JS Signature v1.0.2

by Brainy Data Limited

About JS Signature

Introduction

JS Signature is an Omnis java script component that provides signature capturing capabilities for your Omnis Studio java script client applications. The intended target implementation is touch-screen devices but JS Signature can also be used with mouse or touch pad driven devices.

Downloading the Software and Examples

If you have not done so already you can download the demo or full release software components from the following locations:

Demo Software and examples: https://demos.brainydata.com/download.htm

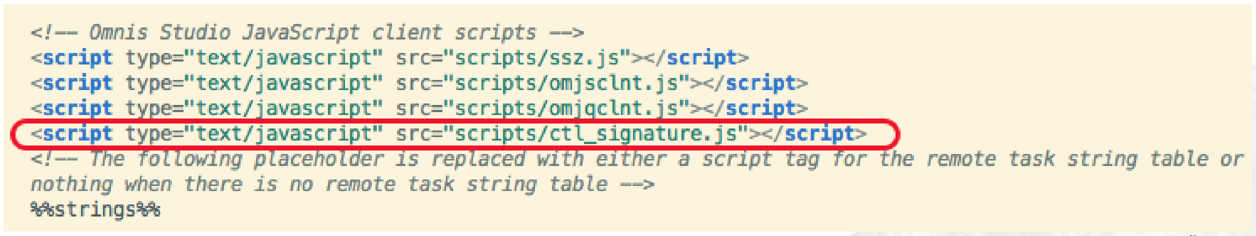
Release Software: https://support.brainydata.com/jssignature.htm

Installing the Software

JS Signature software requires Omnis Studio 6.0.2 or better.

Please refer to the readme.txt file in the downloaded folder for detailed instructions regarding installation. Here is a quick summary of what has to be installed prior to using the software

1. Copy the jsSignatureBD component DLL to the JSCOMP folder in your Omnis tree.
2. Copy the ctl_signature.js script file to the HTML/SCRIPTS folder in your Omnis' application support folder.
3. Modify the jsctempl.htm file in your Omnis' application support folder to include the circled line of code in figure 1 below.
4. If using Studio 6.0.2, replace the Omnis JS script files in your Omnis' application support folder with the provided 6.0.2 patch scripts.



```
<!-- Omnis Studio JavaScript client scripts -->
<script type="text/javascript" src="scripts/ssz.js"></script>
<script type="text/javascript" src="scripts/omjsclnt.js"></script>
<script type="text/javascript" src="scripts/omjqclnt.js"></script>
<script type="text/javascript" src="scripts/ctl_signature.js"></script>
<!-- The following placeholder is replaced with either a script tag for the remote task string table or
nothing when there is no remote task string table -->
%%strings%%
```

figure 1

Deploying your software

Please refer to the license agreement for rules on deployment.

Documentation

This documentation describes the functionality provided by JS Signature only. You may also wish to read the documentation relating to PDFDevice as this component is used in the JS Signature example library.

As a minimum we recommend that you read the following Chapters:

Introduction and Designing JS Signature.

Introduction

Overview

The JS Signature software consists of the following

i. The example classes in the "jssignature.lbs" library.
ii. The external component library jsSignatureBD for the Omnis design IDE
iii. The web client script ctl_signature.js
iv. A modified jsctempl.htm which includes the ctl_signature.js file.

For an introduction on how to integrate the JS Signature software into your library, please read the chapter Designing JS Signature.

Examples

There are a number of classes that demonstrate the use of the JS Signature software, however, the remote form class rfSignature is designed to demonstrate all the important features of the JS Signature control. The example also requires the Brainy Data PDFDevice external component which should have been included in the JS Signature demo software. PDFDevice is not a requirement to use JS Signature. It merely helps to demonstrate one of its uses.

The example library does not require installation and can be opened from anywhere. For a description of its classes see the Examples Reference.

External component Library

The external component library provides the design interface for the control's properties, methods and events. These are described in more detail in the chapter Designing JS Signature.

The java script

The java script file ctl_signature.js implements the client functionality for the control. It has been minified using the google closure compiler to reduce the size of the script and thus may not be very readable. This file should be installed in accordance with the instructions in the Welcome chapter.

The modified template file

The HTML file jsctempl.htm is based on the original Studio 6.0.2 jsctempl.htm with the addition that it includes the ctl_signature.js java script file. You should modify your ctl_signature.js file in your Omnis tree, or replace it with the provided file. This has to be done so that the control's java script is loaded during client execution.

Designing JS Signature

Introduction

This chapter gives a brief description of what is involved to add the signature control to your java script client application. For a more detailed description of the example classes and a complete list of the external component properties and methods please refer to the chapters Examples Reference and External Component Reference.

