HTML5 Audio & Video for jQuery

jPlayer 2.4.0 Developer Guide

Contents

Related Links

- Release Notes
- jPlayer Google Group (Ask for support and discuss issues here.)
- jPlayer on jQuery website
- jPlayer on GitHub
- HTML5 Media Event Inspector
- HTML5 <audio> and Audio() Support Tester

jPlayer Fundamentals

Essential jPlayer Media Formats

The media formats that are essential to jPlayer are those that are supported by both the Flash solution and the HTML5 browsers that do not support Flash, such as iOS. It is important that one of these formats is supplied to jPlayer so that popular browsers will be able to play your media. After an essential format has been supplied, additional counterpart formats may be supplied to increase cross-browser support of the HTML5 solution. See the `jPlayer({"supplied": "formats"})` option for more details.

Audio

One of the following audio formats must be supplied:

- MP3
- M4A

Note that, there is little to no benefit in providing both, due to the way support pans out with HTML5 browsers supporting either both or neither.

Video

The following video format must be supplied:

- M4V

Setting the size of jPlayer

Set the width and height of jPlayer using the `jPlayer({size:Object})` constructor option.

The full-screen size is set using the `jPlayer({sizeFull:Object})` constructor option.

Note that the background-color may be set using the constructor option `jPlayer({backgroundColor:"#RRGGBB"})`

Flash Security Rules

The security rules for jPlayer's SWF file have been relaxed using `System.security.allowDomain('*')` and may be called from any domain.

Generally, you will upload the SWF file with the JavaScript file to a directory called "js" on your domain. Use the constructor option `jPlayer({"swfPath":path})` to change the path.

Technically, the plugin files may be linked to remotely at jplayer.org. However, we request that you don't link to the files on jplayer.com, since we do not have sufficient resources to become a CDN just now. Additionally, the Flash fallback on the remote server would require the use of absolute URLs for all `jPlayer("setMedia", media)` URLs.

Attempting to run jPlayer locally on your computer will generate Flash security violations and you would need to enable the local file access using the Flash Settings Manager. See the Flash Player Help for more information.

To develop locally, install a server on your system, such as Apache, to enable a localhost on your computer.

Media Encoding

The media supplied must conform with the codecs supported by HTML5 browsers. The essential jPlayer formats (MP3 or M4A for audio and M4V for video) must also conform with the codecs supported by the Adobe Flash plugin.

jPlayer needs to distinguish between audio and video. This is because jPlayer behaves slightly differently for the two different media types. For example, the video needs to be shown. As a result, formats that are containers have an audio and video type included in their abbreviation to make them unique. For example, M4A and M4V are both MP4 files, and jPlayer knows to play them as audio and video respectively.

MP3

Since some browsers use the Flash element of jPlayer, the MP3 files used must be encoded according to the browser's Adobe Flash Plugin limitations:

- Constant Bitrate Encoded.
- Sample Rate a multiple of 11,025Hz. ie., 22,050Hz and 44,100Hz are valid sample rates.

M4A / M4V

The MP4 file is a container that supports both audio and video. The M4A format is an audio MP4, and the M4V format is a video MP4. The recommended encoding options for standard and mobile browsers are:

- H.264 Baseline Profile Level 3.0 video, up to 640 x 480 at 30 fps. Note that B frames are not supported in the Baseline profile.
- AAC-LC audio, up to 48 kHz.

The metadata should be at the start of the encoded data, since the Flash solution must have the metadata in order to begin playling. The option for this varies from encoder to encoder, but usually refers to optimizing for either web or streaming. This 3rd party link might help if you are having problems with the metadata: MetaData Mover

OGA / OGV

The OGG file is a container that supports both audio and video. The OGA format is an audio OGG, and the OGV format is a video OGG. The Vorbis and Theora codecs appear to have full support.

WEBMA / WEBMV

The WebM file is a container that supports both audio and video. The WEBMA format is an audio WebM, and the WEBMV format is a video WebM. The Vorbis and VP8 codecs appear to have full support.

WAV

The WAV format is supported by many HTML5 browsers. We recommend that you avoid it though as a counterpart format. The recommended encoding options are:

- 8-bit and 16-bit linear PCM
- Only codec "1" (PCM) is supported.

FLA / FLV

The Flash FLV file is a container that supports both audio and video. The FLA format is an audio FLV, and the FLV format is a video FLV. The FLA/FLV format is supported by the Flash fallback.

RTMPA / RTMPV

The RTMP protocol uses a Flash Media Server (FMS) to stream media to jPlayer's Flash solution. The RTMPA format is an audio RTMP stream, and the RTMPV format is a video RTMP stream. The HTML solution does not support RTMP.

Encoders:

- Audacity
- Miro
- Firefogg
- FFmpeg2theora
- Handbrake
- Vid.ly
- Archive.org

References:

- Safari: Creating Video
- Android Supported Media Formats
- Opera: Everything you need to know about HTML5 video and audio
- Firefox: Media formats supported by the audio and video elements

Server Response

MIME Type

Your domain's server must give the correct MIME type (content type) for all media URLs.

Failure to give the correct MIME type will stop the media from working on some HTML5 browsers. This is a common cause of problems that only affect Firefox and

Opera. Other browsers are less strict, but the MIME type should always be checked that it is correct if you having problems getting media to play on any browser.

Media MIME Types

- MP3: audio/mpeg
- MP4: audio/mp4 video/mp4
- OGG: audio/ogg video/ogg
- WebM: audio/webm video/webm
- WAV: audio/wav

If you use a common extension for both audio and video media, for example audio.mp4 and video.mp4, then simply use the video version of the MIME type for both of them. ie., video/mp4

On Apache servers, you can use the .htaccess file to set the MIME type based on the file extension:

```
1.  # AddType TYPE/SUBTYPE EXTENSION
2.
3.  AddType audio/mpeg mp3
4.  AddType audio/mp4 m4a
5.  AddType audio/ogg ogg
6.  AddType audio/ogg oga
7.  AddType audio/webm webma
8.  AddType audio/wav wav
9.
10. AddType video/mp4 mp4
11. AddType video/mp4 m4v
12. AddType video/ogg ogv
13. AddType video/webm webm
14. AddType video/webm webmv
```

Do Not GZIP the Media

Disable GZIP encoding of all the media files. Media files are already compressed and the GZIP will just waste CPU on your server.

The Adobe Flash Plugin will experience issues if you GZIP the media.

Do not GZIP the Jplayer.swf file either. Feel free to GZIP the JavaScript.

Byte-Range Requests

Your server must enable `Range` requests. This is easy to check for by seeing if your server's response includes the `Accept-Ranges` in its header. Most HTML5 browsers enable seeking to new file positions during a download, so the server must allow the new `Range` to be requested.

Failure to accept byte `Range` requests will cause problems on some HTML5 browsers. Often the duration cannot be read from the file as some formats require that the start and end of the file is read to know its duration. Chrome tends to be the browser that has most problems if the `Range` request is not enabled on the server, but all browsers will have some issue even if it is only that you have to wait for all the media to load before jumping close to the end.

This problem is known to affect Jetty 6 servers with their default configuration.

A PHP function has been written by the jPlayer community that can serve media files with support for Range requests. See this jPlayer Support Group Post on the topic.

Protecting Your Media

Be careful when trying to restrict access to your media files. The media URL must be accessible over the internet by the user and its response must be in the format expected.

Using the server response to disable the local cache of media can cause problems with some HTML5 browsers. This can cause the duration of the media to be unknown, which will show as a NaN in the duration of jPlayer.

If you do some magic on the backend to make it more secure, them make sure you are accepting the byte `Range` requests described above.

How to disable jPlayer with CSS or jQuery

Since jPlayer uses Flash on some browsers, the jPlayer must not be hidden. Mobile browsers do not appear to like their HTML5 media being hidden either.

You can change the size of the jPlayer to zero if you want to make it invisible. You should also be careful that any jQuery animations, such as fadeIn() or fadeOut(), are not acting on the jPlayer or any of its parents.

jPlayer will attempt to recover and reconfigure the Flash solution when shown again after being hidden. However, this assumes that you did not issue any commands to jPlayer while it was hidden.

For example, do **not** do the following:

```
1.   <head>
2.      <style>
3.         #jquery_jplayer {
4.            display:none; /* Disables jPlayer when using Flash */
5.         }
6.      </style>
7.      <script type="text/javascript">
8.        $("#jquery_jplayer").hide(); /* Disables jPlayer when using
   Flash */
```

```
    </script>
</head>

<body>
    <div id="jquery_jplayer"></div>
</body>
```

Security Restrictions

In order to eliminate cross site scripting (XSS), there are some limitations placed on the `{noConflict}` option and on the ID name of the jPlayer element. These restrictions are due to the communication between the Flash fallback and the JavaScript in the page, which require the jQuery variable name and the ID of the jPlayer element. This exposed a potential vulnerability through direct access to the Jplayer.swf file, which has been eliminated through the addition of these rules.

The whitelisted characters are: `A-Z a-z 0-9 _ - .` Or in other words, the alpha-numeric characters, underscore, hyphen/minus and period.

jPlayer Data

The jPlayer data is usually accessed via the event object, since the internal data tends to be updated before events are generated. An example of how a timeupdate event handler can be created through the constructor option is:

```
$('#jp').jPlayer({
    timeupdate: function(event) { // 4Hz
        // Restrict playback to first 60 seconds.
        if (event.jPlayer.status.currentTime > 60) {
            $(this).jPlayer('stop');
        }
    }
    // Then other options, such as: ready, swfPath, supplied and so on.
});
```

The jPlayer object itself may also be accessed through `$('#jp').data('jPlayer')`. However, since this gives full read/write access to all data and access to all the methods, even the internal methods, this is only recommended for advanced developers. Reading data is fine, but setting data might have unexpected side effects.

