jquery download audio file

HTML5 Audio & Video for jQuery.

Developing and supporting jPlayer is almost a full-time job. Help us continue to help you.

Please or buy a theme over at.

Thank you to all those that have contributed!

Latest Release.

jPlayer is on GitHub.

jQuery Audio & Video player plugin.

Download the latest release from GitHub Clone or fork the jPlayer Repository on GitHub Install the jPlayer Node Packaged Module for node.js using npm install jplayer The jPlayer Composer Package at Packagist The jPlayer CDN at cdnjs.com (The examples are in the GitHub repository.) For support, use the jPlayer Google Group Licensed under the MIT license.

Development on GitHub.

jPlayer uses the version system of Major.Minor.Patch , where all versions are tagged and available on GitHub under releases. The jPlayer master branch and the current Minor version branch will be identical as patches are applied to the project. Feature development is tracked in the dev branch. Significant patches and new features will cause an update to the Minor version. Minor versions have the feature development branch merged with the patches.

The jPlayer Google Group will provide extra information on the current state of development, where a sticky thread with a title matching the current Minor release will give more details.

Release Archive.

The release archive is maintained in the jPlayer Repository on GitHub.

Below is a record of the release dates of the major and minor versions of jPlayer.

jQuery Audio & Video player plugin.

jPlayer 2.9.0 : 27th November 2014 jPlayer 2.8.0 : 11th November 2014 jPlayer 2.7.0 : 1st September 2014 jPlayer 2.6.0 : 2nd April 2014 jPlayer 2.5.0 : 7th November 2013 jPlayer 2.4.0 : 5th June 2013 jPlayer 2.3.0 : 20th April 2013 † jPlayer 2.2.0 : 13th September 2012 † jPlayer 2.1.0 : 1st September 2011 † jPlayer 2.0.0 : 20th December 2010 †

† Warning: A security vulnerability in the Flash SWF enabled Cross Site Scripting (XSS).

jQuery Audio player plugin.

jPlayer 1.2.0 : 11th July 2010 † jPlayer 1.1.1 : 29th April 2010 † jPlayer 1.1.0 : 26th March 2010 † jPlayer 1.0.0 : 18th February 2010 † jPlayer 0.2.5 : 25th August 2009 † jPlayer 0.2.4 : 1st July 2009 † jPlayer 0.2.3 : 22nd June 2009 † jPlayer 0.2.2 : 20th May 2009 † jPlayer 0.2.1 : 4th May 2009 †

† Warning: A security vulnerability in the Flash SWF enabled Cross Site Scripting (XSS).

jPlayer Community.

7000 members and growing!

jPlayer Newsletter.

New React jPlayer.

Stop the press! The jPlayer community has stepped up to the plate once again and created a React version of jPlayer.

You might also want to check out the following repos: react-jPlayer-examples and react-jPlaylist all available on jPlayer's GitHub account.

Visit.

For new and exciting skins and themes!

Hire Us!

Need a media based solution realized or just need some help. Hire Happyworm! Contact: .

Help us improve jPlayer.

Developing and supporting jPlayer is almost a full-time job. Help us continue to help you.

Rendering and playing audio files with HTML5 and Jquery.

Playing audio files in a rails app is very simple. By following just a few steps, you can embed an audio file in your html and use jquery to play those files in response to certain events. We'll be using the example of a simple rails app, amiruby.com, that answers "Yes" or "Nope" to the question of whether or not a site is builtwith ruby. We want our user to hear a chorus of cheers if the answer is "Yes" and a chorus of boos if the answer is "Nope".

1. Add the audio files to your app In the public directory of your rails app, creates an audios subdirectory. This is where we will store cheers.mp3 and boos.mp3 .

2. Embed the audio files in your html document.

We gave our audio tags an id of 'audios' which we'll set to display: none; in our css file. This way the audio player will be hidden from the viewer.
3. Write the javascript.

In brief, submitting a search request on this site sends an ajax post request (using remote: true) to the create method of our Searches controller. The Searches controller renders the create.js.erb view. This view contains the javascript that we want to fire on the completion of that post request.

At the completion of our post request, once we have the result (the "Yes" or "Nope" answer), we want to play the appropriate audio file:

On our index.html.erb, we have a div with an id of "result". This is where we will 'appear' our yes or no response. We've passed in the @result variable returned by the create method of the searches controller and used the .html jquery method to fill in the result div with @result.

Then, we invoke the playAudio function, which is defined below. The playAudio function examines the value of our result div. If the result is "Yes", it used the .trigger("play") jquery method to play the audio file with an id of 'yes-audio'. If it the result is "Nope", we will play the audio file with an id of 'no-audio'.

Javascript Tutorial: Record Audio and Encode it to mp3.

A few weeks ago, I was building a web app using the Google Cloud Voice API and I needed to build a ‘record’ feature in the browser. I discovered Javascript’s MediaRecorder API for recording audio, and the ffmpeg node library for encoding audio.

Here is a link to a CodePen with all the front end code I am covering — https://codepen.io/jeremyagottfried/pen/bMqyNZ. I linked to it instead of embedding because embedded CodePen doesn’t allow you to ask for microphone permission.