Designing the appearance

The JS Client component implements external DLLs for the design interface. This allows you to pick the JS client control from the component store, place it on your remote form and start designing the properties. The control's properties allow you to design the style and color of the signature reproduction and the background and border appearance.

Choosing the signature appearance

The JS Signature control is designed to capture pen strokes and from these pen strokes reproduce a fluid signature. Recording pen strokes involves the recording of pen placements and movements, which are essentially device mouse or touch events. Simply reproducing lines based on these events does not always result in a satisfactory reproduction of a signature. Thus JS Signature implements special smoothing algorithms as well as thickening and thinning algorithms for the individual strokes of a signature. To offer some control over these algorithms, JS Signature provides the properties $sigcolor, $sigpenstartsharpness, $sigpenendsharpness and $sigpenthickness.

Other appearances

The control further provides properties that allow some control over the appearance and placement of the clear icon (the clear icon allows a signee to clear a signature and start again) and the prompt text (text such as "Sign here" can be displayed inside the signature box).

The properties $cleariconmargin and $cleariconsize control the clear icon, the standard Omnis font properties together with the properties $textcolor, $textcolorwhendrawing, $textprompt, $textx and $texty control the prompt.

Handling the events

As the signee signs within the signature box, the control generates the event evSignatureChanged every time the signee lifts the pen or stylus. It is not advisable that the signature is committed to the server at this point as the signature may consist of multiple strokes. It is recomended that a seperate button is placed in the form which can be clicked to submit the signature to the server once the signee is satisfied. The evSignatureChanged event can be used to enable this button. This description implies that the $event method which handles this event should be implemented to be executed on the client as there is no need for server involvement.

This recommendation also applies for the evSignatureCleared event which is generated when the signee clicks the clear icon. In this case, the event can be used to disable the submit button.

Handling the data

The signature control provides a $dataname property. This property should be assigned with the name of an instance variable of type binary. When the server is called, any signature data within the control would have been assigned to this variable. THe signature data will be a URL with an imbedded asset which is base 64 encoded image data and a mime type. The static method $decodesignature can be used to convert this data to raw PNG for use with Omnis.

Examples Reference

Introduction

This reference serves as a guide to the classes that are provided by the JS Signature example. It gives a brief description of each class. More information can be found in the classes' $desc property and the many comments in the classes' code.

The example library implements two facets. The capturing and converting of signatures to a PNG image, and the combining of the image with a document for display on the client.

Please note: The example requires the use of the Brainy Data PDFDevice product which is provided as part of the demo software. It is not a requirement to license this additional software in order to implement the JS Signature control for the purpose of signature capturing.

Example Classes

rfSignature

This remote form provides the interface for capturing the signature and displaying a sample document with the signature superimposed.

When the form receives a submit request, typically after a signature was provided on the client, the method "handleSubmit" converts the signature data to a PNG image and produces a PDF document containing the signature below some sample text. This document is stored in the instance and the generic HTML control is assigned an embed tag that will cause an ultra-thin client request for fetching the document via the rtGetPdf task.

rtSignature

The design task for the remote form. It does not implement any functionality and is purely there as it is required by Omnis.

rtGetPdf

This remote task is called when the HTML control on page two of the rfSignature form is assigned an URL, that executes an ultra-thin request to return a PDF document in mime format suitable for display in an embed tag. This task simply returns the already produced PDF from the rfSignature server based class instance.

rPdf

This report class is used to print the sample text and signature image to a memory based PDF file.

External Component Reference

Introduction

This chapter lists all the JS Signature control properties, static methods, JS client methods and events.

Contents

Static Methods - Static methods are not associated with the control and can be used anywhere in an Omnis server method. They cannot be used in methods that are executed on the client. You can select static methods from the Functions tab in the Omnis Catalogue.

Object Properties - The control implements a number of properties that allow some control over the signature reproduction as well as user prompts and appearance changes during signature capture. These properties can only be assigned during design mode as the JS Signature control does not implement any client side property handling. It was deemed unnecessary as the design properties offer most of the required functionality for signature capturing, and thus substantially reduced the size of the java script.

The control may also display properties that are standard properties for JS controls. This documentation will only document standard properties if they require further explanation in relation to their use with the JS Signature control.

Properties shown in red are read-only and cannot be assigned.

Object Methods - Object methods are associated with the client control and can only be called from methods that are executed on the client.

Object Events - Events are usually generated as a result of user actions. It is generally recommended that $event methods that receive these events are executed on the client and if any server interaction is required, specific server methods can be called from the client method.