The previous timeupdate event code could be (crudely) emulated using an interval with the following code:

```
var jp = $('#jp'), jpData = jp.data('jPlayer');
setInterval(function() {
    // Restrict playback to first 60 seconds.
    if (jpData.status.currentTime > 60) {
        jp.jPlayer('stop');
    }
```

```
7.   },100); // 10Hz
```

See also: `jPlayer("option")`

jPlayer and Zepto

jPlayer has been tweaked so that you can now use Zepto instead of jQuery.

Zepto 1.0+ compiled with the data module is required.

If you want to use jPlayer as an AMD module for Zepto, then you will need to edit lines of code in the source. By default, the AMD module has its dependencies set to jQuery, not Zepto. The following is the top of the jPlayer code and you can clearly see the two lines that need to be switched.

```
1.   (function (root, factory) {
2.      if (typeof define === 'function' && define.amd) {
3.         // AMD. Register as an anonymous module.
4.         define(['jquery'], factory); // jQuery Switch
5.         // define(['zepto'], factory); // Zepto Switch
6.      } else {
7.         // Browser globals
8.         if(root.jQuery) { // Use jQuery if available
9.            factory(root.jQuery);
10.        } else { // Otherwise, use Zepto
11.           factory(root.Zepto);
12.        }
13.     }
14.  }(this, function ($, undefined) {
```

But to avoid presumption... These lines here:

```
1.         define(['jquery'], factory); // jQuery Switch
2.         // define(['zepto'], factory); // Zepto Switch
```

Change to:

```
1.         // define(['jquery'], factory); // jQuery Switch
2.         define(['zepto'], factory); // Zepto Switch
```

If you are not using the AMD module, then jPlayer is setup ready to use Zepto. Simply include Zepto in your page instead of jQuery.

Zepto support was officially released in jPlayer 2.4.0

jPlayer Files

jPlayer requires that two files are uploaded to your server.
`jquery.jplayer.min.js`
`Jplayer.swf`

When updating jPlayer, ensure that both files are uploaded from the plugin ZIP, since both files are subject to change.

Plugin Files

jquery.jplayer.min.js

47.7KB (Gzip: 11.9KB)

The plugin's JavaScript file, compiled using closure-compiler.appspot.com with "Simple" optimization.

The jquery.jplayer.min.js file should be added to the `<head>` of the HTML file after the jQuery JavaScript file. Below is how to include the JavaScript file using an absolute path, relative to the server root.

```
1.    <head>
2.      <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.1/jqu
   ery.min.js">
3.      </script>
4.      <script type="text/javascript"
   src="/jPlayer/js/jquery.jplayer.min.js">
5.      </script>
6.    </head>
```

Jplayer.swf

13.7KB

The jPlayer plugin's Flash 10 file, compiled using the Flex 4.5.1 open-source compiler. The most recent version is available here: Adobe Flex SDK.

This file is used in browsers that do not support HTML5 `<video>` and `<audio>`.

The Jplayer.swf file must be uploaded to your server. The default location is relative to your current page in a folder named "js". To modify the location of the Flash file, use the `swfPath` constructor option. Below is how to set the `swfPath` to a path relative to the server root. See the `jPlayer({swfPath})` constructor option for more information.

```
1.    $("#jpId").jPlayer( {
2.      swfPath: "/scripts/jPlayer"
3.    });
```

Source Files

jquery.jplayer.js

jPlayer's JavaScript source file. This file may be switched with the jquery.jplayer.min.js file. The operation is identical, but the file size is larger.

The JavaScript source is verified using JSHint and the options can be found in the source.

This file is readable by humans, assuming they understand JavaScript and jQuery.

Jplayer.fla

This file is no longer needed, as the Flex compiler is now being used to compile the ActionScript. It is left in the source as you can use it to compile the ActionScript if you choose.

jPlayer's Adobe Flash CS4 Professional editor file. This file can be compiled using Adobe Flash CS4 Professional into the Jplayer.swf file used in the jPlayer plugin. The Jplayer.fla file is not required for jPlayer, and does nothing unless compiled.

This file is empty, except for setting its class to Jplayer.as.

Jplayer.as

jPlayer's ActionScript 3 source file. The Jplayer.as file is not required for jPlayer, and does nothing unless compiled.

This file is readable by humans, assuming they understand Flash's ActionScript 3.

happyworm.jPlayer AS3 package

Contains jPlayer's ActionScript 3 source files for classes used in Jplayer.as. These files are not required for jPlayer, and do nothing unless compiled.

These files are readable by humans, assuming they understand Flash's ActionScript 3.

jPlayer Constructor

$(id).jPlayer(Object: options) : jQuery

Description

The jPlayer constructor is applied to the given jQuery selector (normally an ID css selector) and uses the options, if provided. The used by jPlayer should be empty and not used for anything else.

In some cases, the jPlayer div can be placed at the top level of the <body> to help avoid effects on the page corrupting jPlayer's operation. This applies to an audio player that does not use the poster feature of jPlayer.

Note that the most important option is the `ready` event handler, which is a function defining the actions to perform once jPlayer is ready for use. Attempting to issue commands to jPlayer before the `ready` event occurs will result in jPlayer raising error events if the Flash fallback is used and commands will be ignored.

The next most important option is the `swfPath` element, which defines the path of jPlayer's SWF file. Remember to upload the SWF file to your server!

Changing options after instancing jPlayer

Options that may be changed using the `jPlayer("option", key, value)` method after instancing jPlayer, are indicated by the † symbol.

Event handlers created through the jPlayer options

Event handlers that may be created through the jPlayer constructor options, are not really options. They are event handlers bound to the jPlayer element, with the ".jPlayer" jQuery event namespace, and are included in the options as an easy way to bind a handler to an event. Event handlers that are listed here in the options are indicated by the ‡ symbol.

Parameters

options

Object defining any changes to the default configuration of jPlayer.

ready ‡

> Function : (Default: undefined) : Defines an event handler function that is bound to the $.jPlayer.event.ready event. To reference the current instance, use `$(this)`. Generally, it is recommended to use a function here to at least `$(this).jPlayer("setMedia",media)` the instance to a valid url ready for use.
>
> The ready event handler is defined at creation to eliminate a race condition between the JavaScript code and Flash code. Thus ensuring the Flash function definitions exist when the JavaScript code is executed.
> The handler is bound with the `.jPlayer` namespace.

swfPath

> String : (Default: "js") : Defines the path of jPlayer's Jplayer.swf file. This allows the developer to place the SWF file wherever they choose by using an alternative relative path, absolute path or server root relative path.

The swfPath option may be a path or a URL to the SWF file with extension `.swf`. For example, `jPlayer({swfPath:"/scripts/banana.swf"})` where you renamed the Jplayer.swf file to "banana.swf". Do not rename the `.swf` extension, or a path will be assumed.

You can test that your swfPath is correct on an HTML5 browser that supports flash, by using the constructor option `jPlayer({solution:"flash, html").`

An incorrect swfPath will generate `$.jPlayer.event.error` events with error type `$.jPlayer.error.FLASH` when you issue commands like `jPlayer("setMedia", media)`

Independent of trailing slash, ie., "myPath/" is the same as "myPath".
The URL given must conform to standard URL Encoding Rules.

solution

String : (Default: "html, flash") : Defines the priority of the html and flash solutions. The default is to use HTML first, with a Flash fallback. Swapping the order to `"flash, html"` causes the Flash to be used first, with an HTML fallback.

Depending on the `supplied` formats, jPlayer may even use both solutions. For example, using jPlayer as a media player with `{supplied:"mp3,oga,m4v"}` would on browsers like Firefox require the Flash solution to play the video media format, while the audio media can be played in the HTML solution.

While you could specify only one solution, we do not recommend that you do it!

The solution picked by jPlayer depends on your brower and the formats in the supplied option.

supplied

String : (Default: "mp3") : Defines the formats supplied to jPlayer. The order defines the priority, with the left most format being highest. Formats to the right are lower priority.

See Media Encoding for more details on the formats. See Essential Formats for more details on why some formats are manditory.

Essential Audio formats: **mp3** or **m4a**.
Essential Video format: **m4v**.

Counterpart formats: webma, webmv, oga, ogv, wav, fla, flv, rtmpa, rtmpv.

All formats in the supplied option must be given in every `jPlayer("setMedia",media)` command.

The exception to this rule applies to a Media Player. ie., A player that plays both video and audio. Both the video and audio formats are defined in the supplied option. Then use either the video or audio media when using `jPlayer("setMedia",media)` command. (You must give all the audio formats or all the video formats defined in the supplied option.)

The `solution` priority dominates the priority of the `supplied` formats. jPlayer works by finding the first working format in a given solution. So in Chrome with `{solution:"html,flash"}` and `{supplied:"mp3,oga"}` the mp3 file would play in native HTML, even though Chrome could play either.

The format picked by jPlayer depends on your brower and the solution option. Essential formats are those supported by Flash and (luckily) by browsers that do not allow Flash. ie., iPad/iPhone
Counterpart formats are supplied to improve x-browser HTML5 media support.

size †

object : Sets the size of the restored screen mode. The defaults depend on whether audio or video formats are supplied. The video defaults are used when both media types are supplied.

