www.bytescout.com

How to read from live camera with barcode reader sdk in ASP.NET C# using ByteScout Data Extraction Suite

Learn to read from live camera with barcode reader sdk in ASP.NET C#

The sample source code below will teach you how to read from live camera with barcode reader sdk in ASP.NET C#. What is ByteScout Data Extraction Suite? It is the set that includes 3 SDK products for data extraction from PDF, scans, images and from spreadsheets: PDF Extractor SDK, Data Extraction SDK, Barcode Reader SDK. It can help you to read from live camera with barcode reader sdk in your ASP.NET C# application.

The SDK samples given below describe how to quickly make your application do read from live camera with barcode reader sdk in ASP.NET C# with the help of ByteScout Data Extraction Suite. Just copy and paste the code into your ASP.NET C# application's code and follow the instructions. If you want to use these ASP.NET C# sample examples in one or many applications then they can be used easily.

Our website gives trial version of ByteScout Data Extraction Suite for free. It also includes documentation and source code samples.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout Data Extraction Suite

Explore API Documentation

Get Free Training for ByteScout Data Extraction Suite

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

!!!ReadmeFIRST!!!.txt

```
Readme for live web cam barcode scanning ASP.NET demo

Supports all browsers by using 2 modes:

(required mode is automatically detected by the javascript)
- HTML5 based web camera capture (supported by Google Chrome, Firefox, Safari, Safari
- Flash based web camera capture (supported by Internet Explorer 9+ and any browser wi

IMPORTANT about HTML5 webcamera support (Chrome, Firefox, Safari on desktop and iOS):

Visual Studio 2008/2010 or higher is required for this sample to run!

1) Install evaluation copy of BarCode Reader SDK from www.bytescout.com
2) in Visual Studio use File - Open - Web-Site.. and open the folder with this sample
3) Add a reference to ByteScout BarCode Reader dll using the menu: Website - Add Refere
4) Right-click on "Default.aspx" file and select "Set As Start Page"
5) Run the project in debug mode
6) Visual Studio will run your project in Internet Explorer or another browser
If it run Internet Explorer then better copy the link from the IE and copy-and-paste to
7) When browser asks for permission to access web-cam click "Allow" to allow access to
8) click "START BARCODE SCAN.." button so the page will start automatically to grab fr
9) IMPORTANT: barcode should be in focus. Some webcams are not focusing on small barco

10) Click Stop to stop barcode scanning
Select barcode type to scan if need to scan a particular barcode type (by default scans


TESTING:
- we recommend to use the Conveyor plugin to test your web app from local net or public
```

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_De

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1. Transitional//EN" "http://www.w3.org/TR/xh

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title></title>
    <script src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.js" type="text/jav
    <script type="text/javascript" src="webcam.js"></script>
      <!-- this javascript plugin uses the webcam.swf file to capture image and send t
    <script type="text/javascript">
        var canvas, context, timer;
        var constraints = window.constraints = {
            audio: false,
```

```
        video: { facingMode: "environment" } // change to "user" for front camera,
    };
    //  (HTML5 based camera only) this portion of code will be used when browser su
    window.addEventListener("DOMContentLoaded", function () {
        canvas = document.getElementById("canvasU"),
        context = canvas.getContext("2d"),
        video = document.getElementById("video"),
        videoObj = { "video": true },
        errBack = function (error) {
            console.log("Video capture error: ", error.code);
        };

        // check if we can use HTML5 based camera (through mediaDevices.getUserMed
        if (navigator.mediaDevices.getUserMedia) { // Standard browser
            // display HTML5 camera
            document.getElementById("userMedia").style.display = '';
            // adding click event to take photo from webcam
            document.getElementById("snap").addEventListener("click", function () {
                context.drawImage(video, 0, 0, 640, 480);
            });

            navigator.mediaDevices
                .getUserMedia(constraints)
                .then(handleSuccess)
                .catch(handleError);
        }
        // check if we can use HTML5 based camera (through .getUserMedia() functio
        else if (navigator.webkitGetUserMedia) { // WebKit-prefixed for Google Chr
            // display HTML5 camera
            document.getElementById("userMedia").style.display = '';
            // adding click event to take photo from webcam
            document.getElementById("snap").addEventListener("click", function () {
                context.drawImage(video, 0, 0, 640, 480);
            });
            navigator.webkitGetUserMedia(videoObj, function (stream) {
                video.src = window.webkitURL.createObjectURL(stream);
                video.play();
            }, errBack);
        }
        // check if we can use HTML5 based camera (through .getUserMedia() functio
        else if (navigator.mozGetUserMedia) { // moz-prefixed for Firefox
            // display HTML5 camera
            document.getElementById("userMedia").style.display = '';
            // adding click event to take photo from webcam
            document.getElementById("snap").addEventListener("click", function () {
                context.drawImage(video, 0, 0, 640, 480);
            });
            navigator.mozGetUserMedia(videoObj, function (stream) {
                video.mozSrcObject = window.webkitURL.createObjectURL(stream);
                video.play();
            }, errBack);
        }
        else
        // if we can not use any of HTML5 based camera ways then we use Flash base
        {
            // display Flash camera
            document.getElementById("flashDiv").style.display = '';
            document.getElementById("flashOut").innerHTML = (webcam.get_html(640, 4
        }
```

```
        }, false);

        // (all type of camera) set the default selection of barcode type
        var selection = "All barcodes (slow)";

        // (all type of camera) gets the selection text of "barcode type to scan" combo
        function selectType(type) {
            selection = type.options[type.selectedIndex].text;
        }

        function handleSuccess(stream) {
            var video = document.querySelector('video');
            var videoTracks = stream.getVideoTracks();
            console.log('Got stream with constraints:', constraints);
            console.log(`Using video device: ${videoTracks[0].label}`);
            window.stream = stream; // make variable available to browser console
            video.srcObject = stream;
        }

        function handleError(error) {
            if (error.name === 'ConstraintNotSatisfiedError') {
                var v = constraints.video;
                errorMsg(`The resolution ${v.width.exact}x${v.height.exact} px is not s
            } else if (error.name === 'PermissionDeniedError') {
                errorMsg('Permissions have not been granted to use your camera and ' +
                    'microphone, you need to allow the page access to your devices in
                    'order for the demo to work.');
            }
            errorMsg(`getUserMedia error: ${error.name}`, error);
        }

        function errorMsg(msg, error) {
            var errorElement = document.querySelector('#errorMsg');
            errorElement.innerHTML += `<p>${msg}</p>`;
            if (typeof error !== 'undefined') {
                console.error(error);
            }
        }

        // (HTML5 based camera only)
        // uploads the image to the server
        function UploadToCloud() {
            document.getElementById('report').innerHTML = "scanning the current frame.
            context.drawImage(video, 0, 0, 640, 480);
            $("#upload").attr('disabled', 'disabled');
            $("#upload").attr("value", "Uploading...");
            var img = canvas.toDataURL('image/jpeg', 0.9).split(',')[1];
            // send AJAX request to the server with image data
            $.ajax({
                url: "HTML5Camera.aspx/Upload",
                type: "POST",
                data: "{ 'image': '" + img + "' , 'type': '" + selection + "'}",
                contentType: "application/json; charset=utf-8",
                dataType: "json",
                // on success output the result returned by the server side (See HTML5C
                success: function (data, status) {
                    $("#upload").removeAttr('disabled');
                    $("#upload").attr("value", "Upload");
                    if (data.d.length != 0) {
                        var htmlSelect = document.getElementById('OutListBoxHTML5');
```

```
                    //var selectBoxOption = document.createElement("option");
                    //selectBoxOption.text = data.d;
                    //selectBoxOption.id = "child";
                    //htmlSelect.insertBefore(selectBoxOption, htmlSelect.childNode
                    htmlSelect.value = data.d + "\r\n" + htmlSelect.value;
                }
            },
            // on error just show the message that no barcodes were found
            error: function (data) {
                document.getElementById('report').innerHTML = "no barcode found, s
                $("#upload").removeAttr('disabled');
                $("#upload").attr("value", "Upload");
            },
            async: false
        });
        timer = setTimeout(UploadToCloud, 3000);  // will capture new image to dete
    }

