Slyce Web SDK

Version 1.0.1

Last updated: Feb 19, 2017

Slyce

Table of contents

GETTING STARTED 3

Overview 3

Prerequisites 4

Methods 4

init(clientId) 4

recognitionByUrl(url, callbackProgress, useVisualProgress = true) 5

recognitionByFile(file, canvas, callbackProgress, useVisualProgress = true) 6

similar(key, useVisualProgress = true) 7

progressClose() 7

GETTING STARTED

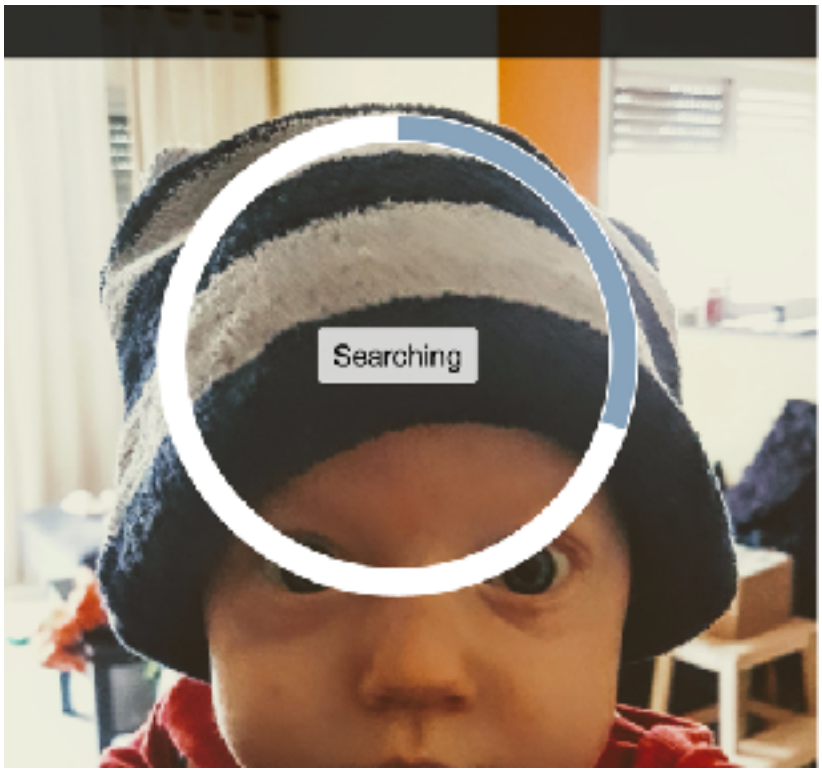
Overview

The Slyce Web SDK works with all mobile native browsers on both iOS and Android devices.

The Web SDK enables developer to use Slyce solutions in Web based applications for translating images into descriptive product information. It supports any approach a developer would decide to take - snapping photos from the mobile device camera, or opening image from the photo gallery.

After an image is snapped/selected, the SDK performs image detection while showing progress indication. At the end of the progress, results are delivered to developer in a structured form. SDK does also support delivering progress results while identifying the image. It allows creating of flexible and interactive experience for any development need.

In order to provide the best flexibility as well being able to integrate seamlessly in any existing e-commerce system, Web SDK imbeds almost no GUI. It is up to developer to decide what UI to use and on what Web platforms to support.



Prerequisites

- A Slyce client ID

Setup

Copy/paste the following HTML code into the page you created:

```
<script type="text/javascript" src="../dist/slyce.sdk.js"></script>
<link rel="stylesheet" href="../dist/slyce.ui.css">
```

Add this code somewhere in you application:

```
var sdk = new window.slyceSDK();
```

This will load the SDK code in your page context.
The SDK is now ready to operate.

If you use AMD or CommonJS2, the following is the recommended code for you to use:

```
var slyceSDK = require('../dist/slyce.sdk');
var sdk = new slyceSDK();
```

Methods

init(clientId)

Calls Slyce backend system and provides the client ID. The ID is validated and the connection with Slyce recognition system is set.

Params:

clientId - the client ID you have received from your Slyce representative.

Response:

The method returns native promise object.

In case of success - an object with result information is returned. The information includes allowed services turned on by the system.

An example response:

```
{ recognition : true, similar: false }
```

In cases of rejecting the request - An error message is returned.

Code sample:

```
var id = 'superidididid';
sdk.init(id).then(function(data) {
    console.log(data);  // { recognition : true, similar: true }
}, function(error) {
    console.log(error); // Error('Incorrect client id')
});
```

recognitionByUrl(url, callbackProgress, useVisualProgress = true)

The method allows sending a picture to the Slyce services for recognition. The picture should be hosted on the Web and the access has to be opened to public. The file will be automatically downloaded by the backend system and analyzed for results.

Params:

url - URL of the image you want to recognize
callbackProgress - function to be called called when status changes
useVisualProgress - the parameter indicates if a visual progress bar has to be displayed. If set to true, the progress will be displayed until results arrive. It will remain on the screen until progressClose() function is called.

Response:

Method returns native promise object.

In case of success - returns an array that includes product list.
In case of failure - an object with an error message is returned.

Code sample:

```
var url = 'http://imageurl.com/image.jpg';
sdk.recognitionByUrl(url,  function(data) {
    console.log(data);
   // progress event
   // {“code”:8000,"message":"Processing started”,”progress”:20,"token":"..."}
}).then(function(data) {
    console.log(data);
   // success
  //product list,[{“itemId”:”0024984171” … , “productPrice":"69.95", "productURL":""}]
}, function(data) {
    console.log(data);
   // Error('Image not found')
});
```

recognitionByFile(file, canvas, callbackProgress, useVisualProgress = true)

The method allows sending a picture to the Slyce services for recognition. The picture does not have to be hosted on the Web. The file will be automatically uploaded and analyzed by the backend system. An expected response is a set of products.

Params:

file - The file to be uploaded by user or developer (from camera or local storage).
canvas - a predefined canvas DOM element. It is used for rotating and resizing. In case operations are not required, the parameter should be null.
callbackProgress - function to be called called when status changes
useVisualProgress - the parameter indicates if a visual progress bar has to be displayed. If set to true, the progress will be displayed until results arrive. It will remain on the screen until progressClose() function is called.

Response:

Method returns native promise object.

In case of success - returns an array with products
In case of failure - an object with an error message is returned

Code sample:

```
$('input[type="file"]').on('change', function(e) {
    sdk.recognitionByFile(e.target.files[0], document.getElementById('canvas'),
function(data) { }).then(function(data) {}, function(data) {});
})
```

similar(key, useVisualProgress = true)

The method uses prebuilt image catalogs (provided by the client) in order to deliver related similar products.

Params:

key - URL of the image of the product that is requested to be searched for similar items.
useVisualProgress - the parameter indicates if a visual progress bar has to be displayed. If set to true, the progress will be displayed until results arrive. It will remain on the screen until progressClose() function is called.

Response:

Native promise object

In case of success - returns an array with products
In case of failure - an object with an error message is returned

Code sample:

```
sdk.similar(url).then(function(data) {
  console.log(data); // [{},{}]
}, function(data) {
  console.log(data); // Error('Products not found')
});
```

progressClose()

The method uses for closes visual progress overlay.

Params:

Receives no parameters

Response:

Returns nothing