INFO: Client-Side Scripting in ASP.NET (Legacy Web Controls)

Legacy Controls NOTICE

This article references our legacy Web Forms Web Viewing controls (WebImageViewer, WebAnnotationViewer, WebThumbnailViewer). It is preserved for archival purposes, but support strongly recommends using our modern HTML5 web controls: WebDocumentViewer, WebDocumentThumbnailer instead.

INFO: WebDocumentViewer Whitepaper - Getting Started With Web Viewing

Main Article Content

The DotImage WebImageViewer, WebThumbnailViewer, and WebAnnotationViewer server controls support client side scripting for manipulation without post backs. You can access nearly all properties and methods of the control via client side JavaScript.

JavaScript API Quick Reference

- **WebImageViewer**
- **WebThumbnailViewer**
- **WebAnnotationViewer**
- **ClientTools**
- **Enums**
- **Web Configuration Parameters**

Basic Syntax

To make the transition from a server oriented control to a client oriented control easier, the client side versions of the WebImageViewer, WebThumbnailViewer, and WebAnnotationViewer controls use almost identical syntax to the server side code that you would need to accomplish similar tasks on the server side.

Public Functions

All public constructors and global variables that are included within

- WebImageViewer.js
- WebThumbnailViewer.js
- WebAnnotationViewer.js
- ClientTools.js
- Enums.js

are prefixed with the word atalaThis makes it unlikely that any another components used on the same page will have a similar function names.

Note: This is provided as general information, and **does not** mean that every function in the API reference requires the atala prefix, the syntax provided in the API reference should be used exactly as shown.

OnPageLoad and using atalaInitClientScript

The WebImageViewer, WebThumbnailViewer, and WebAnnotationViewer use the window.onload event to initialize the client side components of the control. This is necessary because as the page loads, elements can move depending on how long images or objects take to load, and whether or not the size attributes of those objects are defined.

You can use atalaInitClientScript to run code in the OnLoad event. This function queues up each string sent to it and tries to execute it as JavaScript code. This allows you to execute multiple scripts OnLoad This function is located in ClientTools.js and is required by all of the DotImage WebControls.

Here is a general example that pops up an alert when the page is done loading.

Example

```
!-- You will need a reference to ClientTools.js before this snippet. -- This reference is automatically added to the page inline with the WebControls, -- so placing this snippet below one of these controls will be sufficient. --> <script language="javascript" type="text/javascript"> atalaInitClientScript("OnPageLoad()"); function OnPageLoad() { alert("Page is finished loading."); } </script>
```

Objects

INFO: Client-Side Scripting in ASP.NET (Legacy Web Controls)

Every WebImageViewer, WebThumbnailViewer, and WebAnnotationViewer control on the page is accessed by the ClientID generated by ASP.NET. Therefore WebImageViewer1 on your WebForm can be accessed in JavaScript using the same name.

Placing the WebImageViewer, WebThumbnailViewer, and WebAnnotationViewer controls inside a container control such as a ContentPlaceHolder or a UserControl, changes the ClientID. You may need to take this into account when accessing methods on the client side.

The example below shows how you can get the ClientID inline on an aspx page.

Example

```
asp:content ID="Content1" contentplaceholderid="_mainContent" runat="server">
<cc1:WebImageViewer ID="WebImageViewer1" runat="server" Width="500px" Height="460px" />
<script type="text/javascript"> var myViewer; atalaInitClientScript("OnPageLoad()"); function
OnPageLoad() { myViewer = <%= WebImageViewer1.ClientID %>; myViewer.RemoteInvoked =
Invalidate; } // This function forces the WebImageViewer to update all visible tiles, as //
it may not know that the image has changed on the server side. function Invalidate() {
myViewer.Update(); } // This function calls a RemoteInvokable function on the server side,
and // passes in an integer indicating which page to open. function GoToPage(n) { var vals =
new Array(); vals.push(n); myViewer.RemoteInvoke('GoToPage', vals); } </script>
</asp:content>
```

Properties

All properties for a given object are named the same as they are in the server code (unless otherwise noted on the table below). They are prefixed with the words `get` and `set`, and require parentheses to get the return values.

The following example sets mySize to an atalaSize object representing the size of WebImageViewer1 selection and sets the width of WebImageViewer2.

Example

```
ar mySize = WebImageViewer1.getSelection().getSize(); WebImageViewer2.setWidth('400px');
```

Events

All the event handlers for the WebControls can be used by creating your own function and setting the event to that function, or by using atalaEventAdd. These examples pop up an alert box every time the ScrollPosition is changed. All of the methods shown below are valid.

The ZoomChanged event handler on the WebAnnotationViewer cannot be set without breaking the automatic annotation zooming. Method 3 is the only method that works in this situation.

Method 1

```
/ First Method atalaInitClientScript("OnPageLoad()"); function OnPageLoad() {
WebImageViewer1.ScrollPositionChanged = myPositionChanged; } function myPositionChanged() {
alert("ScrollPosition Changed!"); }
```

Method 2

```
/ Second Method atalaInitClientScript("OnPageLoad()"); function OnPageLoad() {
WebImageViewer1.ScrollPositionChanged = function(){ alert("ScrollPosition Changed!"); }; }
```

Method 3

```
/ Third Method // This method allows you to add multiple handlers to the same event
atalaInitClientScript("OnPageLoad()"); function OnPageLoad() { atalaEventAdd(this,
WebImageViewer1, 'ScrollPositionChanged', myPositionChanged); atalaEventAdd(this,
WebImageViewer1, 'ScrollPositionChanged', myUpdate); } function myPositionChanged() {
alert("ScrollPosition Changed!"); } function myUpdate() { alert("myUpdate!"); }
```

See Also

HOWTO: Remotely Invoke ASP.NET Page Methods (Legacy Web Controls)

Original Article

Q10354 - INFO: ClientSide Scripting in ASP.NET (Legacy Web Controls)

Last Update

INFO: Client-Side Scripting in ASP.NET (Legacy Web Controls)

2026-11-06 - TD - (Redid all code formatting and fixed legacy article links)

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50133/INFO-ClientSide-Scripting-in-ASPNET-...