Object Properties:

width
String : (Default: [Video:"480px"] [Audio:"0px"]) : The width as a CSS rule.
height
String : (Default: [Video:"270px"] [Audio:"0px"]) : The height as a CSS rule.
cssClass
String : (Default: [Video:"jp-video-270p"] [Audio:""]) : The class name added to the cssSelectorAncestor when in this mode.

sizeFull †

object : Sets the size of the full screen mode. The defaults depend on whether audio or video formats are supplied. The video defaults are used when both media types are supplied.

Object Properties:

width
String : (Default: [Video:"100%"] [Audio:"0px"]) : The width as a CSS rule.
height
String : (Default: [Video:"100%"] [Audio:"0px"]) : The height as a CSS rule.

cssClass
String : (Default: [Video:"jp-video-full"] [Audio:""]) : The class name added to the cssSelectorAncestor when in this mode.

smoothPlayBar †

Boolean : (Default: false) : Smooths the play bar transitions.

The play bar width changes are animated over 250ms to smooth the change, rather than a step change. This also affects clicks on the play bar, where the bar animates to the new position.

Short duration media benefits the most, since their step changes are the largest.

The 250ms animation period approximately matches the period of the timeupdate event for both the html and flash solutions.

fullScreen †

Boolean : (Default: false) : Sets the inital state of the full screen mode.

This option is normally changed through the full screen and restore screen buttons on the interface. The video will be displayed in true full screen mode if supported by the browser, otherwise the display will be full window.

This option affects the `jPlayer({cssSelectorAncestor})` element, so that the jPlayer display tracks the GUI display.

Changing the `fullScreen` option will change the `fullWindow` option to match it. The exception to this rule is when WebKit Video is being used on iOS, where the `fullWindow` option will not be changed.

The full screen mode uses the W3C Full Screen API, which means both the HTML and Flash solutions are displayed using this mechanism. This allows the GUI defined in the CSS to be used over the Flash while in full screen mode. Remember that the Flash has no GUI controls built into it, which is why it is not displayed full screen through the Adobe Flash plugin.

fullWindow †

Boolean : (Default: false) : Sets the inital state of the full window mode.

This option is a fallback for when full screen is not available and is set automatically when attempting to go full screen. In general, you will not use this option and instead use the `fullScreen` option to control how the video is displayed.

This option has been left here since you can use the full window system for other uses, where you do not want to activate the full screen code as well.

This option affects the `jPlayer({cssSelectorAncestor})` element, so that the jPlayer display tracks the GUI display.

Changing the `fullWindow` option will not affect the `fullScreen` option, but may affect the full screen GUI display, since the CSS classes would be changed. ie., If you were in full screen mode when you unset the fullWindow option.

audioFullScreen †

Boolean : (Default: false) : Enables keyboard controls to enter full screen with audio media.

This option allows key controls to display audio poster images in full screen, which is useful for media players. eg., A player that has both video and audio media in a playlist.

For more information on keyboard controls see the `jPlayer({keyEnabled})` option.

autohide †

object : Sets the auto-hide options for the GUI in the screen display states.

Object Properties:

restored
Boolean : (Default: false) : Auto-hide the GUI when in the restored screen display state.
full
Boolean : (Default: true) : Auto-hide the GUI when in the full screen display state.
fadeIn
Number : (Default: 200) : The period of the fadeIn animation in milliseconds.
fadeOut
Number : (Default: 600) : The period of the fadeOut animation in milliseconds.
hold
Number : (Default: 1000) : The period of the pause before autohide begins in milliseconds.

preload

String : (Default: "metadata") : Valid values are "none", "metadata" and "auto", which matches the HTML5 draft standard. Use "auto" to preload the file.

Preload is a hint to the user agent, not a command. Some browsers ignore this option.

volume †

Number : (Default: 0.8) : Defines the initial volume as a value from 0 to 1.

muted †

Boolean : (Default: false) : Defines the initial muted state.

verticalVolume

Boolean : (Default: false) : By default, clicks on the volume bar are calculated from the left. Setting this option to true, causes the calculation to be from the bottom.

If this option is used, then the following CSS styles are required on the volume bar elements to make them display vertically:

```
1.  <style>
2.  .jp-volume-bar {
3.      position:relative;
4.  }
5.  .jp-volume-bar-value {
6.      position:absolute;
7.      bottom: 0;
8.  }
9.  </style>
```

backgroundColor

String : (Default: "#000000") : Defines the background color of the jPlayer and the Flash. The string is an RGB hash of the form "#RRGGBB".

cssSelectorAncestor †

String : (Default: "#jp_container_1") : Defines the cssSelector of an ancestor of all cssSelectors. Usually an id selector, which is the outer divider wrapper of the skin HTML.

See also: `jPlayer({fullScreen})` and `jPlayer({fullWindow})`

cssSelector †

Object : (Default: {cssSelectors}) : This object defines all the selectors used to associate jPlayer's controls and feedback with the HTML in the page. By default, jPlayer uses a predefined set of cssSelectors.

noConflict

String : (Default: "jQuery") : Allows the global variable name of jQuery to be set.

This option can be changed to what you require after using the `jQuery.noConflict(true)` command. The jQuery variable name is important for the Flash fallback to communicate with jPlayer's JavaScript.

Due to a security requirement, the variable name given to the new jQuery variable must have the term `jQuery` in it. For example:

```
1.  var lib = {
2.     jQuery: jQuery.noConflict(true)
3.  }
4.  $('#jp').jPlayer({noConflict: 'lib.jQuery'});
```

See also: Security Restrictions.

wmode

String : (Default: "opaque") : Allows the wmode of the Flash fallback to be set.

Valid wmode values: window, transparent, opaque, direct, gpu

Note that audio players in Firefox 3.6 using the Flash solution will require that the `{wmode:"window"}` option is set. Otherwise, the browser does not put the Flash in the page properly. This problem does not effect Firefox 4+, nor does it affect video players in Firefox 3.6.

For example, if you used the option setting `{supplied:"mp3"}`, then you must set the `{wmode:"window"}` option to ensure Firefox 3.6 instances the Flash solution correctly.

For example, if you used the option setting `{solution:"flash,html"}`, then you must set the `{wmode:"window"}` option to ensure Firefox 3.6 instances the Flash solution correctly.

For example, if you used the option setting `{supplied:"mp3,oga"}`, then the `wmode` option does not matter, as the HTML solution will be used to play the OGA format in Firefox 3.6.

This problem will be addressed in the future, where supplying only audio formats will change the default wmode value to "window".

loop †

Boolean : (Default: false) : Sets the initial loop state.

The loop option works in tandem with the default repeat event handler. Clicks on the GUI repeat/repeat-off buttons, toggles the loop option and then generates a repeat event.

repeat †

Function : (Default: event handler) : Handler function for the repeat event.

The default repeat event handler works in tandem with the loop option. Clicks on the GUI repeat/repeat-off buttons, toggles the loop option and then generates a repeat event.

Unlike other events, the repeat event has a default handler. The default handler is shown below:

```
1.     repeat: function(event) {
2.       if(event.jPlayer.options.loop) {
3.
   $(this).unbind(".jPlayerRepeat").bind($.jPlayer.event.ended
   + ".jPlayer.jPlayerRepeat", function() {
4.           $(this).jPlayer("play");
5.         });
6.       } else {
7.         $(this).unbind(".jPlayerRepeat");
8.       }
9.     }
```

To change the repeat event handler after instancing jPlayer, unbind the handler and then bind a new handler to the event. Note that the jPlayer namespace is used to only unbind the repeat handler added by jPlayer.

```
1.     $("#my-jplayer").unbind($.jPlayer.event.repeat +
   ".jPlayer");
2.     $("#my-jplayer").bind($.jPlayer.event.repeat +
   ".jPlayer", function() {
3.       // Your new repeat handler code
4.     });
```

Remember that you may need to clean up the old handler's actions too. There may still be that ended event handler that the repeat handler generates.

```
1.     $("#my-jplayer").unbind(".jPlayerRepeat");
```

The author cannot think of an example where you would want to change the repeat handler after instancing jPlayer, but it has been documented here just in case.

emulateHtml †

Boolean : (Default: false) : Enables the HTML bridge.

This option enables a bridge that emulates the HTML media properties and events on the jPlayer element. All properties are read only. The majority of the useful media properties are emulated.

Example use with Popcorn.js. Note that this code is for an audio player.

```
1.    ready: function(event) {
2.      if(event.jPlayer.html.used &&
   event.jPlayer.html.audio.available) {
3.        // Use the actual HTML media element
4.        p = Popcorn('#' +
   $(this).data("jPlayer").internal.audio.id);
5.      } else {
6.        // Enable and use the HTML bridge with the Flash
   fallback
7.        $(this).jPlayer("option","emulateHtml",true);
8.        p = Popcorn('#' + $(this).attr("id"));
9.      }
10.   }
```

nativeVideoControls †

Object : Defines the user agent blocklist, which contains regular expressions, which cause the native controls to be used if a match is found.

Native video controls are disabled when audio media is supplied and `jPlayer({noFullWindow})` is set if nativeVideoControls found a matched.

The default object is empty:

```
1.    nativeVideoControls: {
2.      // Works well on standard browsers.
3.      // Phone and tablet browsers can have problems with
   the controls disappearing.
4.    }
```

noFullWindow †

Object : Defines the user agent blocklist, which contains regular expressions, which cause the full-screen and restore-screen buttons to be hidden if a match is found.