    // (flash based camera only) stop the capturing
    function stopCall() {
        document.getElementById('report').innerHTML = "STOPPED Scanning."
        clearTimeout(timer);
    }

    // (flash based camera only) sets the handler for callback completition to outp
    Webcam.on('onComplete', 'my_completion_handler');

    // (flash based camera only) this function will start flash to capture image ar
    function take_snapshot() {
        // set api to be called by flash camera
        webcam.set_api_url('FlashCamera.aspx?type=' + selection);
        webcam.set_quality(100);
        // enable the shutter sound
        webcam.set_shutter_sound(true);
        document.getElementById('upload_results').innerHTML = '<h4>Scanning...</h4>
        // capture image from the webcam
        webcam.snap();
        // set timout to capter new image (interval between captures)
        timer = setTimeout(take_snapshot, 3000);
    }

    // (flash based camera only) this one will work after recieving barcode from se
    // this function writes the output result back to the front page (from server s
    function my_completion_handler(msg) {
        var str = msg;
        // encode into html compatible string
        var result = str.match(/<d>(.*?)<\/d>/g).map(function (val) {
            return val.replace(/<\/?d>/g, '');
        });
        // add new result into the listbox
        var htmlSelect = document.getElementById('OutListBoxFlash');
        //var selectBoxOption = document.createElement("option");
        //selectBoxOption.text = result;
        //selectBoxOption.id = "child";
        //htmlSelect.insertBefore(selectBoxOption, htmlSelect.childNodes[0]);
        htmlSelect.value = result + "\r\n" + htmlSelect.value;

        // reset webcam and flash to capture new image. this reset process flickers
        webcam.reset();
```

```
        }

        // (flash based camera only) stop the scan and set the message on the page
        function stopScanning() {
            document.getElementById('upload_results').innerHTML = "STOPPED Scanning."
            clearTimeout(timer);
        }
    </script>
    <style>
    span
    {
        font-size:20px;
    }
    </style>
</head>
<body>
    <form id="form1" runat="server">
    <!-- HTML5 camera based capturing section -->
    <!-- this div block will be shown when browser supports get user media-->
        <div id="userMedia"  style="display:none; height: 575px; width: 1182px">
        <table>
        <tr align="left" valign="top">
        <td valign="top" colspan="2">
        <h3 style=" color:Green; ">(HTML5 based camera) This works in Chrome, Firefox,
        <strong>IMPORTANT: html5 webcam access (Chrome, Firefox, Safari on desktop and
        </td>
        </tr>
          <tr align="left" valign="top">
                <td valign="top">
                    <h2 id="choiceU"> <b>Barcode Type To Scan (to start barcode scan -
                    <br />
          <select id="comboBoxBarCodeTypeHTML5" onchange="selectType(this)">
            <option value="1">All Barcodes (slow)</option>
            <option value="2">All Linear Barcodes (Code39, Code 128, EAN 13, UPCA, UPCE
            <option value="3">All 2D Barcodes (QR Code, Aztec, DataMatrix, PDF417, etc
            <option value="4">Code 39c</option>
            <option value="5">Code 128</option>
            <option value="6">EAN 13</option>
            <option value="7">UPCA</option>
            <option value="8">GS1-128</option>
            <option value="9">GS1DataBarExpanded</option>
            <option value="10">GS1DataBarExpandedStacked </option>
            <option value="11">GS1DataBarLimited</option>
            <option value="12">GS1DataBarOmnidirectional</option>
            <option value="13">GS1DataBarStacked</option>
            <option value="14">I2of5</option>
            <option value="15">Intelligent Mail</option>
            <option value="16">Patch Code</option>
            <option value="17">PDF 417</option>
            <option value="18">QR Code</option>
            <option value="19">DataMatrix</option>
            <option value="20">Aztec</option>
            <option value="21">MaxiCode</option>
         </select>
        <br />
        <span style=" font-size:20px; ">Output barcode values read appears below: </spa
        <br />
            <!-- decoding results appears in this listbox -->
            <textarea style="width:500px; height:100px;" size="8" id="OutListBoxHTML5">
            </textarea>
```

```
        <br />
            <input id="snap"  style="color:black; font-weight:bold; font-size:x-large;"
            <input id="pause" style="color:black;" type="button" onclick="stopCall();"
         </td>
         <td valign="top">
            <span>Webcam preview shows below:</span>
            <video id="video" width="640" height="480" autoplay playsinline muted></vi
         <!-- canvas with the output -->
       <canvas id="canvasU" width="640" height="480" style="display:none" ></canvas>

         </td>
         </tr>
         </table>
    </div>

    <!-- Flash (SWF) camera based capturing section -->
    <!-- this div block will be shown when browser does not support user media so the
         <div id="flashDiv" style=" display:none; ">
            <table>
            <tr align="left" valign="top">
            <td colspan="2"><h3 style=" color:Green; " >(FLASH based camera) This works
            </tr>
            <tr align="left" valign="top">
                <td valign="top">
                    <h4 id="choice"> <b>Barcode Type To Scan (to start barcode scan -
                    <br />
                     <select id="comboBoxBarCodeTypeFlash" onchange="selectType(this)">
            <option value="1">All Barcodes (slow)</option>
            <option value="2">All Linear Barcodes (Code39, Code 128, EAN 13, UPCA, UPC
            <option value="3">All 2D Barcodes (QR Code, Aztec, DataMatrix, PDF417, etc
            <option value="4">Code 39</option>
            <option value="5">Code 128</option>
            <option value="6">EAN 13</option>
            <option value="7">UPCA</option>
            <option value="8">GS1-128</option>
            <option value="9">GS1DataBarExpanded</option>
            <option value="10">GS1DataBarExpandedStacked </option>
            <option value="11">GS1DataBarLimited</option>
            <option value="12">GS1DataBarOmnidirectional</option>
            <option value="13">GS1DataBarStacked</option>
            <option value="14">I2of5</option>
            <option value="15">Intelligent Mail</option>
            <option value="16">Patch Code</option>
            <option value="17">PDF 417</option>
            <option value="18">QR Code</option>
            <option value="19">DataMatrix</option>
            <option value="20">Aztec</option>
            <option value="21">MaxiCode</option>
    </select>
            <br />
            <span style=" font-size:20px; ">Output barcode values read appears below:
        <br />

            <!-- decoding results appears in this listbox -->
          <textarea style="width:500px; height:100px;" size="8" id="OutListBoxFlash">
     <br />
                <input type="button"  style="color:black; font-weight:bold; font-size:x
        <br />
        <input type="button" style="color:black; font-weight:bold; font-size:x-large;"
        <br />
```

```
    </div>
    <div id="upload_results" style="background-color:#eee;"></div>
    </td>
    <td>
    <div id="flashOut"> </div>
    </td>
   </tr>
    </table>

    </form>

</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        /*
        IF YOU SEE TEMPORARY FOLDER ACCESS ERRORS:

        Temporary folder access is required for web application when you use ByteScout
        If you are getting errors related to the access to temporary folder like "Acces

        SOLUTION:

        If your IIS Application Pool has "Load User Profile" option enabled the IIS pr

        If you are running Web Application under an impersonated account or IIS_IUSRS g

        In this case
        - check the User or User Group your web application is running under
        - then add permissions for this User or User Group to read and write into that
        - restart your web application and try again

