
Interactive Image Documentation

Release 2.7.1

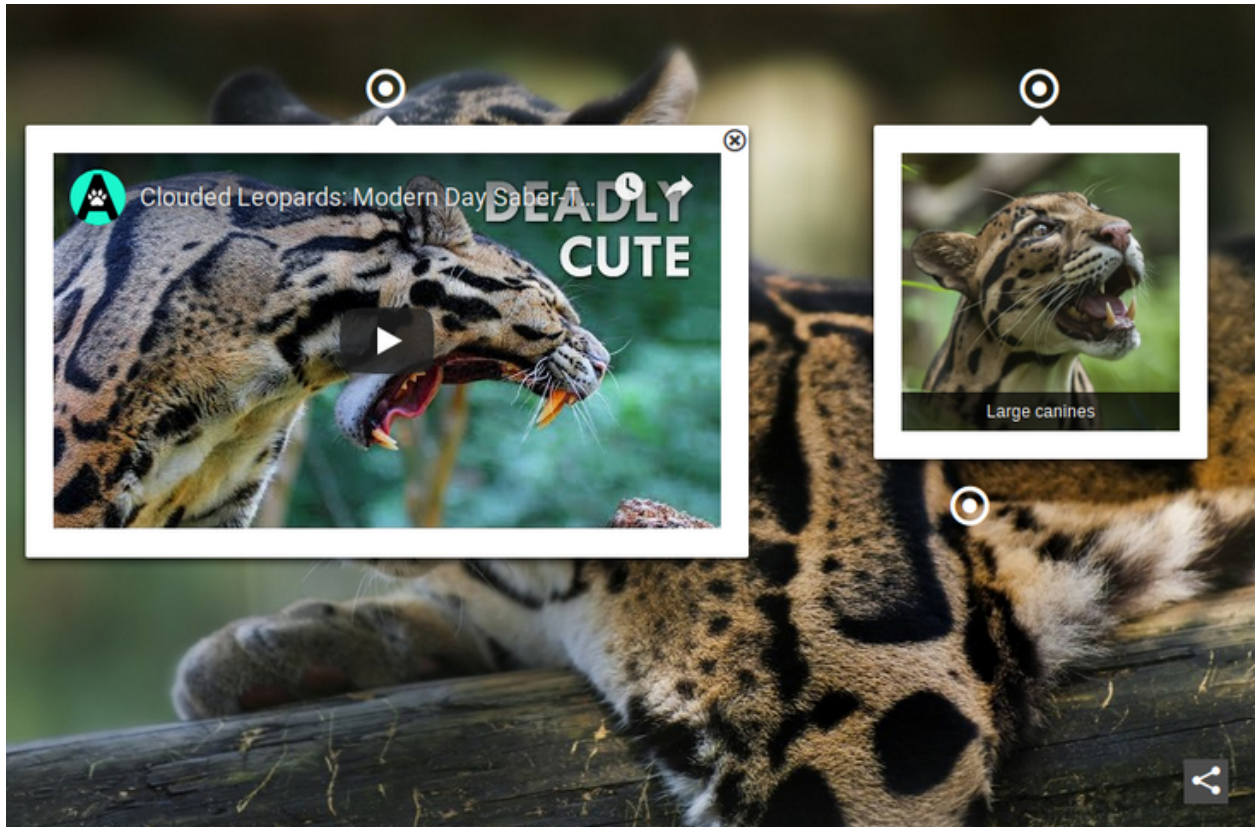
Jean-Philippe Chateau

Jul 31, 2020

1	Installation	3
1.1	Package managers	3
1.2	Content Delivery Network	3
1.3	Download	4
2	Basic usage	5
3	Allow HTML markup	9
4	Trigger Event	11
5	Social Media Share	13
6	Sticky items	15
7	Style customization	17
8	Destroying and recreating an instance	19
9	I18n	21
10	Debugging	23
11	Plugin configuration options	25
11.1	Options	25
11.2	SocialMedia	25
12	Items	27
12.1	Text Item	27
12.2	Picture Item	28
12.3	Audio Item	28
12.4	Video Item	28
12.5	Provider Item	29
13	Other objects	31
13.1	Position	31
13.2	Link	31
13.3	Parameters	31

14	Requirements	33
15	Dependencies	35
16	Tests	37

Interactive Image is a jQuery plugin that eases the creation of media experiences.