The default object is:

```
1.    noFullWindow: {
2.      msie: /msie [0-6]\./,
3.      ipad: /ipad.*?os [0-4]\./,
4.      iphone: /iphone/,
5.      ipod: /ipod/,
6.      android_pad: /android [0-3]\.(?!.*?mobile)/,
7.      android_phone: /android.*?mobile/,
```

```
      blackberry: /blackberry/,
      windows_ce: /windows ce/,
      iemobile: /iemobile/,
      webos: /webos/
    }
```

noVolume †

Object : Defines the user agent blocklist, which contains regular expressions, which cause the volume controls to be hidden if a match is found.

The default object is:

```
    noVolume: {
      ipad: /ipad/,
      iphone: /iphone/,
      ipod: /ipod/,
      android_pad: /android(?!.*?mobile)/,
      android_phone: /android.*?mobile/,
      blackberry: /blackberry/,
      windows_ce: /windows ce/,
      iemobile: /iemobile/,
      webos: /webos/,
      playbook: /playbook/
    }
```

timeFormat †

Object : Defines the display format of the currentTime and duration times.

The default object is whatever the `$.jPlayer.timeFormat` object is at the time of instancing.

keyEnabled †

Boolean : (Default: false) : Enables the keyboard controls feature for this instance.

The last instance played has focus. During jPlayer instancing, the first instance with the feature enabled gains focus. Only instances with the feature enabled may gain focus.

The keyboard commands are directed at the jPlayer instance in focus.

See also:

```
jPlayer({keyBindings})
jPlayer({audioFullScreen})
jPlayer("focus")
$.jPlayer.keys(enabled)
```

keyBindings †

Object : Defines the `event.which` key codes and their actions.

The keyBindings object is made up of objects of the following structure:

```
1.    helloWorld: { // A unique name or override the default
   with the name.
2.      key: 72, // The event.which key code for h
3.      fn: function(f) {
4.        // f is the instance in focus, which is this
   instance.
5.        // f.status is the status object.
6.        // f.play() to execute methods, such as play().
7.        alert("Hello World");
8.      }
9.    }
```

The instance in focus parameter `f` relates to the current instance, since the function is only called when this instance is in focus. Basically, it is a mechanism for passing through the pointer to `this` of the instance. The `this` is the JavaScript object pointer, not the DOM pointer used in the jQuery `$(this)` coding.

The jPlayerPlaylist add-on adds 2 key controls to the list, LEFT and RIGHT to move to the previous and next tracks. The jPlayerPlaylist source code may help you understand how the system works.

The default object is:

```
1.    keyBindings: {
2.      play: {
3.        key: 32, // space
4.        fn: function(f) {
5.          if(f.status.paused) {
6.            f.play();
7.          } else {
8.            f.pause();
9.          }
10.       }
11.     },
12.     fullScreen: {
13.       key: 13, // enter
14.       fn: function(f) {
15.         if(f.status.video || f.options.audioFullScreen) {
16.           f._setOption("fullScreen",
   !f.options.fullScreen);
17.         }
18.       }
19.     },
20.     muted: {
21.       key: 8, // backspace
22.       fn: function(f) {
23.         f._muted(!f.options.muted);
24.       }
25.     },
```

```
26.      volumeUp: {
27.        key: 38, // UP
28.        fn: function(f) {
29.          f.volume(f.options.volume + 0.1);
30.        }
31.      },
32.      volumeDown: {
33.        key: 40, // DOWN
34.        fn: function(f) {
35.          f.volume(f.options.volume - 0.1);
36.        }
37.      }
38.    }
```

idPrefix

String : (Default: "jp") : Defines the Id prefix for jPlayer's internally generated HTML code.

Useful if you have a naming conflict, but it is unlikely that the developer will need to change this setting.

errorAlerts

Boolean : (Default: false) : Enables error reporting through alerts.

Enable this option to help debug your jPlayer application.

warningAlerts

Boolean : (Default: false) : Enables warning reporting through alerts. Warnings are useful for the developer and will inform you of css selectors that are not found, and some other warning types. Often warnings can be ignored, such as the videoPlay selector not being found with an audio player.

Enable this option to help debug your jPlayer application.

eventType ‡

Function : (Default: undefined) : Just like the jPlayer ready event, you can bind a handler to any of the jPlayer Events Types. The events include the HTML5 media events.

You can bind handlers to events like `timeupdate` to perform an action when it occurs. See Using jPlayer Events for more information and examples.

Note that **eventType** itself is not a constructor option. Eg., The `ready` constructor option is an eventType.

Warning: The flash fallback does not emulate all events.
Warning: HTML5 media events vary x-browser.

HTML entry, with an example id for jPlayer:

```
1.  <head>
2.    <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.j
   s">
3.    </script>
4.    <script type="text/javascript" src="js/jquery.jplayer.min.js">
5.    </script>
6.    <script>
7.      $(document).ready(function() { /* Your Code */ });
8.    </script>
9.  </head>
10. <body>
11.   <div id="jpId"></div>
12. </body>
```

Code Example #1:

```
1.  $(document).ready(function() {
2.    $("#jpId").jPlayer( {
3.      ready: function () {
4.        $(this).jPlayer("setMedia", {
5.          mp3: "/mp3/elvis.mp3" // Defines the mp3 url
6.        });
7.      }
8.    });
9.  });
```

Code Example #2:

```
1.  $(document).ready(function() {
2.    $("#jpId").jPlayer( {
3.      ready: function () {
4.        $(this).jPlayer("setMedia", {
5.          m4a: "/mp3/elvis.m4a", // Defines the m4a (AAC) url
6.          oga: "/ogg/elvis.ogg" // Defines the counterpart oga url
7.        });
8.      },
9.      supplied: "m4a, oga",
10.     swfPath: "/jPlayer/js"
11.   });
12. });
```

Code Example #3:

```
1.  $(function() { // executed when $(document).ready()
2.    $("#jpId").jPlayer( {
3.      ready: function () {
4.        $(this).jPlayer("setMedia", {
```

```
5.           m4v: "http://www.myDomain.com/myVideo.m4v" // Defines
   the m4v url
6.         }).jPlayer("play"); // Attempts to Auto-Play the media
7.       },
8.       supplied: "m4v",
9.       swfPath: "jPlayer/js"
10.    });
11.  });
```

Code Example #4:

```
1.   $(function() { // executed when $(document).ready()
2.     $("#jpId").jPlayer( {
3.       ready: function () {
4.         $(this).jPlayer("setMedia", {
5.           m4v: "/media/myVideo.m4v", // Defines the m4v url
6.           ogv: "/media/myVideo.ogv" // Defines the counterpart ogv
   url
7.         }).jPlayer("play"); // Attempts to Auto-Play the media
8.       },
9.       solution: "flash, html", // Flash with an HTML5 fallback.
10.      supplied: "m4v, ogv",
11.      swfPath: "/scripts"
12.    });
13.  });
```

Code Example #5: **Bad Code!**

```
1.   $(document).ready(function() {
2.     $("#jpId").jPlayer( {
3.       ready: function () {
4.         $(this).jPlayer("setMedia", {
5.           mp3: "/mp3/elvis.mp3"
6.         });
7.       }
8.     });
9.     $("#jpId").jPlayer("play"); // BAD: The plugin is not ready
   yet
10.  });
```

jPlayer Methods

jPlayer is controlled by sending the method name through the `$(id).jPlayer()` plugin method.

Methods that are an option alias

Methods that are a shortcut/alias to change options using the `jPlayer("option", key, value)` method, are indicated by the † symbol.

$(id).jPlayer("setMedia", Object: media) : jQuery

Description

This method is used to define the media to play. The media parameter is an object that has properties defining the different encoding formats and a poster image.

The `jPlayer("setMedia", media)` method must be used to define the media before jPlayer will be able perform other methods like `jPlayer("play")`.

All formats defined in the constructor option `jPlayer({supplied: "f1, f2, fN"})` must be given a valid url to the media encoded in that format. The exception is when using both video and audio media types in a single jPlayer. Then either all the `supplied` **audio** formats or all the `supplied` **video** formats must be defined.

The constructor option `jPlayer({solution: "s1, s2")` can be used to define the priority of the HTML and Flash solutions used by jPlayer.

jPlayer distinguishes between the media types, and using both video and audio together in a single `setMedia` command makes it difficult to determine what actually plays. The decision is based on the priority of `solution` and `supplied` and the browser being used. ie., The 1st one `supplied` that can work in the `solution`, will work, be that video or audio.

The point at which jPlayer begins downloading the media is complicated by changes in the by changes in the W3C HTML5 media spec for the `media.load()` method and the `media.src` property. In order for older HTML5 browsers to work with jPlayer, the `media.load()` command is still used, where really it should be removed as it is no longer required.

The jPlayer constructor option `jPlayer({preload})` is used to control when jPlayer begins downloading new media. Some browsers will begin the download for the whole media when the preload option is "metadata". Others will require that "auto" is used. The only way to ensure download does not begin automatically is to use the preload option "none". Remember that, the media may not download as it depends on the browser you are using. Mobile browser such as Mobile Safari on iOS 4.2 require a gesture before any media is downloaded or played. ie., The user must press a button to initiate the load or play operations the 1st time.

