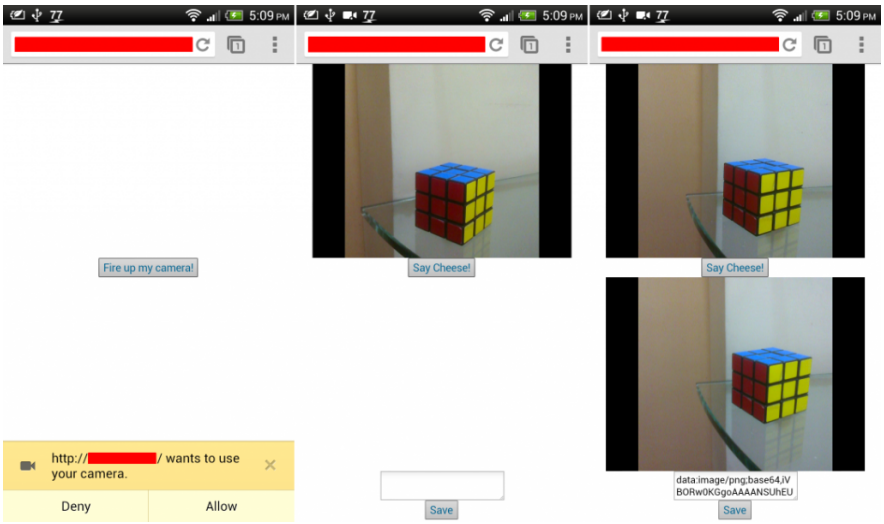
We call this Java class from *querySaveDocument* event.

And we ready to take snaps from our webcam.

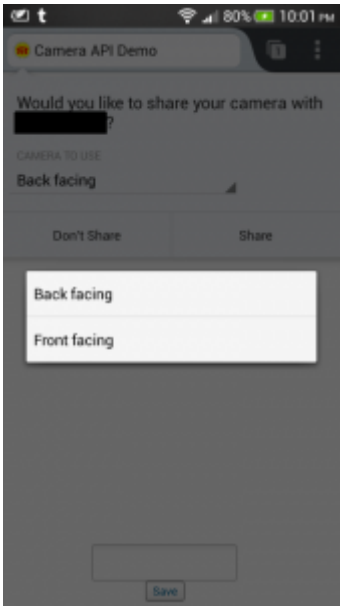
When you click on the button “Fire up my camera!” browser will ask for your permission. Once the permission is granted you can see your video and take snaps.



Camera API Demo on Google Chrome for PC



Camera API Demo on Google Chrome for Android



Camera API Demo on Firefox for Android

On Firefox for Android it even allows you to specify which camera do you want to use – back facing or front facing.

Download Demo Database

November 5, 2013 [http://www.pipalia.co.uk/capturing-image-camera-using-getusermedia-saving-document/] | by Naveen Maurya | in HTML5, java, JavaScript, xpages .

9 Comments

9 thoughts on “Capturing image from camera using getUserMedia() and saving it in a document using XPages”

Thorben Hellweg
November 6, 2013 at 8:44 pm

Great article, thanks for sharing this!

DavidLeedy
November 7, 2013 at 2:15 am

Wow! that’s an awesome blog post! A great topic and extremely well written. Thank you so much for sharing it!

Naveen Maurya Post author
November 7, 2013 at 4:38 am

Thanks David.

TomDixon
November 12, 2013 at 12:44 pm

TY for sharing this, someone had shared this in the domino group on LinkedIn and as a 1st time visitor to your website I am very impressed with your blog.. As a domino developer I will have to subscribe… cheers..

Samir Pipalia
November 13, 2013 at 10:57 am

Thanks Tom, appreciate your feedback.

javier
October 29, 2015 at 3:40 am

how can i install it Camera API Demo on Firefox for Android?

Naveen Maurya Post author
October 30, 2015 at 3:27 am

You don't need to install camera API on Firefox for Android. It's already enabled by default. Download the demo database and run it from a Domino server and you should be good to go.

marcel Bussien
July 8, 2016 at 7:07 am

Great Post
really usefull stuff

Marcel Bussien
July 13, 2016 at 1:29 pm

I mentioned you, hope you'll do so
https://github.com/StayAt/imagescaledown02
@BootstrapXpages
https://www.linkedin.com/in/marcel-bussien-1b345b34
best regards
Marcel