ANJAM User Guide

ANJAM User Guide

The Advertiser to Native JavaScript API for Mobile, or 'ANJAM', gives mobile creatives access to additional features that are not provided by MRAID. This includes the ability to perform deep linking.

Note that mobile creatives running in iframes can use MRAID alongside this API.

On This Page

Usage

All ANJAM functions are asynchronous. Any function that queries for a result will need to install a function to handle the response. The callback function will be called with a result object whose properties contain the results of the function call.

Note that all result objects will also contain a `result.caller` property with the value of the calling function's name (for example: `DeepLink` or `GetDeviceID`).

Using ANJAM/MRAID from within an iframe
In order for a creative to use ANJAM, MRAID from within an iFrame in the SDK (including when "Serve in iFrame" is checked), the creative needs to add this HTML snippet at the beginning:

```
<script src="https://acdn.adnxs.com/mobile/anjam/anjam.js"></script>
```

Exported Functions

This section lists all of the functions exported by ANJAM. Each function is described and used in an example.

`anjam.MayDeepLink(url, result_callback)`

Creatives may use this function to determine whether the app they are deep-linking to is installed on the device.

- `url`: A string indicating a deep link URL that is handled by a native app. For example, `"fb://post?1234"`. See also the Facebook documentation on deep linking.
- `result_callback` - `function(result) {}`: Will be called with a result object, with a value of `true` or `false` in the `result.mayDeepLink` property indicating whether the deep link will be handled on this device or not.

Example:

```
var url = "example://widget?id=123";

ExampleDeepLink = function (url) {
    // Test for success.
    anjam.MayDeepLink(url, function (result) {
    // This will log "Callback for MayDeepLink: true".
        console.log("Callback for " + result.caller + ": " + result.mayDeepLink);
        if (result.mayDeepLink) {
            anjam.DeepLink(url, function (deeplinkresult) {
             console.log(deeplinkresult.caller + " call failed.");
            });
        }
    });
}

ExampleDeepLink(url);
```

anjam.DeepLink(url, err_callback)

Creatives will use this function to call the deep link URL, usually as a response to an `onClick()` event and after receiving a `true` result from `mayDeepLink()`.

If the `DeepLink()` launch was successful and the ad was expanded, the ad will close and the callback will not be called.

- `url`: A string indicating a deep link URL.
- `err_callback`: `function(result) {}` - Called only if the `Deeplink()` operation was unsuccessfully launched. The result object is only useful for the `result.caller` property.

See the documentation for `anjam.MayDeepLink()` for an example.

anjam.ExternalBrowser(url)

Allows the creative to override the SDK configuration and force the landing page to open in the device's native browser (as opposed to the default in-app browser).

- `url`: A URL to open in the native platform browser. Similar to the `DeepLink()` call, but meant for launching web sites in the native browser.

Example:

```
var url = "http://www.example.com";

ExampleExternalBrowser = function(url) {
    anjam.ExternalBrowser(url);
}

ExampleExternalBrowser(url);
```

anjam.InternalBrowser(url)

URLs are normally opened within the in-app browser in an invisible mode to check whether the URL redirects to a store or not. This call forces the in-app browser to become immediately visible. If the URL or one of its redirects goes to the app store, the app store will be opened and the in-app browser dismissed as normal. This method allows overriding the initial invisible mode.

- `url`: A URL to open within the in-app browser.

Example:

```
var url = "http://www.example.com";

ExampleInternalBrowser = function(url) {
    anjam.InternalBrowser(url);
}

ExampleInternalBrowser(url);
```

anjam.RecordEvent(url)

Used for tracking events; the URL will be loaded by the SDK in the background. This is usually used for tracking pixels.

- `url`: The URL the SDK will load in the background. The URL will accept `'data'` URLs or `'javascript:'`.

Example:

```
ExampleRecordEvent = function() {
    anjam.RecordEvent("http://www.example.com/pixel?id=123");
}
```

anjam.DispatchAppEvent(event, data)

Gives the creative the ability to send a custom event to the app. To receive this event, the app must implement code.

On iOS, the application would implement `AppEventDelegate`'s `didReceiveAppEvent` method.

On Android, the application would implement the `onAppEvent` in the `AppEventListener` class.

- `event`: A string delivered to the app's handler.
- `data`: A string delivered to the app's handler.

Example:

```
var e = 'SomeEvent';
var d = 'TheEventData';

ExampleDispatchAppEvent = function(evt, data) {
    anjam.DispatchAppEvent(evt, data);
}

ExampleDispatchAppEvent(e, d);
```

anjam.GetVersion(callback)

Used to get the current version of the ANJAM API.

- `callback`: `function(result) {}` - Called with the current API version string ("1.0") in the `result.version` property.

Example:

```
ExampleGetVersion = function() {
    anjam.GetVersion(function (result) {
        // This will log "GetVersion returned: 1.0".
        console.log(result.caller + " returned: " + result.version);
    });
}

ExampleGetVersion();
```

anjam.GetDeviceID(callback)

Used to retrieve the current device ID for Advertising.

- `callback`: `function(result) {}` - Called with the `result.idname` and `result.id` properties. `idname` is a string with the type of ID being passed. `id` is the string value of the advertising ID.
- For iOS devices `idname` will be `"idfa"` and ID is the Apple ID For Advertising.
- For Android devices using the Google Play services, `idname` will be `"aaid"` and the `id` will be the Google Android Advertising ID.
- For Android devices not using Google Play services, `idname` will be `"sha1udid"` and the `id` will be the SHA1 hash of the device's Android ID.

Example:

```
ExampleGetDeviceID = function() {
    anjam.GetDeviceID(function (result) {
        // This will log "GetDeviceID: idfa = AEBE52E7-03EE-455A-B3C4-E57283966239".
        console.log(result.caller + ": " + result.idname + " = " + result.id);
    });
}

ExampleGetDeviceID();
```

anjam.SetMRAIDRefreshFrequency(milliseconds)

Provides the ability for the creative developer to specify a custom time interval for refreshing the values for size, position, and viewability. In addition, on iOS this call should enable `NSRunLoopCommonModes` on the `NSTimer`.

- `milliseconds`: Integer value describing the time interval in milliseconds.

Example:

```
<script src="mraid.js"></script>
<script type="text/javascript">
if (mraid.getState() == 'loading') {
    mraid.addEventListener('ready', onSDKReady);
} else {
    onSDKReady();
}

function onSDKReady() {
 if ((typeof window.anjam.SetMRAIDRefreshFrequency) !== "undefined") {
  anjam.SetMRAIDRefreshFrequency(10);
 }
 setInterval(function() {
  p = mraid.getCurrentPosition(), s = mraid.getScreenSize(), d =
document.getElementById('pc'), y = (1-(s.height-p.y)/s.height)*300;
  d && (d.scrollTop = y);
 }, 10);
}
</script>

<style type="text/css">
#pc { position: relative; width: 300px; height: 300px; overflow: hidden; }
</style>

<div id="pc">
 <a href="https://www.reddit.com/r/cats/"><img src="http://placekitten.com/300/600"
width="300" height="600"></img></a>
</div>
```

Related Topics

- Mobile SDKs
- Integrate with iOS
- Integrate with Android