        */

    }
}
```

FlashCamera.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="FlashCamera.aspx.cs" Inherits="

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xh

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

FlashCamera.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.IO; //For MemoryStream
using System.Web.Services; //For WebMethod attribute
using Bytescout.BarCodeReader;
using System.Web.UI.HtmlControls;
using System.Drawing;
using System.Drawing.Imaging;
using System.Threading;

public partial class uploadimage : System.Web.UI.Page
{
    /*
    IF YOU SEE TEMPORARY FOLDER ACCESS ERRORS:

    Temporary folder access is required for web application when you use ByteScout SDK
    If you are getting errors related to the access to temporary folder like "Access to

    SOLUTION:
```

```
    If your IIS Application Pool has "Load User Profile" option enabled the IIS provide

    If you are running Web Application under an impersonated account or IIS_IUSRS group

    In this case
    - check the User or User Group your web application is running under
    - then add permissions for this User or User Group to read and write into that temp
    - restart your web application and try again

    */

        protected void Page_Load(object sender, EventArgs e)
    {

        try
        {
            String send = "";
            System.Drawing.Image originalimg;
            // read barcode type set
            String type = Request.QueryString["type"].ToString();
            MemoryStream log = new MemoryStream();
            byte[] buffer = new byte[1024];
            int c;
            // read input buffer with image and saving into the "log" memory stream
            while ((c = Request.InputStream.Read(buffer, 0, buffer.Length)) > 0)
            {
                log.Write(buffer, 0, c);
            }
            // create image object
            originalimg = System.Drawing.Image.FromStream(log);
            // resample image
            originalimg = originalimg.GetThumbnailImage(640, 480, new System.Drawing.Ir
            Bitmap bp = new Bitmap(originalimg);
            // create bytescout barcode reader object
            Reader reader = new Reader();
            // set barcode type to read
            reader.BarcodeTypesToFind = GetBarcodeTypeToFindFromCombobox(type);
            // read barcodes from image
            reader.ReadFrom(bp);
            // if there are any result then convert them into a single stream
            if (reader.FoundBarcodes != null)
            {
                foreach (FoundBarcode barcode in reader.FoundBarcodes)
                {
                    // form the output string
                    send = send + (String.Format("{0} : {1}", barcode.Type, barcode.Va
                }
            }
            // close the memory stream
            log.Close();
            // dispose the image object
            originalimg.Dispose();
            // write output
            Response.Write("<d>" + send + "</d>");
        }
        catch (Exception ex)
        {
            // write the exception if any
            Response.Write("<d>" + ex + "</d>");
        }
```

```
    }
    public bool ThumbnailCallback() { return false; }

    /// <summary>
    /// Get symbology filter for the SDK from the combobox selection text
    /// </summary>
    /// <param name="barType"></param>
    /// <returns></returns>
    private static BarcodeTypeSelector GetBarcodeTypeToFindFromCombobox(string barType)
    {
        string selectedItemText = barType.Trim().ToUpper();
        BarcodeTypeSelector barcodeTypeToScan = new BarcodeTypeSelector();

        if (selectedItemText.IndexOf("ALL BARCODES") > -1)
            barcodeTypeToScan.SetAll();
        else if (selectedItemText.IndexOf("ALL LINEAR BARCODES") > -1)
            barcodeTypeToScan.SetAll1D();
        else if (selectedItemText.IndexOf("ALL 2D BARCODES") > -1)
            barcodeTypeToScan.SetAll2D();
        else if (selectedItemText.IndexOf("AZTEC") > -1)
            barcodeTypeToScan.Aztec = true;
        else if (selectedItemText.IndexOf("CODABAR") > -1)
            barcodeTypeToScan.Codabar = true;
        else if (selectedItemText.IndexOf("CODE 39") > -1)
            barcodeTypeToScan.Code39 = true;
        else if (selectedItemText.IndexOf("CODE 128") > -1)
            barcodeTypeToScan.Code128 = true;
        else if (selectedItemText.IndexOf("DATAMATRIX") > -1)
            barcodeTypeToScan.DataMatrix = true;
        else if (selectedItemText.IndexOf("EAN 13") > -1)
            barcodeTypeToScan.EAN13 = true;
        else if (selectedItemText.IndexOf("GS1-128") > -1)
            barcodeTypeToScan.GS1 = true;
        else if (selectedItemText.IndexOf("GS1DATABAREXPANDED") > -1)
            barcodeTypeToScan.GS1DataBarExpanded = true;
        else if (selectedItemText.IndexOf("GS1DATABAREXPANDEDSTACKED") > -1)
            barcodeTypeToScan.GS1DataBarExpandedStacked = true;
        else if (selectedItemText.IndexOf("GS1DATABARLIMITED") > -1)
            barcodeTypeToScan.GS1DataBarLimited = true;
        else if (selectedItemText.IndexOf("GS1DATABAROMNIDIRECTIONAL") > -1)
            barcodeTypeToScan.GS1DataBarOmnidirectional = true;
        else if (selectedItemText.IndexOf("GS1DATABARSTACKED") > -1)
            barcodeTypeToScan.GS1DataBarStacked = true;
        else if (selectedItemText.IndexOf("I2OF5") > -1)
            barcodeTypeToScan.Interleaved2of5 = true;
        else if (selectedItemText.IndexOf("INTELLIGENT MAIL") > -1)
            barcodeTypeToScan.IntelligentMail = true;
        else if (selectedItemText.IndexOf("PATCH") > -1)
            barcodeTypeToScan.PatchCode = true;
        else if (selectedItemText.IndexOf("PDF 417") > -1)
            barcodeTypeToScan.PDF417 = true;
        else if (selectedItemText.IndexOf("QR CODE") > -1)
            barcodeTypeToScan.QRCode = true;
        else if (selectedItemText.IndexOf("UPCA") > -1)
            barcodeTypeToScan.UPCA = true;
        else if (selectedItemText.IndexOf("UPCE") > -1)
            barcodeTypeToScan.UPCE = true;
        else if (selectedItemText.IndexOf("MAXICODE") > -1)
            barcodeTypeToScan.MaxiCode = true;
```

```
        return barcodeTypeToScan;
    }
}
```

HTML5Camera.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="HTML5Camera.aspx.cs" Inherits="

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xh

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

HTML5Camera.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Web.UI;
using System.IO; //For MemoryStream
using System.Web.Services; //For WebMethod attribute
using Bytescout.BarCodeReader;
using System.Web.UI.HtmlControls;
using System.Drawing;
using System.Drawing.Imaging;
using System.Threading;
using System.Text;


public partial class Camera : System.Web.UI.Page
```

```
{
    /*
    IF YOU SEE TEMPORARY FOLDER ACCESS ERRORS:

    Temporary folder access is required for web application when you use ByteScout SDK
    If you are getting errors related to the access to temporary folder like "Access t

    SOLUTION:

    If your IIS Application Pool has "Load User Profile" option enabled the IIS provid

    If you are running Web Application under an impersonated account or IIS_IUSRS grou

    In this case
    - check the User or User Group your web application is running under
    - then add permissions for this User or User Group to read and write into that tem
    - restart your web application and try again

    */

    protected void Page_Load(object sender, EventArgs e)
    {
    }

    /// <summary>
    /// Takes the base64 encoded image from the browser
    /// and tries to read barodes from it
    /// </summary>
    /// <param name="image"></param>
    /// <param name="type"></param>
    /// <returns></returns>
    [WebMethod]
    public static string Upload(string image, string type)
    {
        try
        {
            StringBuilder send = new StringBuilder();

            // lock by send variable
            lock (send)
            {
                // Convert base64 string from the client side into byte array
                byte[] bitmapArrayOfBytes = Convert.FromBase64String(image);
                // Create Bytescout.BarCodeReader.Reader object
                Reader reader = new Reader();
                // Get the barcode type from user's selection in the combobox
                reader.BarcodeTypesToFind = GetBarcodeTypeToFindFromCombobox(type);
                // Read barcodes from image bytes
                reader.ReadFromMemory(bitmapArrayOfBytes);
                // Check whether the barcode is decoded
                if (reader.FoundBarcodes != null)
                {
                    // Add each decoded barcode into the string
                    foreach (FoundBarcode barcode in reader.FoundBarcodes)
                    {
                        // Add barcodes as text into the output string
                        send.AppendLine(String.Format("{0} : {1}", barcode.Type, barco
                    }
                }
```