[Online Demo | CodePen](#)

This documentation is organized into the following sections:

CHAPTER 1

Installation

1.1 Package managers

Install Interactive Image via a package manager.

npm

```
$ npm install --save interactiveimagejs
```

Yarn

```
$ yarn add interactiveimagejs
```

1.2 Content Delivery Network

You can also include Interactive Image from a CDN:

jsDelivr

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/interactiveimagejs@2.7.1/
↪dist/interactive-image.min.css" />

<script src="https://cdn.jsdelivr.net/npm/interactiveimagejs@2.7.1/dist/interactive-
↪image.min.js"></script>
```

View Interactive Image at jsDelivr.

unpkg

```
<link rel="stylesheet" href="https://unpkg.com/interactiveimagejs@2.7.1/dist/
↪interactive-image.min.css" />

<script src="https://unpkg.com/interactiveimagejs@2.7.1/dist/interactive-image.min.js
↪"></script>
```

(continues on next page)

(continued from previous page)

[View unpkg.](#)

1.3 Download

Or download the production / development files from GitHub.

All the files (.js, .css and fonts) are located in the `dist` directory.

The latest Zip archive is available at [Github](#), and you can browse all the releases [here](#).

CHAPTER 2

Basic usage

Edit the source code of your web page:

HTML

```
<head>
  <!-- Include Interactive Image jQuery plugin Styles -->
  <link rel="stylesheet" href="interactive-image.min.css" />

  <!-- Specific styles of a scene -->
  <style>
    .interactive-image {
      width: 900px;
      height: 600px;
      background: url("/path/to/main-image.png");
    }
  </style>
</head>
<body>
  <!-- Main container of a scene -->
  <div id="my-interactive-image"></div>

  <!-- Include jQuery -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></
↪script>

  <!-- Include Interactive Image jQuery plugin JavaScript -->
  <script src="interactive-image.min.js"></script>
</body>
```

Note regarding the CSS file: You may have to rewrite the relative paths of the fonts.

JavaScript

This code block describes the 5 types of items: a simple text, a picture, a sound, a video, and a content provided by a tier.

```
// Items collection
var items = [
  {
    type: "text",
    title: "Fur",
    description: "The fur of clouded leopards is...",
    position: {
      left: 100,
      top: 50
    }
  },
  {
    type: "picture",
    path: "/path/to/picture.png",
    caption: "A baby clouded leopard",
    position: {
      left: 200,
      top: 300
    }
  },
  {
    type: "audio",
    path: "/path/to/sound.mp3",
    caption: "A clouded leopard growl",
    position: {
      left: 300,
      top: 500
    }
  },
  {
    type: "video",
    path: "/path/to/video.mp4",
    caption: "A clouded leopard walking",
    poster: "/path/to/poster.png",
    position: {
      left: 400,
      top: 550
    }
  },
  {
    type: "provider",
    providerName: "youtube",
    parameters: {
      videoId: "iPRiQ6SBntQ"
    },
    position: {
      left: 600,
      top: 550
    },
    sticky: true
  }
];

// Plugin activation
$(document).ready(function() {
  $("#my-interactive-image").interactiveImage(items);
});
```

You can see a full working example including all types of items in the `examples` directory or a simple example at [CodePen](#).

CHAPTER 3

Allow HTML markup

In order to enable HTML markup in descriptions, you have to set the `allowHtml` flag to `true` in the options before activating the plugin.

```
var options = {  
  allowHtml: true  
};  
  
$("#my-interactive-image").interactiveImage(items, options);
```


CHAPTER 4

Trigger Event

There are two ways to trigger the display of an item corresponding to a hotspot. When the hotspot is clicked or when it is hovered. Default behavior is `hover`. You can override this behavior via the options.