Any media playing when the command is issued will be paused. The the download may not be cancelled as it depends on whether the browser follows the recent W3C HTML media spec. In practice, except for the first time setMedia is used in the ready event, the setMedia command is followed by a play command which stops the old download. The exception is with the media player, where the old media of different type (audio or video), will continue downloading. Otherwise the iPhone's built in video player hangs when the src is cleared to cancel the download.

Do not use relative URLs for the media. The media URLs must be absolute or relative to the server root. For example, "http:www.jplayer.org/media/video.m4v" or "/media/video.m4v".

Relative URLs will have problems with the Flash fallback with M4A and M4V formats. If your Jplayer.swf file is on another domain, (for some reason that makes sense to you,) then absolute URLs must be used.

The URLs given must conform to standard URL Encoding Rules.

Parameters

media

Object : Defines the media format URLs and poster

mp3
: String : Defines the URL of the mp3 audio format

m4a
: String : Defines the URL of the mp4 audio format (MP4: AAC)

m4v
: String : Defines the URL of the mp4 video format (MP4: H.264/AAC)

webma
: String : Defines the URL of the webm audio format (WebM: Vorbis)

webmv
: String : Defines the URL of the webm video format (WebM: VP8/Vorbis)

oga
: String : Defines the URL of the ogg audio format (OGG: Vorbis)

ogv
: String : Defines the URL of the ogg video format (OGG: Theora/Vorbis)

fla
: String : Defines the URL of the flv audio format (FLV: Flash)

flv
: String : Defines the URL of the flv video format (FLV: Flash)

wav
: String : Defines the URL of the wav audio format

poster
: String : Defines the URL of the poster image.
The poster image should have the same aspect ratio as the jPlayer size, or the image will be distorted when it is resized to fit the area.

track
: Array : Each array element is an object defining a `<track>` element.
This property is partially implemented without suitable GUI controls exposed.
There is no polyfill for browsers (or the flash solution), which does not support WebVTT.
While the use of this is limited, it might be suitable when the

`jPlayer({nativeVideoControls})` option is used and your target browser support is specific to those with WebVTT support.
This follows the W3C Media Element Living Standard and the WebVTT Living Standard, see them for more details.
kind
String : Defines the type of track.
src
String : Defines the URL of the track.
srclang
String : Defines the language of the track.
label
String : Defines the label for the track.
def
Boolean : True if this is the default track.

Note that only the properties listed are used by jPlayer, so you could store all kinds of other info in the same object if your project works better that way. For example, the title and album names of an audio track. However, we do plan to expand the media object properties in the future Properties may one day include: stream, chapters, subtitles.

Code Examples:

```
1.   $("#jpId").jPlayer( {
2.     ready: function() {
3.       $(this).jPlayer( "setMedia", {
4.         m4a: "m4a/elvis.m4a",
5.         oga: "oga/elvis.oga",
6.         webma: "webm/elvis.webm"
7.       });
8.     },
9.     supplied: "webma, m4a, oga"
10.  );
```

$(id).jPlayer("clearMedia") : jQuery

Description

This method is used to clear the media and stop playback. If a media file is downloading at the time, the download will be cancelled.

After this command is issued, playback commands such as `jPlayer("play")` will be ignored and raise error events until after new media has been specified using the `jPlayer("setMedia",media)` method.

There are very few circumstances when it is appropriate to use this method. In general when you are changing media, this action is automatically performed.

Using this clearMedia by itself on the iOS iPhone or iPod Touch will corrupt the built in video player if it is open at the time the command executes.

Parameters

This method has no parameters.

Code Examples:

```
1.  $("#jpId").jPlayer( "clearMedia" );
```

$(id).jPlayer("load") : jQuery

Description

This method is used to preload the media before the play command is given. There is no point using `load` if you are going to `jPlayer("play", [time])` immediately afterwards, just `play` it. Likewise, with `jPlayer("pause", [time])`, if a time greater than zero is given.

This method allows you to selectively preload the files you choose after changing the file using setMedia. If you want all files to preload by default, use the jPlayer constructor option `{preload:"auto"}`.

This command is affected by browser and some ignore the command, or require a gesture to initiate it the 1st time.

Parameters

This method has no parameters.

Code Examples:

```
1.  $("#jpId").jPlayer("load");
```

$(id).jPlayer("play", [Number: time]) : jQuery

Description

This method is used to play the media specified using `jPlayer("setMedia", media)`.

If necessary, the file will begin downloading.

Without the `time` parameter, new media will play from the start. Open media will play from where the play-head was when previously paused using `jPlayer("pause", [time])`.

The optional `time` paramater will move the play-head position to the time given in seconds. `jPlayer("play", 0)` is useful for forcing play from the start of the track, but is redundant for new media, which always starts from the beginning. This can be used to jump immediately to a play time afer changing media. jPlayer will seek to the time given and play when able to.

If issued immediately after a `setMedia` command, with the `time` parameter, and when the browser is using the HTML5 solution, this command will initially fail and an internal timeout is setup to retry the command every 100ms until it succeeds.

Parameters

time

> [Optional] Number : Defines the new play-head position in seconds.
>
> If used while downloading, play will begin once the media is seekable to that point.

To make media playback loop, add `jPlayer("play")` to the ended event. jPlayer deals with reseting the media to the start in the ended event, so the `time` equals zero is not required.

Code Examples:

```
1.  $("#jpId").jPlayer( {
2.    ready: function() { // The $.jPlayer.event.ready event
3.      $(this).jPlayer("setMedia", { // Set the media
4.        m4v: "m4v/presentation.m4v"
5.      }).jPlayer("play"); // Attempt to auto play the media
6.    },
7.    ended: function() { // The $.jPlayer.event.ended event
8.      $(this).jPlayer("play"); // Repeat the media
9.    },
10.   supplied: "m4v"
11. );
12.
13. $("#jumpToTime").click( function() {
14.   $("#jpId").jPlayer("play", 42); // Begins playing 42 seconds
    into the media.
15. });
```

$(id).jPlayer("pause", [Number: time]) : jQuery

Description

This method is used to pause the media.

The `jPlayer("pause", time)` can be used to jump immediately to a play-head time afer changing media. The media will begin downloading, and cue the media ready when able to.

Without the `time` parameter, new media that has not been loaded or played will ignore this command. This avoids downloading the new media when the redundant command is given. Open media will pause if it was playing, otherwise there will be no effect.

The optional `time` paramater will move the play-head position to the time given in seconds and if necessary, the file will begin downloading. `jPlayer("pause", 0)` is the same as `jPlayer("stop")`. New media will ignore the `jPlayer("pause", 0)` commmand until after the media has started loading. This avoids downloading the new media when the redundant command is given.

If issued immediately after a `setMedia` command, with the `time` parameter, and when the browser is using the HTML5 solution, this command will initially fail and an internal timeout is setup to retry the command every 100ms until it succeeds.

Parameters

time

> [Optional] Number : Defines the new play-head position in seconds.
>
> If used while downloading, the play-head will cue once the media is seekable to that point.

Using a `jPlayer("pause", 0)` in the `ended` event is redundant and should not be used.

Code Examples:

```
1.  $("#jpId").jPlayer("pause", 10); // Pauses and moves the play-
    head 10 seconds into the song.
```

$(id).jPlayer("pauseOthers") : jQuery

Description

This method pauses all instances except the instance that invoked the command. This is useful in events such as the play event to pause all the other players on the page when an instance starts playing.

The play event handler can be defined in the constructor options.

Parameters

This method has no parameters.

Code Examples:

```
1.   $(id).bind($.jPlayer.event.play, function() { // Bind an event
   handler to the instance's play event.
2.     $(this).jPlayer("pauseOthers"); // pause all players except
   this one.
3.   });
```

$(id).jPlayer("stop") : jQuery

Description

This method is used to stop the media and reset the play-head to the start of the media.

This command is the same as `jPlayer("pause", 0)`. The method is available so that a stop button can be used in your interface, if you want one.

If issued immediately after a `setMedia` command, this command is ignored.

Parameters

This method has no parameters.

Using a `jPlayer("stop")` in the `ended` event is redundant and should not be used. This avoids a fatal bug on Chrome (Win/OSX), where the browser crashes if you were to follow the `jPlayer("stop")` with a `jPlayer("setMedia")` in the `ended` event.

Code Examples:

```
1.   $("#jpId").jPlayer("stop");
```

$(id).jPlayer("playHead", Number: percentOfSeekable) : jQuery

Description

This method moves the play-head to a new position. The primary use is internal to the plugin to handle clicks on the seekBar and move the play-head to the new position.

Note that this only moves the play-head. Whether the media plays from that point depends on its current state. ie., If it was playing, play continues from the new play-head. If it was paused, the media is cued to the new play-head position.

If issued immediately after a `setMedia` command, the effect is the same as `jPlayer("pause", 0)`, which is ignored.

Parameters

percentOfSeekable

Number (0 to 100) defining the percentage played when compared to the current percentage seekable.

Only when completely seekable does the percentage relate to the total length of the media.
Most HTML5 browsers have seeking enabled, so the seekable value jumps to 100% when the media starts downloading.

Code Examples:

```
1.  $("#jpId").jPlayer("playHead", 0); // Move play-head to start.
2.  $("#jpId").jPlayer("playHead", 10); // Move play-head to 10% of
    the seekable length.
```

$(id).jPlayer("option", [String: key, [Mixed: value]]) : Mixed

Description

This method is used to access configuration information inside jPlayer.