```
                // Return the output string as JSON
                            return send.ToString();
            }
        }
        catch (Exception ex)
        {
            // return the exception instead
            return (ex.Message + "\r\n" + ex.StackTrace);
        }
    }

    /// <summary>
    /// Get symbology filter for the SDK from the combobox selection text
    /// </summary>
    /// <param name="barType"></param>
    /// <returns></returns>
    private static BarcodeTypeSelector GetBarcodeTypeToFindFromCombobox(string barType)
    {
        string selectedItemText = barType.Trim().ToUpper();
        BarcodeTypeSelector barcodeTypeToScan = new BarcodeTypeSelector();

        if (selectedItemText.IndexOf("ALL BARCODES") > -1)
            barcodeTypeToScan.SetAll();
        else if (selectedItemText.IndexOf("ALL LINEAR BARCODES") > -1)
            barcodeTypeToScan.SetAll1D();
        else if (selectedItemText.IndexOf("ALL 2D BARCODES") > -1)
            barcodeTypeToScan.SetAll2D();
        else if (selectedItemText.IndexOf("AZTEC") > -1)
            barcodeTypeToScan.Aztec = true;
        else if (selectedItemText.IndexOf("CODABAR") > -1)
            barcodeTypeToScan.Codabar = true;
        else if (selectedItemText.IndexOf("CODE 39") > -1)
            barcodeTypeToScan.Code39 = true;
        else if (selectedItemText.IndexOf("CODE 128") > -1)
            barcodeTypeToScan.Code128 = true;
        else if (selectedItemText.IndexOf("DATAMATRIX") > -1)
            barcodeTypeToScan.DataMatrix = true;
        else if (selectedItemText.IndexOf("EAN 13") > -1)
            barcodeTypeToScan.EAN13 = true;
        else if (selectedItemText.IndexOf("GS1-128") > -1)
            barcodeTypeToScan.GS1 = true;
        else if (selectedItemText.IndexOf("GS1DATABAREXPANDED") > -1)
            barcodeTypeToScan.GS1DataBarExpanded = true;
        else if (selectedItemText.IndexOf("GS1DATABAREXPANDEDSTACKED") > -1)
            barcodeTypeToScan.GS1DataBarExpandedStacked = true;
        else if (selectedItemText.IndexOf("GS1DATABARLIMITED") > -1)
            barcodeTypeToScan.GS1DataBarLimited = true;
        else if (selectedItemText.IndexOf("GS1DATABAROMNIDIRECTIONAL") > -1)
            barcodeTypeToScan.GS1DataBarOmnidirectional = true;
        else if (selectedItemText.IndexOf("GS1DATABARSTACKED") > -1)
            barcodeTypeToScan.GS1DataBarStacked = true;
        else if (selectedItemText.IndexOf("I2OF5") > -1)
            barcodeTypeToScan.Interleaved2of5 = true;
        else if (selectedItemText.IndexOf("INTELLIGENT MAIL") > -1)
            barcodeTypeToScan.IntelligentMail = true;
        else if (selectedItemText.IndexOf("PATCH") > -1)
            barcodeTypeToScan.PatchCode = true;
        else if (selectedItemText.IndexOf("PDF 417") > -1)
            barcodeTypeToScan.PDF417 = true;
        else if (selectedItemText.IndexOf("QR CODE") > -1)
```

```
            barcodeTypeToScan.QRCode = true;
        else if (selectedItemText.IndexOf("UPCA") > -1)
            barcodeTypeToScan.UPCA = true;
        else if (selectedItemText.IndexOf("UPCE") > -1)
            barcodeTypeToScan.UPCE = true;
        else if (selectedItemText.IndexOf("MAXICODE") > -1)
            barcodeTypeToScan.MaxiCode = true;

        return barcodeTypeToScan;
    }


}
```

Web.config

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
        <system.web>
                <compilation debug="true" targetFramework="4.0"/>
                <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"
</configuration>
```

webcam.js

```
// WebcamJS v1.0.25
// Webcam library for capturing JPEG/PNG images in JavaScript
// Attempts getUserMedia, falls back to Flash
// Author: Joseph Huckaby: http://github.com/jhuckaby
// Based on JPEGCam: http://code.google.com/p/jpegcam/
// Copyright (c) 2012 - 2017 Joseph Huckaby
// Licensed under the MIT License

(function (window) {
    var _userMedia;
```

```
    // declare error types

    // inheritance pattern here:
    // https://stackoverflow.com/questions/783818/how-do-i-create-a-custom-error-in-jav
    function FlashError() {
        var temp = Error.apply(this, arguments);
        temp.name = this.name = "FlashError";
        this.stack = temp.stack;
        this.message = temp.message;
    }

    function WebcamError() {
        var temp = Error.apply(this, arguments);
        temp.name = this.name = "WebcamError";
        this.stack = temp.stack;
        this.message = temp.message;
    }

    var IntermediateInheritor = function () { };
    IntermediateInheritor.prototype = Error.prototype;

    FlashError.prototype = new IntermediateInheritor();
    WebcamError.prototype = new IntermediateInheritor();

    var Webcam = {
        version: '1.0.25',

        // globals
        protocol: location.protocol.match(/https/i) ? 'https' : 'http',
        loaded: false,   // true when webcam movie finishes loading
        live: false,     // true when webcam is initialized and ready to snap
        userMedia: true, // true when getUserMedia is supported natively

        iOS: /iPad|iPhone|iPod/.test(navigator.userAgent) && !window.MSStream,

        params: {
            width: 0,
            height: 0,
            dest_width: 0,         // size of captured image
            dest_height: 0,        // these default to width/height
            image_format: 'jpeg',  // image format (may be jpeg or png)
            jpeg_quality: 90,      // jpeg image quality from 0 (worst) to 100 (best)
            enable_flash: true,    // enable flash fallback,
            force_flash: false,    // force flash mode,
            flip_horiz: false,     // flip image horiz (mirror mode)
            fps: 30,               // camera frames per second
            upload_name: 'webcam', // name of file in upload post data
            constraints: null,     // custom user media constraints,
            swfURL: '',            // URI to webcam.swf movie (defaults to the js locat
            flashNotDetectedText: 'ERROR: No Adobe Flash Player detected.  Webcam.js re
            noInterfaceFoundText: 'No supported webcam interface found.',
            unfreeze_snap: true,    // Whether to unfreeze the camera after snap (defau
            iosPlaceholderText: 'Click here to open camera.',
            user_callback: null,    // callback function for snapshot (used if no user_
            user_canvas: null       // user provided canvas for snapshot (used if no us
        },

        errors: {
            FlashError: FlashError,
            WebcamError: WebcamError
```

```
        },

        hooks: {}, // callback hook functions

        init: function () {
            // initialize, check for getUserMedia support
            var self = this;

            // Setup getUserMedia, with polyfill for older browsers
            // Adapted from: https://developer.mozilla.org/en-US/docs/Web/API/MediaDev
            this.mediaDevices = (navigator.mediaDevices && navigator.mediaDevices.getU
                navigator.mediaDevices : ((navigator.mozGetUserMedia || navigator.webk
                    getUserMedia: function (c) {
                        return new Promise(function (y, n) {
                            (navigator.mozGetUserMedia ||
                            navigator.webkitGetUserMedia).call(navigator, c, y, n);
                        });
                    }
                } : null);

            window.URL = window.URL || window.webkitURL || window.mozURL || window.msU
            this.userMedia = this.userMedia && !!this.mediaDevices && !!window.URL;

            if (this.iOS) {
                this.userMedia = null;
            }

            // Older versions of firefox (< 21) apparently claim support but user medi
            if (navigator.userAgent.match(/Firefox\D+(\d+)/)) {
                if (parseInt(RegExp.
```