```
var options = {
  triggerEvent: 'click'
};

$("#my-interactive-image").interactiveImage(items, options);
```

Social Media Share

By default, a social media share box is displayed at the bottom right of the scene.
You can prevent this behavior before activating the plugin:

```
var options = {  
  shareBox: false  
};  
  
$("#my-interactive-image").interactiveImage(items, options);
```

You can also customize the social media share properties of your Interactive Image:

```
var options = {  
  socialMedia: {  
    url: "https://www.jpchateau.com/demo/interactive-image",  
    text: "Clouded Leopard",  
    hashtags: ["jQuery", "cloudedLeopard"],  
    twitterUsername: "my_twitter_account",  
  }  
}  
  
$("#my-interactive-image").interactiveImage(items, options);
```

Supported social media: email, [Twitter](#) and [Facebook](#).

In case of email share, the subject of the email is the webpage title.

CHAPTER 6

Sticky items

You can make some items having a sticky behavior.

Add the `sticky` flag and set it to `true` for each `Item` objects you do not want to be hidden when the mouse leaves it.

```
// Item object
{
  //...
  sticky: true
}
```

Style customization

You may want to change the rendering of a specific item.
Fill the `customClassName` property:

```
// Item object
{
  //...
  customClassName: "my-custom-css-class"
}
```

You may also want to change the global rendering of all items, as for example the background and the front color of text items.

Adapt this CSS snippet to your needs and add it after the `interactive-image` CSS file is loaded:

```
.interactive-image .text-item {
  background-color: blue;
  color: yellow;
}
```

Available Item classes:

- `text-item`
- `picture-item`
- `audio-item`
- `video-item`
- `provider-item`

Destroying and recreating an instance

Sometimes you might need to destroy an instance and recreate it.

```
// jQuery native function to remove all attached event handlers
$('#my-interactive-image').off();

// Do your own stuff (resizing image, updating positions values of icons, ...)

// Create a new instance
$("#my-interactive-image").interactiveImage(items, options);
```


Only one message can be translated.

“Please rotate your device.” appears when the width of the user’s screen is less than 320 pixels (smartphones portrait) to prevent bad user experience.

You can override this text using CSS after the Interactive Image stylesheet.

```
.interactive-image .unsupported-screen::before {  
  content: 'Localized message.'  
}
```


CHAPTER 10

Debugging

In order to display some console messages to see the different steps of the processing and the time each one has taken, you can enable the debug mode before activating the plugin:

```
var options = {  
  debug: true  
};  
  
