Intel® RealSense™ SDK

Browser Support Tutorial

With the Intel® RealSense™ SDK, you have access to robust, natural human-computer interaction (HCI) algorithms such as face tracking, finger tracking, gesture recognition, speech recognition and synthesis, fully textured 3D scanning and enhanced depth augmented reality.

Using the SDK you can create web applications and games that offer innovative user experiences.

This tutorial explains how to use the SDK with JavaScript* for HTML5 web pages and how to set up Unity* software for building applications that run on the Unity web player.

Contents

- **Overview**
  - JavaScript* Programs
- **JavaScript Programs**
  - Check User Platform for SDK Compatibility
  - Hand Tracking Code Sample
  - Creating a Session
  - Initialize the Pipeline and Capture Data
  - Cleaning Up
- **Run the Tutorial Code Samples**
- **To learn more**

Overview

This tutorial explains how to use the Intel RealSense SDK for web development.

JavaScript* Programs

The SDK provides browser support for JavaScript in an HTML5 page. This tutorial explains how to set up and run the Hand Module with JavaScript in an HTML5 page.

A complete list of supported functions in the SDK can be found in the List of Supported Functions section under JavaScript Programs in the SDK Reference Manual.

Table 1: Code Samples

Code Sample	For more information, see:
Capturing Hand Data using JavaScript in an HTML5 page File: JavaScript_Hand_Tracking/index.cpp	This Tutorial. Also see Hand Tracking Data section in the SDK Reference Manual.
Capturing Face Data and Voice Recognition using JavaScript in an HTML5 page	Face Tracking Data and Speech Recognition Data sections in the SDK Reference Manual.

A JavaScript sample is provided in the SDK located at **RSSDK/framework/JavaScript** directory.

JavaScript Programs

The following two short tutorials assume you have basic knowledge of JavaScript and HTML5.

Check User Platform for SDK Compatibility

Refer to the Visual Studio* Tutorial Solution Project directory: RealSense_Info

The **RealSenseInfo** JavaScript provides basic checks, like whether the **RealSenseWeb** local websocket server is not installed or outdated. You can then prompt the user to install/upgrade the websocket server using a dialog box like the one shown in Figure 1.

Figure 1: Prompt the user for incompatibility.

If you are starting a project from scratch, make sure to include the script in the project directory by right clicking the current project directory and **Add->Existing Item...** located at:

$(RSSDK_DIR)/framework/JavaScript/common/realsenseinfo.js

Your project directory should now look like this (Figure 2).

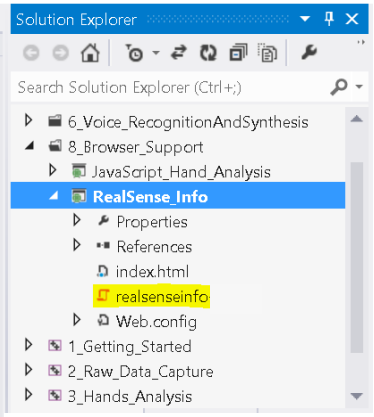

Figure 2. Visual Studio* Project Directory Structure

Include the script with the appropriate path in the **head** section of your HTML5 page.

```
<head>
<script type="text/javascript" src="realsenseinfo.js"></script>
</head>
```

The **RealSenseInfo** function takes 2 arguments:

- Components: list of algorithm components used by the application
- Callback: receives the platform information

Use the callback to retrieve the following information:

1. Check if the browser supports the websocket using **IsBrowserSupported**.
2. Check if the platform is equipped with an Intel® RealSense™ Camera using **IsPlatformSupported**.
3. Check if any updates are needed by verifying the length.