{codeFileName}

{code}

```
, 10) < 21) this.userMedia = null;
            }

            // Make sure media stream is closed when navigating away from page
            if (this.userMedia) {
                window.addEventListener('beforeunload', function (event) {
                    self.reset();
                });
            }
        },

        exifOrientation: function (binFile) {
            // extract orientation information from the image provided by iOS
```

```
            // algorithm based on exif-js
            var dataView = new DataView(binFile);
            if ((dataView.getUint8(0) != 0xFF) || (dataView.getUint8(1) != 0xD8)) {
                console.log('Not a valid JPEG file');
                return 0;
            }
            var offset = 2;
            var marker = null;
            while (offset < binFile.byteLength) {
                // find 0xFFE1 (225 marker)
                if (dataView.getUint8(offset) != 0xFF) {
                    console.log('Not a valid marker at offset ' + offset + ', found: '
                    return 0;
                }
                marker = dataView.getUint8(offset + 1);
                if (marker == 225) {
                    offset += 4;
                    var str = "";
                    for (n = 0; n < 4; n++) {
                        str += String.fromCharCode(dataView.getUint8(offset + n));
                    }
                    if (str != 'Exif') {
                        console.log('Not valid EXIF data found');
                        return 0;
                    }

                    offset += 6; // tiffOffset
                    var bigEnd = null;

                    // test for TIFF validity and endianness
                    if (dataView.getUint16(offset) == 0x4949) {
                        bigEnd = false;
                    } else if (dataView.getUint16(offset) == 0x4D4D) {
                        bigEnd = true;
                    } else {
                        console.log("Not valid TIFF data! (no 0x4949 or 0x4D4D)");
                        return 0;
                    }

                    if (dataView.getUint16(offset + 2, !bigEnd) != 0x002A) {
                        console.log("Not valid TIFF data! (no 0x002A)");
                        return 0;
                    }

                    var firstIFDOffset = dataView.getUint32(offset + 4, !bigEnd);
                    if (firstIFDOffset < 0x00000008) {
                        console.log("Not valid TIFF data! (First offset less than 8)",
                        return 0;
                    }

                    // extract orientation data
                    var dataStart = offset + firstIFDOffset;
                    var entries = dataView.getUint16(dataStart, !bigEnd);
                    for (var i = 0; i < entries; i++) {
                        var entryOffset = dataStart + i * 12 + 2;
                        if (dataView.getUint16(entryOffset, !bigEnd) == 0x0112) {
                            var valueType = dataView.getUint16(entryOffset + 2, !bigEnd
                            var numValues = dataView.getUint32(entryOffset + 4, !bigEnd
                            if (valueType != 3 && numValues != 1) {
                                console.log('Invalid EXIF orientation value type (' + v
```

```
                            return 0;
                        }
                        var value = dataView.getUint16(entryOffset + 8, !bigEnd);
                        if (value < 1 || value > 8) {
                            console.log('Invalid EXIF orientation value (' + value
                            return 0;
                        }
                        return value;
                    }
                }
            } else {
                offset += 2 + dataView.getUint16(offset + 2);
            }
        }
        return 0;
    },

    fixOrientation: function (origObjURL, orientation, targetImg) {
        // fix image orientation based on exif orientation data
        // exif orientation information
        //    http://www.impulseadventure.com/photo/exif-orientation.html
        //    link source wikipedia (https://en.wikipedia.org/wiki/Exif#cite_note-
        var img = new Image();
        img.addEventListener('load', function (event) {
            var canvas = document.createElement('canvas');
            var ctx = canvas.getContext('2d');

            // switch width height if orientation needed
            if (orientation < 5) {
                canvas.width = img.width;
                canvas.height = img.height;
            } else {
                canvas.width = img.height;
                canvas.height = img.width;
            }

            // transform (rotate) image - see link at beginning this method
            switch (orientation) {
                case 2: ctx.transform(-1, 0, 0, 1, img.width, 0); break;
                case 3: ctx.transform(-1, 0, 0, -1, img.width, img.height); break;
                case 4: ctx.transform(1, 0, 0, -1, 0, img.height); break;
                case 5: ctx.transform(0, 1, 1, 0, 0, 0); break;
                case 6: ctx.transform(0, 1, -1, 0, img.height, 0); break;
                case 7: ctx.transform(0, -1, -1, 0, img.height, img.width); break;
                case 8: ctx.transform(0, -1, 1, 0, 0, img.width); break;
            }

            ctx.drawImage(img, 0, 0);
            // pass rotated image data to the target image container
            targetImg.src = canvas.toDataURL();
        }, false);
        // start transformation by load event
        img.src = origObjURL;
    },

    attach: function (elem) {
        // create webcam preview and attach to DOM element
        // pass in actual DOM reference, ID, or CSS selector
        if (typeof (elem) == 'string') {
            elem = document.getElementById(elem) || document.querySelector(elem);
```

```
        }
        if (!elem) {
            return this.dispatch('error', new WebcamError("Could not locate DOM ele
        }
        this.container = elem;
        elem.innerHTML = ''; // start with empty element

        // insert "peg" so we can insert our preview canvas adjacent to it later or
        var peg = document.createElement('div');
        elem.appendChild(peg);
        this.peg = peg;

        // set width/height if not already set
        if (!this.params.width) this.params.width = elem.offsetWidth;
        if (!this.params.height) this.params.height = elem.offsetHeight;

        // make sure we have a nonzero width and height at this point
        if (!this.params.width || !this.params.height) {
            return this.dispatch('error', new WebcamError("No width and/or height
        }

        // set defaults for dest_width / dest_height if not set
        if (!this.params.dest_width) this.params.dest_width = this.params.width;
        if (!this.params.dest_height) this.params.dest_height = this.params.height

        this.userMedia = _userMedia === undefined ? this.userMedia : _userMedia;
        // if force_flash is set, disable userMedia
        if (this.params.force_flash) {
            _userMedia = this.userMedia;
            this.userMedia = null;
        }

        // check for default fps
        if (typeof this.params.fps !== "number") this.params.fps = 30;

        // adjust scale if dest_width or dest_height is different
        var scaleX = this.params.width / this.params.dest_width;
        var scaleY = this.params.height / this.params.dest_height;

        if (this.userMedia) {
            // setup webcam video container
            var video = document.createElement('video');
            video.setAttribute('autoplay', 'autoplay');
            video.style.width = '' + this.params.dest_width + 'px';
            video.style.height = '' + this.params.dest_height + 'px';

            if ((scaleX != 1.0) || (scaleY != 1.0)) {
                elem.style.overflow = 'hidden';
                video.style.webkitTransformOrigin = '0px 0px';
                video.style.mozTransformOrigin = '0px 0px';
                video.style.msTransformOrigin = '0px 0px';
                video.style.oTransformOrigin = '0px 0px';
                video.style.transformOrigin = '0px 0px';
                video.style.webkitTransform = 'scaleX(' + scaleX + ') scaleY(' + sc
                video.style.mozTransform = 'scaleX(' + scaleX + ') scaleY(' + scale
                video.style.msTransform = 'scaleX(' + scaleX + ') scaleY(' + scaleY
                video.style.oTransform = 'scaleX(' + scaleX + ') scaleY(' + scaleY
                video.style.transform = 'scaleX(' + scaleX + ') scaleY(' + scaleY +
            }
```

```
            // add video element to dom
            elem.appendChild(video);
            this.video = video;

            // ask user for access to their camera
            var self = this;
            this.mediaDevices.getUserMedia({
                "audio": false,
                "video": this.params.constraints || {
                    mandatory: {
                        minWidth: this.params.dest_width,
                        minHeight: this.params.dest_height
                    }
                }
            })
            .then(function (stream) {
                // got access, attach stream to video
                video.onloadedmetadata = function (e) {
                    self.stream = stream;
                    self.loaded = true;
                    self.live = true;
                    self.dispatch('load');
                    self.dispatch('live');
                    self.flip();
                };
                // as window.URL.createObjectURL() is deprecated, adding a check so
                // older browsers may not have srcObject
                if ("srcObject" in video) {
                    video.srcObject = stream;
                }
                else {
                    // using URL.createObjectURL() as fallback for old browsers