The options are defined in the constructor. Full get access is enabled, where a copy of the option is returned. However you may currently only set a limited number of options after instancing jPlayer. These are indicated by the † symbol. See also, the jPlayer Options.

Parameters

key

String defining the option property name. Supports dot notation.

The returned value can is either a Boolean, Number, String, or an Object, depending on the data requested.
Event handlers defined in the constructor are not included, since they can be removed using the ".jPlayer" namepspace.

value

The new value of the option.

Events

$.jPlayer.event.warning
Event Codes
$.jPlayer.warning.OPTION_KEY

Code Examples:

```
1.  var solution = $("#jpId").jPlayer("option", "solution"); // Get
    the solution string, "html, flash" by default.
2.  var cssSelector = $("#jpId").jPlayer("option", "cssSelector");
    // Get the cssSelector object.
3.
4.  var playSelector = $("#jpId").jPlayer("option",
    "cssSelector.play"); // Get the cssSelector for the play method.
5.  var playSelector = $("#jpId").jPlayer("option",
    "cssSelector").play; // Get the cssSelector for the play method.
6.
7.  $("#jpId").jPlayer("option", "cssSelector.play", ".my-new-
    class"); // Set the cssSelector for the play method.
8.  $("#jpId").jPlayer("option", "cssSelector", {play:".my-new-
    class"}); // Set the cssSelector for the play method. (Can set
    multiple selectors in the object.)
9.
10. $("#jpId").jPlayer("option", "cssSelectorAncestor", "#my-new-
    interface"); // Set the cssSelectorAncestor and refresh all
    associations.
```

$(id).jPlayer("focus") : jQuery

Description

This method is used to gain the focus of keyboard controls without playing.

For more information on keyboard controls see the `jPlayer({keyEnabled})` option.

Parameters

This method has no parameters.

Code Examples:

```
1.  $("#jpId").jPlayer("focus");
```

$(id).jPlayer("destroy") : jQuery

Description

This method removes jPlayer. All event and interface bindings created by jPlayer are removed. After destroying the instance, a new jPlayer can be instanced on the element.

All event handlers with the `.jPlayer` namespace are removed by this method.

Parameters

This method has no parameters.

Code Examples:

```
1.  $("#jpId").jPlayer("destroy");
```

$(id).jPlayer("volume", Number: ratio) : jQuery †

Description

This method is used to control the volume of the media being played. Note that the initial volume is set through the constructor option: `jPlayer({volume}:ratio)`. The volume can be changed through the option method: `jPlayer({option}, "volume", ratio)`.

While muted you can change this value, but the media will remain muted.
Setting the volume to zero is not the same as using `mute`. The two systems are independent.

Parameters

ratio

Number (0 to 1) defining the ratio of maximum volume.

Silence: 0
Half: 0.5
Maximum: 1

Code Examples:

```
1.  $("#jpId").jPlayer("volume", 0.75);
```

$(id).jPlayer("mute", [Boolean:mute]) : jQuery †

Description

This method mutes the media's sounds. Note that the initial muted state is set through the constructor option: `jPlayer({muted}:Boolean)`. The mute state can be changed through the option method: `jPlayer({option}, "muted", Boolean)`.

Parameters

mute

Boolean : (Default: true) : The muted state.

Code Examples:

```
1.  $("#jpId").jPlayer("mute");
```

$(id).jPlayer("unmute", [Boolean:unmute]) : jQuery †

Description

This method unmutes the media's sounds. Note that the initial muted state is set through the constructor option: `jPlayer({muted:false})` The mute state can be changed through the option method: `jPlayer({option}, "muted", Boolean).`

Parameters

unmute

Boolean : (Default: true) : The oposite of the muted state.

Code Examples:

```
1.  $("#jpId").jPlayer("unmute");
```

jPlayer Options

jPlayer allows you to change certain options after instancing. These are indicated by the † symbol. All other options must be defined at creation through the constructor options.

Due to the nature of the cssSelector and cssSelectorAncestor options, there are described here in more detail.

$(id).jPlayer("option", "cssSelector", String: method, String: selector) : jQuery

Description

This option allows developers to change the cssSelector associations after jPlayer has been instanced on the page.

The `cssSelector` method is used with the jPlayer constructor option `cssSelectorAncestor` to create associations between jPlayer methods and CSS entities on the webpage. For example, this enables a play button graphic on the webpage to be associated with the method that executes the play command.

By default, jPlayer uses a predefined set of cssSelectors, where all the css selectors are class selectors. The strings are added to the end of the `cssSelectorAncestor` string, with a space in between. The default `cssSelectorAncestor` is an id selector. By using a single id and a common class structure, multiple instances of jPlayer are easy to add to the page. The idea is that jPlayer is associated with a unique interface, hence the id, and then the rest of the structure is common, hence the classes.

The `method` may only have one `selector` associated with it through jPlayer. An existing association will be removed if a new association is given. An empty string clears exisiting associations.

jPlayer uses the ".jPlayer" event namespace for binding and unbinding methods to CSS entities.

Parameters

method

String containing the name of the method to associate with the `selector`.
See the predefined cssSelectors for the list of valid method names.

selector

String containing the CSS Selector to associate with the `method`.
Usually a class selector. Eg., ".jp-play"

Events

$.jPlayer.event.warning
Event Codes
$.jPlayer.warning.CSS_SELECTOR_COUNT
$.jPlayer.warning.CSS_SELECTOR_METHOD
$.jPlayer.warning.CSS_SELECTOR_STRING

Code Examples:

```
1.   $(document).ready(function() {
2.     $("#jpId").jPlayer( {
3.       ready: function () {
4.         $(this).jPlayer("setMedia", {mp3: "mp3/elvis.mp3"});
5.       },
6.       cssSelectorAncestor: "", // Remove the ancestor css selector
   clause
7.       cssSelector: {
8.         play:"#myOldPlayButton" // Set a custom css selector for
   the play button
9.         // The other defaults remain unchanged
10.      }
11.    })
12.
13.    // ... Some time later, otherwise you'd just set it in the
   constructor ...
14.
15.    $("#jpId").jPlayer( "option", "cssSelector", "play",
   "#myNewPlayButton" );
16.  });
```

$(id).jPlayer("option", "cssSelectorAncestor", String: selector) : jQuery

Description

This option allows the developers to change the associations after jPlayer has been instanced on the page.

Setting this option causes all the cssSelector associations to be refreshed with the new ancestor. Allowing you to switch between interfaces easily. I am not sure why you would want to do that, but it's here if you need it.

Parameters

selector

> String containing the CSS Selector to associate with the `cssSelectorAncestor`
> Usually an id selector. Eg., "#jp_container_1"

Events

$.jPlayer.event.warning
> Event Codes
> $.jPlayer.warning.CSS_SELECTOR_COUNT

Code Examples:

```
1.   $(document).ready(function() {
2.     $("#jpId").jPlayer( {
3.       ready: function () {
4.         $(this).jPlayer("setMedia", {mp3: "mp3/elvis.mp3"});
5.       },
6.       cssSelectorAncestor: "#interface-on-the-left" // Define the
   ancestor css selector
7.     })
8.
9.     // ... Some time later, otherwise you'd just set it in the
   constructor ...
10.
11.    $("#jpId").jPlayer( "option", "cssSelectorAncestor",
   "#interface-on-the-right" );
12.  });
```

jPlayer Events

jPlayer communicates with your code via events.

All jPlayer events can have a handler defined using the constructor option of the same name:
`jPlayer({eventName:function(e){...}})`
Event handlers created by constructor options have the `".jPlayer"` namespace.

jPlayer Event Types

1. $.jPlayer.event.ready * † Occurs when jPlayer is ready to receive commands.
2. $.jPlayer.event.flashreset * † Occurs when the Flash solution generates another ready event.
3. $.jPlayer.event.resize * † Occurs when the screen state changes and when the size/sizeFull options are changed.
4. $.jPlayer.event.repeat * † Occurs when the loop state is changed and before the ready event.
5. $.jPlayer.event.click * † Occurs when the user clicks on the poster or video. NB: The GUI skin can interfere with this.
6. $.jPlayer.event.error * ‡
7. $.jPlayer.event.warning * †
8. $.jPlayer.event.loadstart *
9. $.jPlayer.event.progress * Occurs while the media is being downloaded.
10. $.jPlayer.event.suspend
11. $.jPlayer.event.abort
12. $.jPlayer.event.emptied
13. $.jPlayer.event.stalled
14. $.jPlayer.event.play * Occurs when the media is played.
15. $.jPlayer.event.pause * Occurs when the media is paused.
16. $.jPlayer.event.loadedmetadata
17. $.jPlayer.event.loadeddata
18. $.jPlayer.event.waiting
19. $.jPlayer.event.playing
20. $.jPlayer.event.canplay
21. $.jPlayer.event.canplaythrough
22. $.jPlayer.event.seeking *
23. $.jPlayer.event.seeked *
24. $.jPlayer.event.timeupdate * Occurs when the currentTime is changed. (~250ms between events during playback.)
25. $.jPlayer.event.ended * Occurs when the media ends.
26. $.jPlayer.event.ratechange
27. $.jPlayer.event.durationchange
28. $.jPlayer.event.volumechange * Occurs when volume or muted change.

    * A jPlayer event supported by both the Flash fallback and HTML5 browsers
    † jPlayer event that is not part of the HTML5 Spec
    ‡ jPlayer event that differs from the html Spec. ie., There is more than 1 type of error.

