|  |
| --- |
| Microsoft Corporation |
| RIN JavaScript Player |
| Developer Documentation v1.0 |
| Microsoft Research  **Abstract** |
| This document describes how to incorporate the JavaScript RIN player and plugins into a website or other environment capable of rendering HTML5 and JavaScript. |

Table of Contents:

[1 Introduction 2](#_Toc358972419)

[2 Getting Started 2](#_Toc358972420)

[2.1 Downloading the RIN JS Library 2](#_Toc358972421)

[2.2 Using RIN Player 2](#_Toc358972422)

[2.3 Incorporating the RIN Player into your HTML page 3](#_Toc358972423)

[2.4 Incorporating the Custom Experience Providers into your HTML page 3](#_Toc358972424)

[3 Instantiating the RIN player control 3](#_Toc358972425)

[3.1 Adding the player control declaratively 3](#_Toc358972426)

[3.2 Creating the player control programmatically 4](#_Toc358972427)

[3.3 Programmatically controlling the RIN player 5](#_Toc358972428)

[4 Setting up Deep-referencing 6](#_Toc358972429)

[5 Appendix 6](#_Toc358972430)

[5.1 RIN-Core Dependencies 6](#_Toc358972431)

[5.2 RIN-Experiences Dependencies 6](#_Toc358972432)

# Introduction

Rin.js is a JavaScript library that plays Rich Interactive Narratives (RIN). Please see the MSR Technical Report [MSR-TR-2012-78](http://research.microsoft.com/apps/pubs/default.aspx?id=170526) for an introduction to RIN.

RIN content may be presented to the RIN player either as URLs to pre-existing JSON files or as JSON content given at runtime. This documentation describes how to use RIN JS library to play RIN files.

# Getting Started

## Downloading the RIN JS Library

RIN JS consists of “rin-core-1.0.js” file containing the core player,, “rin-experiences-1.0.js” file containing many of the experience streams and their supporting frameworks like jquery, knockout etc., stylesheet file, template html file, sample html pages and sample narratives. <download link etc will come here later>

## Using RIN Player

RIN JS Player requires a server for execution and storage of the supporting files. The following are the steps to setup a server on your machine.

1. Install IIS, Apache or equivalent server software on the machine.
2. Create a site in the installed server with the “web” directory as the root. Grant appropriate permissions for access.
3. Open the site's root (default document is index.html) in your browser

## Incorporating the RIN Player into your HTML page

Insert the following lines in header of a new html file or existing html file:

<!-- rin player -->

<script src="lib/rin-core-1.0.js" type="text/javascript"></script>

**Note**: If you are creating new html file, please make sure it is saved in UTF-8 format.

## Incorporating the Custom Experience Providers into your HTML page

Insert the following lines in header of a new html file or existing html file:

<!-- Optional Custom Libraries -->

<script type="text/javascript" src="http://ajax.aspnetcdn.com/ajax/knockout/knockout-2.1.0.js"></script>

<!-- Custom Experience Stream Scripts -->

<script type="text/javascript" src="lib/SampleImageES.js"></script>  
 **Note:** The custom Experience Providers need to be loaded after the rin-core.

# Instantiating the RIN player control

## Adding the player control declaratively

Here is simple example of adding a player control in html:

<body onload="rin.processAll()">

<div class="rinPlayer"

style="width:100%;height:100%;margin:0;padding:0;overflow: hidden;"

data-src="narratives/sample/narrative.js"></div>

</body>

In this example, there is a div in body with class name “rinPlayer”. The rin.processAll call in the onload event causes RIN to look at all contents of the document and embed RIN player when an element has class “rinPlayer”. If the data-src attribute points to a valid JSON URL, then that JSON RIN is played automatically.

To see this example working as sample, please run sdk-test.html under web folder in RIN sdk.

To add options to the player, add data-options attribute with options specified in query-string format. For example:

<div id="rinPlayer"

class="rinPlayer"

style="width: 100%;height: 100%;margin: 0;padding: 0;overflow: hidden;"

data-src="narratives/sample/narrative.js"

data-options="autoplay=false&loop=true"></div>

This example sets autoplay to false and loop to true. Now, the player will not start automatically and once started manually, it’ll loop forever. List of options available:

|  |  |  |
| --- | --- | --- |
| Name | Default | Comment |
| autoplay | false | Indicates if RIN should autoplay contents if valid source is specified. |
| controls | true | Indicates if default controls for play/pause/seek etc should be shown. If false, no UI control is shown for users to interact. The player can be manipulated programmatically. |
| loop | false | Indicates if the contents should loop back to start on ending |
| muted | false | Indicates if audio should be muted on startup. |