If the length is more than 0, display the **url**, **name**, or **href.**

```
<script type="text/javascript">
    function LoadService() {
      RealSenseInfo(['face3d', 'hand', 'nuance'], function (info) {
        if (info.IsBrowserSupported != true) {
          $("#service").append("<p>Websocket protocol is not supported</p>");  return;
        }

        if (info.IsPlatformSupported != true) {
          $("#service").append("<p>Intel RealSense 3D camera is not found.</p>");return;
        }
        if (info.Updates.length > 0) {
          $("#service").append("<p>Please download and install the following update(s):</p><ul>");
          for (i = 0; i < info.Updates.length; i++) {
            $("#service").append("<li>" + info.Updates[i].href + "</li>");
          }
          $("#service").append("</ul>"); return;
        }

        if (info.IsReady != true) {
          $("#service").append("<p>The platform is not ready to run Intel® RealSense™ SDK applications.</p>");
          return;
        }
      });
    }
  </script>
```

Hand Tracking Code Sample

Refer to the Visual Studio Tutorial Solution Project Directory: JavaScript_Hand_Tracking

Make sure to perform the checks mentioned in the previous section before beginning the hand tracking module.

To access the Intel RealSense SDK modules and components include the appropriate scripts, such as **hand**, **face**, and **speech modules** along with the **SenseManager** and **Session** instances.

```
<head>
<script type="text/javascript" src="promise-1.0.0.min.js"></script>
<script type="text/javascript" src="realsenseinfo.js"></script>
</head>
```

Creating a Session

The SDK core is represented by two interfaces:

- **PXCMSession:** manages all of the modules of the SDK
- **PXCMSenseManager:** organizes a pipeline by starting, stopping, and pausing the operations of its various modalities.

The first step when creating an application that uses the Intel RealSense SDK is to create a session. A session can be created explicitly by creating an instance of **PXCMSession**. Each session maintains its own pipeline that contains the I/O and algorithm modules.

Another way of creating a session is by creating an instance of the **PXCMSenseManager** using **CreateInstance**. The PXCMSenseManager implicitly creates a session internally.

Initialize the Pipeline and Capture Data

Retrieve a SenseManager instance using **PXCMSenseManager_CreateInstance**.

1. Enable the hand using **EnableHand** with the function **OnHandData** that takes 3 arguments **mid** (callback module id), **module** (module instance), and **data**.
1. Save the returned **HandModule** instance and initialize the pipeline using **init** on the SenseManager instance.
2. Retrieve a hand configuration using **CreateHandConfiguration** on the **HandModule** instance.
3. Disable all alerts, enable all gestures and apply the changes.
4. You can use **QueryCaptureManager** on the sense manager and then **QueryImageSize** instance on the returned **Capture Manager** instance to retrieve the size of the image to adjust the **HTML5 canvas size**.
5. Begin streaming from the camera using **StreamFrames**.

You can also check for errors and display after parsing the JSON string using **catch**(error) function and then **JSON.stringify.**

```
$('#Start').click(function () {
            document.getElementById("Start").disabled = true;
            PXCMSenseManager_CreateInstance().then(function (result) {
               sense = result;
               return sense.EnableHand(onHandData);
            }).then(function (result) {
               handModule = result;
               status('Init started');
               return sense.Init(onConnect);
            }).then(function (result) {
               return handModule.CreateHandConfiguration();
            }).then(function (result) {
               handConfiguration = result;
               return handConfiguration.DisableAllAlerts();
            }).then(function (result) {
               return handConfiguration.EnableAllGestures(false);
            }).then(function (result) {
               return handConfiguration.ApplyChanges();
            }).then(function (result) {
               return sense.QueryCaptureManager();
            }).then(function (capture) {
               return capture.QueryImageSize(pxcmConst.PXCMCapture.STREAM_TYPE_DEPTH);
            }).then(function (result) {
               imageSize = result.size;
               return sense.StreamFrames();
            }).then(function (result) {
               status('Streaming ' + imageSize.width + 'x' + imageSize.height);
               document.getElementById("Stop").disabled = false;
            }).catch(function (error) {
               status('Init failed: ' + JSON.stringify(error));
               document.getElementById("Start").disabled = false;
            });
         });
```

You can then capture the joint data using the **OnHandData** function. You can retrieve an HTML5 canvas and then retrieve the canvas context to draw using the data from the data variable.