We plan to expand the emulation of HTML5 Media events in the Flash.

jPlayer Event Object

eventHandler(event)

event

Object: Standard jQuery $.Event() properties.

event.jPlayer
    Object : The jPlayer information object.
event.jPlayer.error
    Object
event.jPlayer.error.type
    String : The error event code
event.jPlayer.error.context
    String : The cause of the error
event.jPlayer.error.message
    String : Message describing the error
event.jPlayer.error.hint
    String : A suggestion on how to fix the error
event.jPlayer.flash
    Object : Info about the Flash solution
event.jPlayer.html
    Object : Info about the HTML solution
event.jPlayer.options
    Object : The jPlayer options. The volume and muted values are maintained here along with all the other options.
event.jPlayer.status
    Object
event.jPlayer.status.src
    String : The URL being used by jPlayer.
event.jPlayer.status.media
    Object : Pointer to the media object used in setMedia.
event.jPlayer.status.seekPercent
    Number : Percent seekable
event.jPlayer.status.currentPercentRelative
    Number : Current time as a percent of seekPercent
event.jPlayer.status.currentPercentAbsolute
    Number : Current time as a percent of duration
event.jPlayer.status.currentTime
    Number : The current time in seconds
event.jPlayer.status.duration
    Number : The duration of the media
event.jPlayer.status.srcSet
    Boolean
event.jPlayer.status.paused
    Boolean
event.jPlayer.status.waitForPlay
    Boolean
event.jPlayer.status.waitForLoad
    Boolean
event.jPlayer.status.video
    Boolean
event.jPlayer.status.width

String : The current CSS style width of jPlayer.
event.jPlayer.status.height
String : The current CSS style height of jPlayer.
event.jPlayer.status.videoWidth
Number : The intrinsic width of the video in pixels. (Zero before known or if audio.)
event.jPlayer.status.videoHeight
Number : The intrinsic height of the video in pixels. (Zero before known or if audio.)
event.jPlayer.version
Object
event.jPlayer.version.script
String : jPlayer's JavaScript version
event.jPlayer.version.flash
String : jPlayer's Flash version, or "unknown" if Flash is not being used
event.jPlayer.version.needFlash
String : The Flash version compatiable with the JavaScript
event.jPlayer.warning
Object
event.jPlayer.warning.type
String : The warning event code
event.jPlayer.warning.context
String : The cause of the warning
event.jPlayer.warning.message
String : Message describing the warning
event.jPlayer.warning.hint
String : A suggestion on how to fix the warning

jPlayer Error Event Codes

event.jPlayer.error.type

$.jPlayer.error.FLASH
A problem with the Flash insertion on the page.
$.jPlayer.error.FLASH_DISABLED
A ready event was registered from the Flash, but commands are no longer working. The Flash has been disabled by the browser because it, or a parent, has been hidden using `display:none`
$.jPlayer.error.NO_SOLUTION
No media playback solution is available.
$.jPlayer.error.NO_SUPPORT
Not possible to play any media format provided in `setMedia`.
$.jPlayer.error.URL
Error with media URL. †
$.jPlayer.error.URL_NOT_SET
Media playback command not possible as no media is set.
$.jPlayer.error.VERSION

Mismatch in versions of JavaScript and Flash of jPlayer.

† Equivilent to the HTML5 Media error event. (Not exactly true... The HTML error event can occur due to decoding errors too.)

jPlayer constructor option `jPlayer({errorAlerts:true})` will create alerts when an error event occurs.

jPlayer Warning Event Codes

event.jPlayer.warning.type

$.jPlayer.warning.CSS_SELECTOR_COUNT
: The number of jPlayer('cssSelector') or jPlayer('cssSelectorAncestor') found did not equal one.

$.jPlayer.warning.CSS_SELECTOR_METHOD
: The jPlayer('cssSelector') method is not valid.

$.jPlayer.warning.CSS_SELECTOR_STRING
: The jPlayer('cssSelector') was not a valid string.

$.jPlayer.warning.OPTION_KEY
: The option requested in jPlayer('option') is undefined.

jPlayer constructor option `jPlayer({warningAlerts:true})` will create alerts when a warning event occurs.

Using jPlayer Events

Just like any other event in jQuery, jPlayer events are bound to handler functions using `jQuery.bind()`. To remove an event use `jQuery.unbind()`.

Use the $.jPlayer.event object to access event type strings. All events have a jPlayer Event Object with easy access to information at the time the event occured.

All jPlayer events can have a handler defined using the constructor option of the same name.
`jPlayer({eventType:handler})`

Note that, the event type used in the constructor is the property name of the event, not the event string code. For example, the event type `$.jPlayer.event.ready` has the constructor option `jPlayer({ready:handler})`:

Event handlers created by constructor options have the `".jPlayer"` namespace.

Code Examples:

```
1.  $("#repeat-on").click( function() {
```

```
2.     $("#jpId").bind($.jPlayer.event.ended + ".jp-repeat",
   function(event) { // Using ".jp-repeat" namespace so we can
   easily remove this event
3.       $(this).jPlayer("play"); // Add a repeat behaviour so media
   replays when it ends. (Loops)
4.     });
5.     return false;
6.    });
7.
8.   $("#repeat-off").click( function() {
9.     $("#jpId").unbind($.jPlayer.event.ended + ".jp-repeat"); //
   Remove the ended events with the ".jp-repeat" namespace
10.    return false;
11.   });
12.
13.  $("#jpId").bind($.jPlayer.event.play, function(event) { // Add a
   listener to report the time play began
14.    $("#playBeganAtTime").text("Play began at time = " +
   event.jPlayer.status.currentTime);
15.  });
16.
17.  $("#jpId").unbind($.jPlayer.event.play); // Remove all play
   event listeners
18.
19.
20.  $("#jpId").bind($.jPlayer.event.error + ".myProject",
   function(event) { // Using ".myProject" namespace
21.    alert("Error Event: type = " + event.jPlayer.error.type); //
   The actual error code string. Eg., "e_url" for
   $.jPlayer.error.URL error.
22.    switch(event.jPlayer.error.type) {
23.      case $.jPlayer.error.URL:
24.        reportBrokenMedia(event.jPlayer.error); // A function you
   might create to report the broken link to a server log.
25.        getNextMedia(); // A function you might create to move on
   to the next media item when an error occurs.
26.        break;
27.      case $.jPlayer.error.NO_SOLUTION:
28.        // Do something
29.        break;
30.    }
31.  });
32.
33.  $("#jpId").unbind(".myProject"); { // Remove ".myProject"
   namespace event listeners using standard jQuery method
34.
```

jPlayer Functions

$.jPlayer.pause() : Void

Description

Pauses all instances of jPlayer on the current page. If using frames, then only the the frame the command is issued in is affected.

Parameters

This method has no parameters.

Code Examples:

```
1.  $.jPlayer.pause(); // Pause all instances of jPlayer on the page
```

$.jPlayer.convertTime(Number: seconds) : String

Description

This function is used to convert a time, in seconds, to a string formatted in hours, minutes and seconds. The format of the conversion is defined using the object `$.jPlayer.timeFormat`.

This function is used to format the text jPlayer writes to the css selectors for currentTime and duration, however, a jPlayer instance uses it own `jPlayer({timeFormat:Object})` option.

Parameters

seconds
: Number : The number of seconds to convert.

Returns

String : The formatted time.

Code Examples:

```
1.  $("#myTime").text($.jPlayer.convertTime(60)); // One minute
```

$.jPlayer.keys(Boolean: enable) : Void

Description

By default, this function has been executed with `enable` set true. This generates the keyboard event handler on the document, which is used to detect key bindings and direct them to the jPlayer instance in focus.

In general, you should not need to use this function. It has been documented since it may be of use if you wish to globally toggle keyboard controls on and off for multiple instances.

For more information on keyboard controls see the `jPlayer({keyEnabled})` option.

When enabled, the `keydown.jPlayer` event is bound to the `document.documentElement` element.

Parameters

enable
: Boolean : To bind or unbind the global keyboard event handler.

Code Examples:

```
1.  $.jPlayer.keys(false); // Turn off all key controls
2.  $.jPlayer.keys(true); // Turn on key controls for instances with
    keyEnabled set.
```

jPlayer Objects

$.jPlayer.timeFormat : Object

Description

This object is used to format the time returned by the function:
`$.jPlayer.convertTime( seconds )`

The formating is cumulative. For example, the default settings display the minutes and seconds, so a time that is over an hour value will be added on to the minutes. In this case, a time of 1 hour 45 minutes and 10 seconds, would display as 105:10. To make the time display as 1:45:10 then set
`$.jPlayer.timeFormatshowHour = true;`

This object defines the default value of the `jPlayer({timeFormat:Object})` option.

Properties

showHour
: Boolean : (Default: false) : Displays the hours.

showMin
: Boolean : (Default: true) : Displays the minutes.

showSec
: Boolean : (Default: true) : Displays the seconds.

padHour
: Boolean : (Default: false) : Zero pads the hour if less than 10.

padMin
: Boolean : (Default: true) : Zero pads the minute if less than 10.

padSec
: Boolean : (Default: true) : Zero pads the second if less than 10.

sepHour
: String : (Default: ":") : String between hour and minute.

sepMin
: String : (Default: ":") : String between minute and second.

sepSec
: String : (Default: "") : String after second.