## Creating the player control programmatically

One way to create an instance of the RIN player control programmatically is to call rin.createPlayerControl(). This and other methods under the rin namespace to work with player instances are presented in the table below.

|  |  |
| --- | --- |
| Name | Comment |
| createPlayerControl(playerElement, options) | Creates a new player control and returns. PlayerElement is a pointer to a div element where the player will draw its contents. The options parameter is a list of options in query string or JSON format. |
| processAll(rootElement, systemRootUrl) | Looks for elements with class “rinPlayer” under rootElement (or Document if not specified) and creates new player controls for each instance found. systemRootUrl *(optional)* Identifies the path to the [RIN Dependencies](#_RIN_Dependencies) if the files are placed in a different location with respect to the html file |
| getPlayerControl(playerElement) | Returns the RIN player control associated with a given DOM element. |

The same set of player invocation options specified in previous section, represented as a JSON object may be specified as the second parameter to createPlayerControl. For example:

var p = rin.createPlayerControl(

playerElement,

{autoplay: true, controls: true, loop: true });

To see a sample on this approach, please run programmaticRinCreationSample.html under web folder in rin-sdk.

## Programmatically controlling the RIN player

The state of a RIN player control can be manipulated programmatically. Consider the following example:

<script type="text/javascript">  
//Note the $ here refers to jquery object which needs to be loaded explicitly in the page

$(document).ready(function () {

rin.processAll().then(function(){

var myrin = rin.getPlayerControl(document.getElementById('myrin'));

setTimeout(function () { myrin.play(); }, 1000); // play after 1 seconds

setTimeout(function () { myrin.pause(); }, 5000); // pause after 5 seconds

});

});

</script>

The code in the example executes when the HTML document is loaded. The code first calls rin.processAll to bind all RIN elements. It then looks up the first div element with ID “myrin” by an id and retrieves its associated RIN player control. Finally it invokes the players’ play and pause methods within the context of timer callback functions.

The RIN player control object has following public members:

|  |  |  |
| --- | --- | --- |
| Name | Type | Comment |
| play(offset, screenplayId)  play() | method | Causes RIN to play. The offset parameter specifies the offset in seconds at which pay begins. The screenplayId parameter specifies a RIN Screenplay ID to play. If called with no parameters, the play method starts at the current position (which defaults to the beginning), and plays the current screenplay (which defaults to defaultScreenplayId as specified in the RIN data). |
| pause(offset, screenplayId)  pause() | method | Causes RIN to pause. The parameters are same as in play method. |
| load(narrativeUrl, onComplete) | method | Loads the JSON RIN file located in the given narrativeUrl. onComplete callback is fired after load, if provided. |
| loadData(rinData, onComplete) | method | Loads JSON object passed as rinData. onComplete callback is fired after load, if provided. |
| getPlayerState() | method | Returns enumeration value of player state. The values under rin.contracts.PlayerState are : playing, pausedForBuffering, pausedForExplore, stopped, inTransition |
| getCurrentTimeOffset() | method | Returns current time offset in seconds. |

# Setting up Deep-referencing

To setup deep referencing (described in section 5.5 of the MSR Technical Report [MSR-TR-2012-78](http://research.microsoft.com/apps/pubs/default.aspx?id=170526)), the following steps need to be added in the main/startup page of RIN, once player control is ready for processing.

1. Resolve the url to player understandable version

var deepstateUrl = myrin.resolveDeepstateUrlFromAbsoluteUrl(window.location.href);

//myrin refers to the PlayerControl object

1. Load the player control with the new data

myrin.load(deepstateUrl);

Note: This needs to be setup for use of links from share controls which appear in the player

# Appendix

## RIN-Core Dependencies

Currently RIN Core and RIN Player are dependent on the following resources

1. lib/rin-experiences.1.0.js
2. lib/jquery-1.7.2.min.js
3. lib/knockout-2.1.0.js
4. lib/jquery.easing.1.3.js
5. systemResources/themeResources/images *folder*
6. systemResources/themeResources/rin.css
7. systemResources/themeResources/rinTemplates.htm

## RIN-Experiences Dependencies

The following resources are to be explicitly loaded based on the Experience Provider

1. Map Provider   
   <script src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0"/>
2. Deep Zoom Provider <script src="lib/seadragon-0.8.9-rin.js"></script>
3. Panorama Provider  
   <script type="text/javascript" src="lib/pano-viewer.js"></script>

<script src="lib/jquery.pxtouch.js" type="text/javascript"></script>

**Note**: These dependencies may change in future versions of the RIN along with other default experiences