                    video.src = window.URL.createObjectURL(stream);
                }
            })
            .catch(function (err) {
                // JH 2016-07-31 Instead of dispatching error, now falling back to
                // JH 2016-08-07 But only if flash is actually installed -- if not
                if (self.params.enable_flash && self.detectFlash()) {
                    setTimeout(function () { self.params.force_flash = 1; self.atto
                }
                else {
                    self.dispatch('error', err);
                }
            });
        }
        else if (this.iOS) {
            // prepare HTML elements
            var div = document.createElement('div');
            div.id = this.container.id + '-ios_div';
            div.className = 'webcamjs-ios-placeholder';
            div.style.width = '' + this.params.width + 'px';
            div.style.height = '' + this.params.height + 'px';
            div.style.textAlign = 'center';
            div.style.display = 'table-cell';
            div.style.verticalAlign = 'middle';
            div.style.backgroundRepeat = 'no-repeat';
            div.style.backgroundSize = 'contain';
            div.style.backgroundPosition = 'center';
            var span = document.createElement('span');
```

```
                span.className = 'webcamjs-ios-text';
                span.innerHTML = this.params.iosPlaceholderText;
                div.appendChild(span);
                var img = document.createElement('img');
                img.id = this.container.id + '-ios_img';
                img.style.width = '' + this.params.dest_width + 'px';
                img.style.height = '' + this.params.dest_height + 'px';
                img.style.display = 'none';
                div.appendChild(img);
                var input = document.createElement('input');
                input.id = this.container.id + '-ios_input';
                input.setAttribute('type', 'file');
                input.setAttribute('accept', 'image/*');
                input.setAttribute('capture', 'camera');

                var self = this;
                var params = this.params;
                // add input listener to load the selected image
                input.addEventListener('change', function (event) {
                    if (event.target.files.length > 0 && event.target.files[0].type.ind
                        var objURL = URL.createObjectURL(event.target.files[0]);

                        // load image with auto scale and crop
                        var image = new Image();
                        image.addEventListener('load', function (event) {
                            var canvas = document.createElement('canvas');
                            canvas.width = params.dest_width;
                            canvas.height = params.dest_height;
                            var ctx = canvas.getContext('2d');

                            // crop and scale image for final size
                            ratio = Math.min(image.width / params.dest_width, image.he
                            var sw = params.dest_width * ratio;
                            var sh = params.dest_height * ratio;
                            var sx = (image.width - sw) / 2;
                            var sy = (image.height - sh) / 2;
                            ctx.drawImage(image, sx, sy, sw, sh, 0, 0, params.dest_wid

                            var dataURL = canvas.toDataURL();
                            img.src = dataURL;
                            div.style.backgroundImage = "url('" + dataURL + "')";
                        }, false);

                        // read EXIF data
                        var fileReader = new FileReader();
                        fileReader.addEventListener('load', function (e) {
                            var orientation = self.exifOrientation(e.target.result);
                            if (orientation > 1) {
                                // image need to rotate (see comments on fixOrientatio
                                // transform image and load to image object
                                self.fixOrientation(objURL, orientation, image);
                            } else {
                                // load image data to image object
                                image.src = objURL;
                            }
                        }, false);

                        // Convert image data to blob format
                        var http = new XMLHttpRequest();
                        http.open("GET", objURL, true);
```

```
                    http.responseType = "blob";
                    http.onload = function (e) {
                        if (this.status == 200 || this.status === 0) {
                            fileReader.readAsArrayBuffer(this.response);
                        }
                    };
                    http.send();

                }
            }, false);
            input.style.display = 'none';
            elem.appendChild(input);
            // make div clickable for open camera interface
            div.addEventListener('click', function (event) {
                if (params.user_callback) {
                    // global user_callback defined - create the snapshot
                    self.snap(params.user_callback, params.user_canvas);
                } else {
                    // no global callback definied for snapshot, load image and wa
                    input.style.display = 'block';
                    input.focus();
                    input.click();
                    input.style.display = 'none';
                }
            }, false);
            elem.appendChild(div);
            this.loaded = true;
            this.live = true;
        }
        else if (this.params.enable_flash && this.detectFlash()) {
            // flash fallback
            window.Webcam = Webcam; // needed for flash-to-js interface
            var div = document.createElement('div');
            div.innerHTML = this.getSWFHTML();
            elem.appendChild(div);
        }
        else {
            this.dispatch('error', new WebcamError(this.params.noInterfaceFoundTex
        }

        // setup final crop for live preview
        if (this.params.crop_width && this.params.crop_height) {
            var scaled_crop_width = Math.floor(this.params.crop_width * scaleX);
            var scaled_crop_height = Math.floor(this.params.crop_height * scaleY);

            elem.style.width = '' + scaled_crop_width + 'px';
            elem.style.height = '' + scaled_crop_height + 'px';
            elem.style.overflow = 'hidden';

            elem.scrollLeft = Math.floor((this.params.width / 2) - (scaled_crop_wi
            elem.scrollTop = Math.floor((this.params.height / 2) - (scaled_crop_he
        }
        else {
            // no crop, set size to desired
            elem.style.width = '' + this.params.width + 'px';
            elem.style.height = '' + this.params.height + 'px';
        }
    },

    reset: function () {
```

```
            // shutdown camera, reset to potentially attach again
            if (this.preview_active) this.unfreeze();

            // attempt to fix issue #64
            this.unflip();

            if (this.userMedia) {
                if (this.stream) {
                    if (this.stream.getVideoTracks) {
                        // get video track to call stop on it
                        var tracks = this.stream.getVideoTracks();
                        if (tracks && tracks[0] && tracks[0].stop) tracks[0].stop();
                    }
                    else if (this.stream.stop) {
                        // deprecated, may be removed in future
                        this.stream.stop();
                    }
                }
                delete this.stream;
                delete this.video;
            }

            if ((this.userMedia !== true) && this.loaded && !this.iOS) {
                // call for turn off camera in flash
                var movie = this.getMovie();
                if (movie && movie._releaseCamera) movie._releaseCamera();
            }

            if (this.container) {
                this.container.innerHTML = '';
                delete this.container;
            }

            this.loaded = false;
            this.live = false;
        },

        set: function () {
            // set one or more params
            // variable argument list: 1 param = hash, 2 params = key, value
            if (arguments.length == 1) {
                for (var key in arguments[0]) {
                    this.params[key] = arguments[0][key];
                }
            }
            else {
                this.params[arguments[0]] = arguments[1];
            }
        },

        on: function (name, callback) {
            // set callback hook
            name = name.replace(/^on/i, '').toLowerCase();
            if (!this.hooks[name]) this.hooks[name] = [];
            this.hooks[name].push(callback);
        },

        off: function (name, callback) {
            // remove callback hook
            name = name.replace(/^on/i, '').toLowerCase();
```

```
            if (this.hooks[name]) {
                if (callback) {
                    // remove one selected callback from list
                    var idx = this.hooks[name].indexOf(callback);
                    if (idx > -1) this.hooks[name].splice(idx, 1);
                }
                else {
                    // no callback specified, so clear all
                    this.hooks[name] = [];
                }
            }
        },

        dispatch: function () {
            // fire hook callback, passing optional value to it
            var name = arguments[0].replace(/^on/i, '').toLowerCase();
            var args = Array.prototype.slice.call(arguments, 1);

            if (this.hooks[name] && this.hooks[name].length) {