Code Example:

```
1.  $.jPlayer.timeFormat.showHour = true;
2.  $.jPlayer.timeFormat.sepHour = " hours ";
3.  $.jPlayer.timeFormat.sepMin = " minutes ";
4.  $.jPlayer.timeFormat.sepSec = " seconds";
```

$.jPlayer.platform : Object

Description

This object is the result of a user agent sniffer and gives information for mobile and tablet platforms.
When detected, the properties are true:

Properties

mobile
: Boolean : True when a smart phone is detected.

tablet
: Boolean : True when a tablet device is detected.

ipad
iphone
ipod
android
blackberry
playbook
webos
windows_ce

Code Example:

```
1.  if($.jPlayer.platform.tablet) {
2.          // Do something for all tablet devices
3.          if($.jPlayer.platform.ipad) {
4.             // Do something on ipad devices
5.          }
6.          if($.jPlayer.platform.android) {
7.             // Do something on android tablet devices
8.          }
9.  }
```

jPlayer's Predefined CSS Selectors

jPlayer has a predefined set of CSS selectors built in. Below are the default associations between the jPlayer method and the CSS selector.

To define custom CSS selectors use the constructor options:
`jPlayer({cssSelectorAncestor: "#my-unique-id", cssSelector: {...}})`

To change CSS selectors after instancing, use `jPlayer("option", key, value)`:
```
jPlayer("option", "cssSelector", method, selector)
jPlayer("option", "cssSelectorAncestor", selector)
```

Default cssSelectorAncestor

```
1.  cssSelectorAncestor: "#jp_container_1"
```

Default cssSelector

method: "selector"

```
1.  cssSelector: {
2.    videoPlay: ".jp-video-play",
3.    play: ".jp-play",
4.    pause: ".jp-pause",
5.    stop: ".jp-stop",
6.    seekBar: ".jp-seek-bar",
7.    playBar: ".jp-play-bar",
8.    mute: ".jp-mute",
9.    unmute: ".jp-unmute",
10.   volumeBar: ".jp-volume-bar",
11.   volumeBarValue: ".jp-volume-bar-value",
12.   volumeMax: ".jp-volume-max",
13.   currentTime: ".jp-current-time",
14.   duration: ".jp-duration",
15.   fullScreen: ".jp-full-screen",
16.   restoreScreen: ".jp-restore-screen",
17.   repeat: ".jp-repeat",
18.   repeatOff: ".jp-repeat-off",
19.   gui: ".jp-gui",
20.   noSolution: ".jp-no-solution"
21. }
```

Actual CSS Selector String Coding

```
selector = cssSelectorAncestor + " " + cssSelector[method];
```

For example, the play method's default actual selector is: `"#jp_container_1 .jp-play"`
In English, this means select the entity with class `jp-play` that has an ancestor with the id `jp_container_1`.

Remember that ids are unique, while classes are common. An id should only ever be used once in the HTML.

jPlayer Graphical Skins

The HTML and CSS/Artwork varies from skin to skin. The CSS/Artwork is a designed to work with all of that skin's HTML structures. You should get the HTML structure from the demo ZIP, available on the download page. The PSD files are also available for each skin.

Accessibility

While the skin conforms with the HTML 4.01 Spec, unfortunately some browsers do not adhere to the spec. For example, Safari does not include `<a>` elements in the tab order. If you wish to correct this, the `<a>` can be replaced with `<input>` or `<button>` elements. Using these form elements requires an empty transparent GIF file to be specified for each element, which complicates use when used in dynamic web apps such as WordPress. The `<input>` and `<button>` elements also behave differently cross-browser, in particular where Internet Explorer moves the artwork down and to the right when it is clicked on. For these reason, our standard skin uses the `<a>` element.

The `accesskey` attribute may be added to each `<a>` element to provide access key support. The use of access keys depends on your target users. Most users do not even know how to initiate an access key and it varies from browser to browser. The key itself also needs to be conveyed to the user in some manner. Since access keys can interfere with the operation of some browsers we have not used the attribute in the default skin. After invesigation, a clearly labled link is prefered by the majority of screen reader users. ie., Our play button has the word "play" in the link, even though the text is not visible when using a standard browser.

jPlayer Compatibility

jQuery

Compatibility verified with:

- jQuery 2.0.x
- jQuery 1.10.x
- jQuery 1.9.x
- jQuery 1.8.x
- jQuery 1.7.x ‡
- jQuery 1.6.x † ‡
- jQuery 1.5.x † ‡
- jQuery 1.4.x † ‡
- jQuery 1.3.x † ‡

† The jPlayerPlaylist add-on requires jQuery 1.7+
‡ Deprecated.

Zepto

Compatibility verified with:

- Zepto 1.0 compiled with Data module

Browser

Compatibility verified with:

- Chrome 27 (Win, OSX) †
- Firefox 21 (Win, OSX) †
- Firefox 3.6 (Win)
- Safari 6.0.4 (OSX) †
- Opera 12.15 (Win, OSX) †
- Internet Explorer 10 (Win)
- Internet Explorer 9 (Win)
- Internet Explorer 8 (Win)
- Internet Explorer 7 (Win)
- Internet Explorer 6 (Win)

† Tested using the latest browser release when this jPlayer version was released.

Apple dropped support for Windows in Safari 5.1.7

Mobile Browser

Compatibility verified with:

- Mobile Safari (iOS 6.1: iPad / iPhone / iPod Touch) †
    - All iOS browsers use this engine.
    - iPhone / iPod Touch: Video plays in the QuickTime plugin.
    - Prior to iOS 6, the Play and load bar progress did not work well when using multiple instances of jPlayer. This browser enables one media element at a time, so playing 2 instances together is not possible.
- Mobile Chrome (Android 4.2.2: Nexus 7) †
    - The tab must be in focus for the media to play.
- Playbook Browser †

† Tested using the latest browser release when this jPlayer version was released.

jPlayer Known Issues

Android HTML5 Issues

The incomplete implementation of the HTML5 Media Spec on Android 2.3 causes problems with jPlayer's behaviour.

Issues with Android 2.3:

- Missing ended events.
- Failure when `setMedia` is followed by `play`.
- Failure when the media has finished playing. The audio that was just playing is corrupted by it ending.

Take a look at this jPlayer Demo to see a working solution to these issues.

Incompatible Browsers

jPlayer does not work on the following browsers:

- Wii Opera
- Playstation 3 Browser

Event Driven Mobile Browsers

The following Mobile browsers require user gestures to trigger commands that effect media playback.

- Mobile Safari
    - The first time a media element is played must be initiated by a user gester. ie., The user must click the play button. This affects the operation of a `jPlayer("play")` in the ready event handler. The browser will ignore the command. jPlayer will simply wait until the user presses the play button. Once the first gesture has been received, JavaScript code is then allowed to do whatever you want with the media element. Note that a jPlayer media player instance uses a audio and a video element. Each require their own gesture.
      Also affect: `jPlayer("load")` and `jPlayer("pause", time)`

Reference: HTML5 Browser Audio and Video Support

Here are details on the supported formats by HTML5 browsers. All browser versions are their latest official release.

Format names relate to the supplied and setMedia format property names used by jPlayer. For example: WEBMA is an audio WebM, OGV is a video OGG, and M4A is an audio MP4. For more details on the formats, see Media Encoding.

HTML5 Audio Format

An audio WebM file uses the same Vorbis codec as an OGG file.

HTML5 browsers and their supported audio file formats:

- Firefox (OSX, Win): WEBMA, OGA
- Safari (OSX, Win): MP3, M4A
- Mobile Safari iOS4 (iPad, iPhone, iPod): MP3, M4A
- Opera (OSX, Win): WEBMA, OGA
- Chrome (OSX, Win): WEBMA, OGA, MP3, M4A
- IE9 (Win): MP3, M4A (Can install the WebM codec.)

HTML5 Video Format

The video WebM (VP8) codec is superior to the video OGG (Theora) codec.

HTML5 browsers and their supported video file formats:

- Firefox (OSX, Win): WEBMV, OGV
- Safari (OSX, Win): M4V
- Mobile Safari iOS4 (iPad, iPhone, iPod): M4V
- Opera (OSX, Win): WEBMA, OGV
- Chrome (OSX, Win): WEBMV, OGV, M4V. (Will drop support for M4V soon.)
- IE9 (Win): M4V (Can install the WebM codec.)

HTML5 Audio Streams

HTML5 browsers and their support for audio streams. (Note that, jPlayer's Flash fall-back for non-HTML5 browsers works with MP3 streams.)

Audio streams work on:

- Firefox (OSX, Win): OGA
- Safari (OSX): MP3
- Mobile Safari (iOS4 iPad/iPhone/iPod): MP3
- Opera (OSX, Win): OGA
- Chrome (OSX, Win): MP3, OGA
- IE9 (Win): MP3

Audio streams fail on:

- Safari (Win): MP3

External Resources

Some links to resources about HTML5 video and audio support:

- diveintohtml5.org : Video on the Web.
- html5test.com : HTML5 Test - Where you can test your browser's HTML5 support.
- www.trygve-lie.com : HTML5 audio element and streaming

jPlayer Community

Over 5000 members and growing!

your email

Hire Us!

Need a media based solution realized or just need some help. Hire Happyworm!
Contact: hire.us@happyworm.com.

Help us improve jPlayer

Developing and supporting jPlayer is almost a full-time job and we are really just beginning. Help us continue to help you.

Donate

Plugin corner

- Wordpress
- Drupal
- more...

Also by Happyworm:

Qwiiz

Massively multiplayer real-time quiz game. Cross platform and tuned for the iPad. Try it!

 - Last update: 5th June 2013

follow us on twitter