$("#my-interactive-image").interactiveImage(items, options);
```

Plugin configuration options

11.1 Options

Property	Type	Example	Re- quired	De- fault	Purpose
allowHtml	boolean	true	No	false	Enable HTML markup in descriptions
debug	boolean	true	No	false	Enable logs in console
shareBox	boolean	false	No	true	Enable social media share
socialMe- dia	object	See <code>SocialMedia</code> ob- ject	No	–	Social media configuration
trig- gerEvent	string	“clickIhover”	No	‘hover’	Mouse event to trigger the display of an item

11.2 SocialMedia

Property	Type	Example	Required	Default	Purpose
url	string	“ http://www.example.com ”	No	Document URL	URL to share
text	string	“Text”	No	Page title	Text
hashtags	array	[“jQuery”, “cloudedLeopard”]	No	–	Hashtags
twitterUsername	string	“my_twitter_account”	No	–	Twitter account

You do not need to prefix your Twitter account by @.

CHAPTER 12

Items

Each item has several possibilities of configuration. You can add a link and/or a picture to your `text` items, or a caption to your `picture`, `audio` or `video` items.

12.1 Text Item

Property	Type	Example	Re-quired	Default	Purpose
<code>type</code>	string	“text”	Yes	–	Item type
<code>position</code>	object	See <code>Position</code> object	No	{left:0, top:0}	Hotspot position on the scene
<code>title</code>	string	“My title”	Yes	–	Title
<code>description</code>	string	“My description”	Yes	–	Descriptive text
<code>picturePath</code>	string	“/path/to/picture.png”	No	–	Illustration source path
<code>link</code>	object	See <code>Link</code> object	No	–	HTTP Link
<code>sticky</code>	boolean	true	No	false	Sticky behavior
<code>customClass-Name</code>	string	“my-css-class”	No	–	Custom CSS class of the item

12.2 Picture Item

Property	Type	Example	Re- quired	Default	Purpose
type	string	“picture”	Yes	–	Item type
position	ob- ject	See <code>Position</code> ob- ject	No	{left:0, top:0}	Hotspot position on the scene
path	string	“/path/to/picture.png”		–	Image source path
customClass- Name	string	“my-css-class”	No	–	Custom CSS class of the item

12.3 Audio Item

Supported audio formats: MP3, Ogg, WAVE.

For more information about browser support of audio formats, please refer to [Media formats for HTML audio and video](#).

Property	Type	Example	Re- quired	Default	Purpose
type	string	“audio”	Yes	–	Item type
position	object	See <code>Position</code> ob- ject	No	{left:0, top:0}	Hotspot position on the scene
path	string	“/path/to/sound.mp3”	Yes	–	Sound source path
caption	string	“My caption”	No	–	Sound short description
sticky	boolean	true	No	false	Sticky behavior
customClass- Name	string	“my-css-class”	No	–	Custom CSS class of the item

12.4 Video Item

Supported video formats: MP4, WebM.

For more information about browser support of video formats, please refer to [Media formats for HTML audio and video](#).

Property	Type	Example	Re- quired	Default	Purpose
type	string	“video”	Yes	–	Item type
position	object	See <code>Position</code> object	No	{left:0, top:0}	Hotspot position on the scene
path	string	“/path/to/video.mp4”	Yes	–	Video source path
caption	string	“My caption”	No	–	Video short description
poster	string	“path/to/poster.png”	No	–	Image to display while the video is downloading
sticky	boolean	true	No	false	Sticky behavior
customClass-Name	string	“my-css-class”	No	–	Custom CSS class of the item

12.5 Provider Item

Content providers supported: [Dailymotion](#), [SoundCloud](#), [Vimeo](#) and [Youtube](#).

Property	Type	Example	Re- quired	Default	Purpose
type	string	“provider”	Yes	–	Item type
position	object	See <code>Position</code> object	No	{left:0, top:0}	Hotspot position on the scene
providerName	string	“dailymo- tion soundcloud vimeo youtube”	Yes	–	Content provider name
parameters	object	See <code>Parameters</code> object	Yes	–	Content provider parameters
sticky	boolean	true	No	false	Sticky behavior
customClass-Name	string	“my-css-class”	No	–	Custom CSS class of the item

CHAPTER 13

Other objects

13.1 Position

Property	Type	Example	Required	Default	Purpose
left	integer	200	Yes	–	X absolute value
top	integer	50	Yes	–	Y absolute value

13.2 Link

Property	Type	Example	Required	Default	Purpose
url	string	“ https://www.github.com ”	Yes	–	href attribute
label	string	“My webpage”	No	url value	Name of the webpage

13.3 Parameters

Property	Type	Example	Required	Default	Purpose
soundId	string	“123456789”	No	–	Sound identifier (SoundCloud)
videoId	string	“xxxYYY123”	No	–	Video identifier (Dailymotion, Vimeo, Youtube)

Dailymotion, Vimeo and Youtube videos are supported.
SoundCloud tracks are supported.

CHAPTER 14

Requirements

jQuery 1.7.2+ is required.

Interactive Image is tested against these other versions of jQuery to ensure a good compatibility:

- 1.12.4
- 2.2.4
- 3.x

CHAPTER 15

Dependencies

1 dependency comes with Interactive Image package:

- `imagesloaded` to detect when images have been loaded

CHAPTER 16

Tests

All builds are unit tested with [Mocha](#) and [Chai](#).
Check it out on [Travis CI](#)!

Run all unit tests

```
$ npm run test
```

Run all unit tests with code coverage

```
$ npm run test-with-coverage
```