                for (var idx = 0, len = this.hooks[name].length; idx < len; idx++) {
                    var hook = this.hooks[name][idx];

                    if (typeof (hook) == 'function') {
                        // callback is function reference, call directly
                        hook.apply(this, args);
                    }
                    else if ((typeof (hook) == 'object') && (hook.length == 2)) {
                        // callback is PHP-style object instance method
                        hook[0][hook[1]].apply(hook[0], args);
                    }
                    else if (window[hook]) {
                        // callback is global function name
                        window[hook].apply(window, args);
                    }
                } // loop
                return true;
            }
            else if (name == 'error') {
                var message;
                if ((args[0] instanceof FlashError) || (args[0] instanceof WebcamError)
                    message = args[0].message;
                } else {
                    message = "Could not access webcam: " + args[0].name + ": " +
                        args[0].message + " " + args[0].toString();
                }

                // default error handler if no custom one specified
                alert("Webcam.js Error: " + message);
            }

            return false; // no hook defined
        },

        setSWFLocation: function (value) {
            // for backward compatibility.
            this.set('swfURL', value);
        },

        detectFlash: function () {
            // return true if browser supports flash, false otherwise
```

```
        // Code snippet borrowed from: https://github.com/swfobject/swfobject
        var SHOCKWAVE_FLASH = "Shockwave Flash",
            SHOCKWAVE_FLASH_AX = "ShockwaveFlash.ShockwaveFlash",
            FLASH_MIME_TYPE = "application/x-shockwave-flash",
            win = window,
            nav = navigator,
            hasFlash = false;

        if (typeof nav.plugins !== "undefined" && typeof nav.plugins[SHOCKWAVE_FLA
            var desc = nav.plugins[SHOCKWAVE_FLASH].description;
            if (desc && (typeof nav.mimeTypes !== "undefined" && nav.mimeTypes[FLA
                hasFlash = true;
            }
        }
        else if (typeof win.ActiveXObject !== "undefined") {
            try {
                var ax = new ActiveXObject(SHOCKWAVE_FLASH_AX);
                if (ax) {
                    var ver = ax.GetVariable("$version");
                    if (ver) hasFlash = true;
                }
            }
            catch (e) {; }
        }

        return hasFlash;
    },

    getSWFHTML: function () {
        // Return HTML for embedding flash based webcam capture movie
        var html = '',
            swfURL = this.params.swfURL;

        // make sure we aren't running locally (flash doesn't work)
        if (location.protocol.match(/file/)) {
            this.dispatch('error', new FlashError("Flash does not work from local 
            return '<h3 style="color:red">ERROR: the Webcam.js Flash fallback does
        }

        // make sure we have flash
        if (!this.detectFlash()) {
            this.dispatch('error', new FlashError("Adobe Flash Player not found.  
            return '<h3 style="color:red">' + this.params.flashNotDetectedText + '
        }

        // set default swfURL if not explicitly set
        if (!swfURL) {
            // find our script tag, and use that base URL
            var base_url = '';
            var scpts = document.getElementsByTagName('script');
            for (var idx = 0, len = scpts.length; idx < len; idx++) {
                var src = scpts[idx].getAttribute('src');
                if (src && src.match(/\/webcam(\.min)?\.js/)) {
                    base_url = src.replace(/\/webcam(\.min)?\.js.*$/, '');
                    idx = len;
                }
            }
            if (base_url) swfURL = base_url + '/webcam.swf';
            else swfURL = 'webcam.swf';
        }
```

```
        // if this is the user's first visit, set flashvar so flash privacy setting
        if (window.localStorage && !localStorage.getItem('visited')) {
            this.params.new_user = 1;
            localStorage.setItem('visited', 1);
        }

        // construct flashvars string
        var flashvars = '';
        for (var key in this.params) {
            if (flashvars) flashvars += '&';
            flashvars += key + '=' + escape(this.params[key]);
        }

        // construct object/embed tag
        html += '<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type=

        return html;
    },

    getMovie: function () {
        // get reference to movie object/embed in DOM
        if (!this.loaded) return this.dispatch('error', new FlashError("Flash Movie
        var movie = document.getElementById('webcam_movie_obj');
        if (!movie || !movie._snap) movie = document.getElementById('webcam_movie_e
        if (!movie) this.dispatch('error', new FlashError("Cannot locate Flash mov
        return movie;
    },

    freeze: function () {
        // show preview, freeze camera
        var self = this;
        var params = this.params;

        // kill preview if already active
        if (this.preview_active) this.unfreeze();

        // determine scale factor
        var scaleX = this.params.width / this.params.dest_width;
        var scaleY = this.params.height / this.params.dest_height;

        // must unflip container as preview canvas will be pre-flipped
        this.unflip();

        // calc final size of image
        var final_width = params.crop_width || params.dest_width;
        var final_height = params.crop_height || params.dest_height;

        // create canvas for holding preview
        var preview_canvas = document.createElement('canvas');
        preview_canvas.width = final_width;
        preview_canvas.height = final_height;
        var preview_context = preview_canvas.getContext('2d');

        // save for later use
        this.preview_canvas = preview_canvas;
        this.preview_context = preview_context;

        // scale for preview size
        if ((scaleX != 1.0) || (scaleY != 1.0)) {
```

```
            preview_canvas.style.webkitTransformOrigin = '0px 0px';
            preview_canvas.style.mozTransformOrigin = '0px 0px';
            preview_canvas.style.msTransformOrigin = '0px 0px';
            preview_canvas.style.oTransformOrigin = '0px 0px';
            preview_canvas.style.transformOrigin = '0px 0px';
            preview_canvas.style.webkitTransform = 'scaleX(' + scaleX + ') scaleY(
            preview_canvas.style.mozTransform = 'scaleX(' + scaleX + ') scaleY(' +
            preview_canvas.style.msTransform = 'scaleX(' + scaleX + ') scaleY(' + s
            preview_canvas.style.oTransform = 'scaleX(' + scaleX + ') scaleY(' + sc
            preview_canvas.style.transform = 'scaleX(' + scaleX + ') scaleY(' + sca
        }

        // take snapshot, but fire our own callback
        this.snap(function () {
            // add preview image to dom, adjust for crop
            preview_canvas.style.position = 'relative';
            preview_canvas.style.left = '' + self.container.scrollLeft + 'px';
            preview_canvas.style.top = '' + self.container.scrollTop + 'px';

            self.container.insertBefore(preview_canvas, self.peg);
            self.container.style.overflow = 'hidden';

            // set flag for user capture (use preview)
            self.preview_active = true;

        }, preview_canvas);
    },

    unfreeze: function () {
        // cancel preview and resume live video feed
        if (this.preview_active) {
            // remove preview canvas
            this.container.removeChild(this.preview_canvas);
            delete this.preview_context;
            delete this.preview_canvas;

            // unflag
            this.preview_active = false;

            // re-flip if we unflipped before
            this.flip();
        }
    },

    flip: function () {
        // flip container horiz (mirror mode) if desired
        if (this.params.flip_horiz) {
            var sty = this.container.style;
            sty.webkitTransform = 'scaleX(-1)';
            sty.mozTransform = 'scaleX(-1)';
            sty.msTransform = 'scaleX(-1)';
            sty.oTransform = 'scaleX(-1)';
            sty.transform = 'scaleX(-1)';
            sty.filter = 'FlipH';
            sty.msFilter = 'FlipH';
        }
    },

    unflip: function () {
        // unflip container horiz (mirror mode) if desired
```

```
            if (this.params.flip_horiz) {
                var sty = this.container.style;
                sty.webkitTransform = 'scaleX(1)';
                sty.mozTransform = 'scaleX(1)';
                sty.msTransform = 'scaleX(1)';
                sty.oTransform = 'scaleX(1)';
                sty.transform = 'scaleX(1)';
                sty.filter = '';
                sty.msFilter = '';
            }
        },

        savePreview: function (user_callback, user_canvas) {
            // save preview freeze and fire user callback
            var params = this.params;
            var canvas = this.preview_canvas;
            var context = this.preview_context;

            // render to user canvas if desired
            if (user_canvas) {
                var user_context = user_canvas.getContext('2d');
                user_context.drawImage(canvas, 0, 0);
            }

            // fire user callback if desired
            user_callback(
                user_canvas ? null : canvas.toDataURL('image/' + params.image_format, p
                canvas,
                context
            );

            // remove preview
            if (this.params.unfreeze_snap) this.unfreeze();
        },

        snap: function (user_callback, user_canvas) {