The control may also display events that are standard events for JS controls. This documentation will only document standard events if they require further explanation in relation to their use with the JS Signature control.

Static Methods

Static methods are not associated with the control and can be used anywhere in an Omnis server method. They cannot be used in methods that are executed on the client. You can select static methods from the Functions tab in the Omnis Catalogue.

Note: When picking static methods from the catalogue, Omnis appears to prefix static methods implemented by JS client libraries with “RF:”. This prefix must be removed as the method will not function with this prefix in place. We don’t know why Omnis prefixes some methods in this way.

$decodesignature()

Syntax: jsSignatureBD.$decodesignature(&xData)

Version/Platform: v1.0

This method converts the base 64 encoded URL asset to raw PNG format which can be used with Omnis picture controls.

Parameter	Description
xData	This parameter should by of type binary and must contain the valid URL asset. On return, xData will contain the raw PNG image data suitable for use with Omnis.
returns	return kTrue if xData contained a valid asset which was successfully converted to PNG. Returns kFalse otherwise

Object Properties

The control implements a number of properties that allow some control over the signature reproduction as well as user prompts and appearance changes during signature capture. These properties can only be assigned during design mode as the JS Signature control does not implement any client side property handling. It was deemed unnecessary as the design properties offer most of the required functionality for signature capturing, and thus substantially reduced the size of the java script.

The control may also display properties that are standard properties for JS controls. This documentation will only document standard properties if they require further explanation in relation to their use with the JS Signature control.

Properties shown in red are read-only and cannot be assigned.

Name	Type	Description
$cleariconmargin	Integer	Specifies the margin in pixels between the clear icon and the controls border.
$cleariconsize	Integer	Specifies the diameter in pixels for the clear icon.
$dataname	standard property	Specifies a binary variable that is to receive the signature data from the client. When data is received by the server, it is in the format of a HTML 5 URL asset that contains base 64 encoded image data and mime format information. Before it can be used as an image, you must call the static method $decodesignature. This method will convert the data inside var_name to raw PNG.
$nocanvassupport	Character	Specifies the text that is displayed when the browser does not support the canvas control.
$sigcolor	Color	Specifies the color of the signature.
$sigpenendsharpness	Integer	Specifies the number of vertices between the full pen thickness and the point at which the stroke ends. Vertices are time impacted thus a faster stroke will result in a longer path from full thickness to end.
$sigpenstartsharpness	Integer	Specifies the number of vertices between the start of the stroke and the point at which the stroke reaches full pen thickness. Vertices are time impacted thus a faster stroke will result in a longer path to full thickness.

$sigpenthickness	Integer	The pen thickness in pixels (canvas standard pixel measurement).
$textcolor	standard property	The text color property is used for the text prompt that is displayed inside the signature box. See also $textprompt and $textx and $texty.
$textcolorwhendrawing	Color	The text color used for the text prompt when the user has started signing within the box. See also $textcolor.
$textprompt	Character	The prompt displayed inside the signature box. See also $textx and $texty.
$textx	Integer	The horizontal position of the text prompt. See also $textprompt.
$texty	Integer	The vertical position of the text prompt. See also $textprompt.

Client Methods

Object methods are associated with the client control and can only be called from methods that are executed on the client.

$clearsignature()

Syntax: $cinst.$objs.Signature.$clearsignature()

Version/Platform: v1.0

This methods clears the signature box and prepares it to receive a new signature.

Parameter	Description
returns	return kTrue.

Object Events

Events are usually generated as a result of user actions. It is generally recommended that $event methods that receive these events are executed on the client and if any server interaction is required, specific server methods can be called from the client method.

The control may also display events that are standard events for JS controls. This documentation will only document standard events if they require further explanation in relation to their use with the JS Signature control.

evSignatureChanged

This event is generated when the user lifts the pen, after having started to sign. Prior to the control generating the event, the signature data so far is committed to the instance variable in the client form instance.

Please be aware that this does not mean the signature is complete. Many signatures involve multiple strokes where the pen is repeatedly place and lifted. Thus it is not advisable to commit the signature to the server when this event is received. A better strategy would be to place a button control on the form that allows the operator to manually submit the signature to the server.

Thus, the $event method that implements this event must be executed on the client.

Event Parameter	Description
no event parameters	

evSignatureCleared

This event is generated when the user clicks the clear icon, or an Omnis method calls $clearsignature().

This event indicates that the signature box is ready to receive a new signature. Prior to the control generating the event, the instance variable associated with this control would have been cleared by the control.

Event Parameter	Description
no event parameters	