You can also retrieve the gesture data using the data variable. Make sure to use **JSON.stringify** to parse. You can view the hand, face, and voice recognition data structure by referring to the corresponding sections under Browser Support from the SDK reference manual.

Cleaning Up

Make sure to release the **SenseManager** instance by calling **Close** on it to stop streaming from the camera.

Run the Tutorial Code Samples

Locate the index.html file in the **RealSense_info** and **JavaScript_Hand_Tracking** project directories to run the code sample. Alternately, you can load the Visual Studio Tutorial Project Solution and run the corresponding projects.

Figure 3 shows the output of the JavaScript_Hand_Tracking code sample written using JavaScript and rendered using the HTML5 canvas.

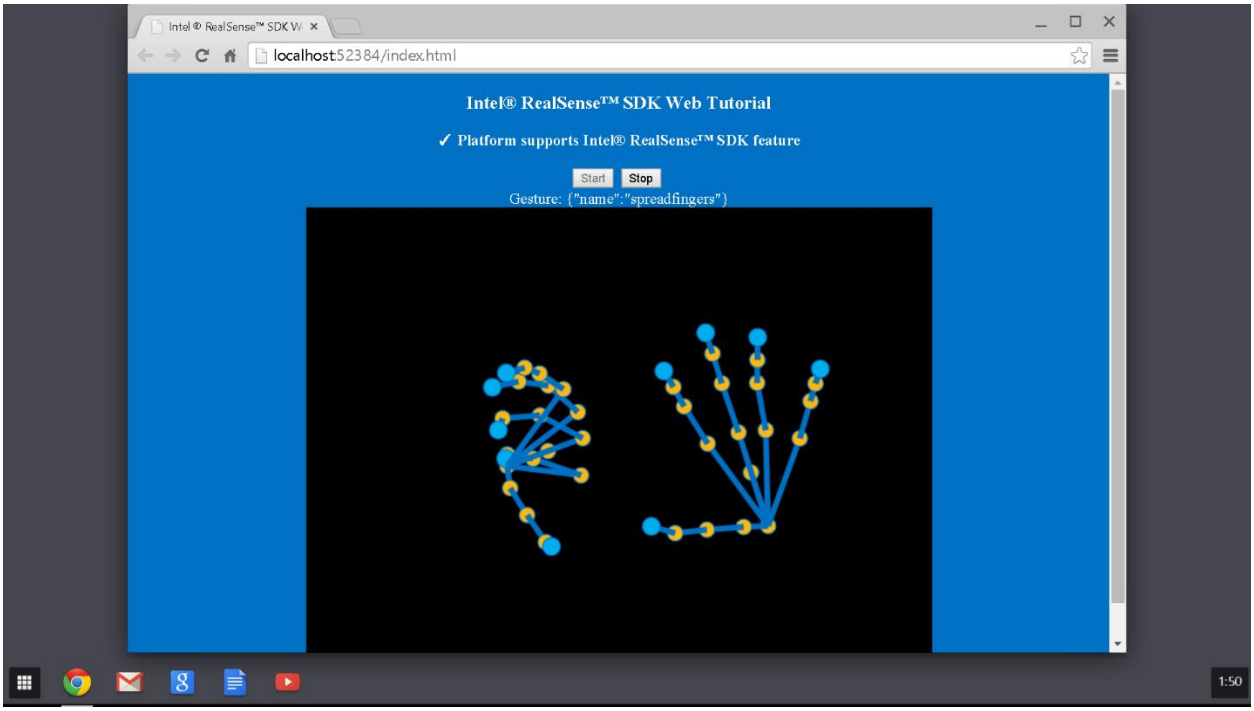

Figure 3. Hand Tracking using JavaScript* in an HTML5 web page.

To learn more

- For more information, read the Browser Support section in the SDK Reference Manual.