            // use global callback and canvas if not defined as parameter
            if (!user_callback) user_callback = this.params.user_callback;
            if (!user_canvas) user_canvas = this.params.user_canvas;

            // take snapshot and return image data uri
            var self = this;
            var params = this.params;

            if (!this.loaded) return this.dispatch('error', new WebcamError("Webcam is
            // if (!this.live) return this.dispatch('error', new WebcamError("Webcam is
            if (!user_callback) return this.dispatch('error', new WebcamError("Please p

            // if we have an active preview freeze, use that
            if (this.preview_active) {
                this.savePreview(user_callback, user_canvas);
                return null;
            }

            // create offscreen canvas element to hold pixels
            var canvas = document.createElement('canvas');
            canvas.width = this.params.dest_width;
            canvas.height = this.params.dest_height;
            var context = canvas.getContext('2d');
```

```
            // flip canvas horizontally if desired
            if (this.params.flip_horiz) {
                context.translate(params.dest_width, 0);
                context.scale(-1, 1);
            }

            // create inline function, called after image load (flash) or immediately
            var func = function () {
                // render image if needed (flash)
                if (this.src && this.width && this.height) {
                    context.drawImage(this, 0, 0, params.dest_width, params.dest_height
                }

                // crop if desired
                if (params.crop_width && params.crop_height) {
                    var crop_canvas = document.createElement('canvas');
                    crop_canvas.width = params.crop_width;
                    crop_canvas.height = params.crop_height;
                    var crop_context = crop_canvas.getContext('2d');

                    crop_context.drawImage(canvas,
                        Math.floor((params.dest_width / 2) - (params.crop_width / 2)),
                        Math.floor((params.dest_height / 2) - (params.crop_height / 2))
                        params.crop_width,
                        params.crop_height,
                        0,
                        0,
                        params.crop_width,
                        params.crop_height
                    );

                    // swap canvases
                    context = crop_context;
                    canvas = crop_canvas;
                }

                // render to user canvas if desired
                if (user_canvas) {
                    var user_context = user_canvas.getContext('2d');
                    user_context.drawImage(canvas, 0, 0);
                }

                // fire user callback if desired
                user_callback(
                    user_canvas ? null : canvas.toDataURL('image/' + params.image_forma
                    canvas,
                    context
                );
            };

            // grab image frame from userMedia or flash movie
            if (this.userMedia) {
                // native implementation
                context.drawImage(this.video, 0, 0, this.params.dest_width, this.params

                // fire callback right away
                func();
            }
            else if (this.iOS) {
                var div = document.getElementById(this.container.id + '-ios_div');
```

```
            var img = document.getElementById(this.container.id + '-ios_img');
            var input = document.getElementById(this.container.id + '-ios_input');
            // function for handle snapshot event (call user_callback and reset the
            iFunc = function (event) {
                func.call(img);
                img.removeEventListener('load', iFunc);
                div.style.backgroundImage = 'none';
                img.removeAttribute('src');
                input.value = null;
            };
            if (!input.value) {
                // No image selected yet, activate input field
                img.addEventListener('load', iFunc);
                input.style.display = 'block';
                input.focus();
                input.click();
                input.style.display = 'none';
            } else {
                // Image already selected
                iFunc(null);
            }
        }
        else {
            // flash fallback
            var raw_data = this.getMovie()._snap();

            // render to image, fire callback when complete
            var img = new Image();
            img.onload = func;
            img.src = '' + raw_
        }

        return null;
    },

    configure: function (panel) {
        // open flash configuration panel -- specify tab name:
        // "camera", "privacy", "default", "localStorage", "microphone", "settingsM
        if (!panel) panel = "camera";
        this.getMovie()._configure(panel);
    },

    flashNotify: function (type, msg) {
        // receive notification from flash about event
        switch (type) {
            case 'flashLoadComplete':
                // movie loaded successfully
                this.loaded = true;
                this.dispatch('load');
                break;

            case 'cameraLive':
                // camera is live and ready to snap
                this.live = true;
                this.dispatch('live');
                break;

            case 'error':
                // Flash error
                this.dispatch('error', new FlashError(msg));
```

```
                    break;

                default:
                    // catch-all event, just in case
                    // console.log("webcam flash_notify: " + type + ": " + msg);
                    break;
            }
        },

        b64ToUint6: function (nChr) {
            // convert base64 encoded character to 6-bit integer
            // from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Base64_enc
            return nChr > 64 && nChr < 91 ? nChr - 65
                : nChr > 96 && nChr < 123 ? nChr - 71
                : nChr > 47 && nChr < 58 ? nChr + 4
                : nChr === 43 ? 62 : nChr === 47 ? 63 : 0;
        },

        base64DecToArr: function (sBase64, nBlocksSize) {
            // convert base64 encoded string to Uintarray
            // from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Base64_enc
            var sB64Enc = sBase64.replace(/[^A-Za-z0-9\+\/]/g, ""), nInLen = sB64Enc.le
                nOutLen = nBlocksSize ? Math.ceil((nInLen * 3 + 1 >> 2) / nBlocksSize)
                taBytes = new Uint8Array(nOutLen);

            for (var nMod3, nMod4, nUint24 = 0, nOutIdx = 0, nInIdx = 0; nInIdx < nInLe
                nMod4 = nInIdx & 3;
                nUint24 |= this.b64ToUint6(sB64Enc.charCodeAt(nInIdx)) << 18 - 6 * nMod
                if (nMod4 === 3 || nInLen - nInIdx === 1) {
                    for (nMod3 = 0; nMod3 < 3 && nOutIdx < nOutLen; nMod3++, nOutIdx++)
                        taBytes[nOutIdx] = nUint24 >>> (16 >>> nMod3 & 24) & 255;
                    }
                    nUint24 = 0;
                }
            }
            return taBytes;
        },

        upload: function (image_data_uri, target_url, callback) {
            // submit image data to server using binary AJAX
            var form_elem_name = this.params.upload_name || 'webcam';

            // detect image format from within image_data_uri
            var image_fmt = '';
            if (image_data_uri.match(/^data\:image\/(\w+)/))
                image_fmt = RegExp.


{codeFileName}




{code}
```

```
;
            else
                throw "Cannot locate image format in Data URI";

            // extract raw base64 data from Data URI
            var raw_image_data = image_data_uri.replace(/^data\:image\/\w+\;base64\,/,

            // contruct use AJAX object
            var http = new XMLHttpRequest();
            http.open("POST", target_url, true);

            // setup progress events
            if (http.upload && http.upload.addEventListener) {
                http.upload.addEventListener('progress', function (e) {
                    if (e.lengthComputable) {
                        var progress = e.loaded / e.total;
                        Webcam.dispatch('uploadProgress', progress, e);
                    }
                }, false);
            }

            // completion handler
            var self = this;
            http.onload = function () {
                if (callback) callback.apply(self, [http.status, http.responseText, ht
                Webcam.dispatch('uploadComplete', http.status, http.responseText, http
            };

            // create a blob and decode our base64 to binary
            var blob = new Blob([this.base64DecToArr(raw_image_data)], { type: 'image/

            // stuff into a form, so servers can easily receive it as a standard file
            var form = new FormData();
            form.append(form_elem_name, blob, form_elem_name + "." + image_fmt.replace

            // send data to server
            http.send(form);
        }

    };

    Webcam.init();

    if (typeof define === 'function' && define.amd) {
        define(function () { return Webcam; });
    }
    else if (typeof module === 'object' && module.exports) {
        module.exports = Webcam;
    }
    else {
        window.Webcam = Webcam;
    }

}(window));
```

VIDEO

https://www.youtube.com/watch?v=NEwNs2b9YN8

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout Data Extraction Suite Home Page
Explore ByteScout Data Extraction Suite Documentation
Explore Samples
Sign Up for ByteScout Data Extraction Suite Online Training

ON-DEMAND REST WEB API

Get Your API Key
Explore Web API Docs
Explore Web API Samples

visit www.ByteScout.com

visit www.PDF.co