If you’re looking to understand how to build a simple web app capable of recording audio and save it to an mp3 file, here is a tutorial:

1. Create a Button.

First, create two html buttons for starting and stopping the recording.

You can set your buttons to be rounded in css using the border-radius attribute.

For my record button, I set my css to border-radius: 50%; , max-width:50%; , max-height: 15%; , and background-color: red; .

2. Create Event Listeners For Your Buttons.

I created two event listeners — one for my start button and one for my stop button. The record.onclick method sets up an event listener for clicks on my record button. The stopRecord.onclick sets up an event listener for the stopRecord button.

Explanation of Event Handlers:

console.log("I was clicked") — set up logs to make sure button clicks are triggering the event handler. record.disabled = true — after clicking on the recording button, disable it so you can’t keep clicking on it during recording. After you click on stopRecord , re-enable the record button with record.disabled = false . record.style.backgroundColor = 'blue' — after the record button is clicked, change the color of the record button to signal that you are currently recording. audioChunks = [] — create an array to hold your audio binary data. rec.start() — begin recording. let rec = new MediaRecorder(stream); MediaRecorder is a built in class in javascript that allows you to save streamed media data inside an object. Call rec.start() to start recording the audio stream and rec.stop() to stop recording the audio stream.

3. Begin Streaming Audio.

The Javascript Web API MediaDevices provides access to connected media devices such as microphones and cameras. getUserMedia API handles streaming from users’ microphones and asking them for permission to use their microphone. In the arguments of getUserMedia() , you pass an object specifying which type of media you’d like to stream. Here I specified audio:true.

Once the media device is streaming, .then(stream =>) will give you access to the stream and execute a callback function handlerFunction() .

4. Handle The Audio Stream.

Explanation of handlerFunction()

rec = new MediaRecorder(stream); — this is where we define the rec variable that we referenced earlier with rec.start() . MediaRecorder is a built in class in javascript that allows you to save streamed media data inside an object. rec.ondataavailable = e => <> — wait for the streamed audio data to become available, then execute a callback to handle the data. let blob = new Blob(audioChunks,); — Blob is a built in javascript class for storing binary data. By default, javascript's getUserMedia() API streams audio data in binary form. Specifying audio/mpeg-3 will allow audio playback in our browser, but it will not encode the audio to a usable format for persisting the audio file. sendData(blob) — execute a function to send your blob to your server. You can do this via jquery's ajax function, javascript's new XMLHttpRequest() , or fetch .

Add A Playback Feature.

1. In order to add a playback feature, you need to create an audio HTML element with id=recordedAudio. Our handlerFunction will fill the recordedAudio element with audio data so you can play back the recording in the browser.

2. recordedAudio.src = URL.createObjectURL(blob) — associate you blob with a URL and set it to the src for your audio element.

3. recordedAudio.controls = true — provide controls for the user to start and stop the recorded audio.

4. recordedAudio.autoplay = true — automatically play back the audio when the user stops recording. You can set this to false if you don't want the audio to play back automatically.

Encode An mp3 File Using ffmpeg.

After you send your blob binary file to the backend, you can use node's ffmpeg library to convert your file to another format.

First, run npm install ffmpeg , then copy the code below.

fnExtractSoundToMP3() is a method built into the ffmpeg class for converting audio files to mp3. This method also allows you to extract audio from a video file. This code will convert your binary blob file to mp3. There are two lines you need to change —

path/to/blob/file , which is the path to your binary audio file.

path/to/new/file.mp3 , which is where your new file will be created.

The rest of the code sets up logging for errors.

Now enjoy and customize your fully functional recording app! You can copy the front end code from the CodePen below. Remember it will not work in this embedded version because embedded CodePen doesn't allow you to ask for microphone permission.

Simple Clean HTML5 Audio Player With jQuery.

A minimal, clean, jQuery based HTML5 audio player plugin which allows you to split the audio into several chapters (great for long songs and audiobooks). Licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

How to use it:

1. Link to jQuery library and the simple audio player's files:

2. Embed a sound element into your document.

3. Call the function on the audio tag and specify the player title & chapters as follows:

This awesome jQuery plugin is developed by dradl. For more Advanced Usages, please check the demo page or visit the official website.

Customizable HTML5 Audio Player - jQuery mb.miniAudioPlayer.

A jQuery plugin that transforms an audio link into a customizable, skinnable, minimal-looking HTML5 audio player on the page.

More Features:

Fallbacks to a flash player on legacy browsers. 5 pre-built skins or create your own themes. Autoplay and auto pause on blur. Infinite loop. Show/hide controls as per your needs. Easy to download audio files with a single click.

How to use it:

1. Load the jQuery mb.miniAudioPlayer plugin's files in the document.

2. Create an audio link on the page and config the audio player by passing the options object to the class attribute as follows:

3. Initialize the plugin on the audio link and we're done.

4. All possible plugin options to customize the audio player.

5. Callback functions.

6. Available API methods to control the audio player programmatically.

7. Create your own themes.

This awesome jQuery plugin is developed by pupunzi. For more Advanced Usages, please check the demo page or visit the official website.