**jquery.magnific-popup.js初始化及内容类型**

lightbox弹窗特效框架

**1.依赖：**

*<!-- Magnific Popup core CSS file -->*

<link rel="stylesheet" href="magnific-popup/magnific-popup.css">

*<!-- jQuery 1.7.2+ or Zepto.js 1.0+ -->*

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>

*<!-- Magnific Popup core JS file -->*

<script src="magnific-popup/jquery.magnific-popup.js"></script>

**2.popup初始化**

popup的初始化代码应该在文档加载完成时执行

$(document).ready(**function**() {

$('.image-link').magnificPopup({type**:**'image'});

});

下面是三个初始化popup的方法：

1） 来自 HTML element

<a class="test-popup-link" href="path-to-image.jpg">Open popup</a>

$('.test-popup-link').magnificPopup({

type**:** 'image'

*// other options*

});

2.）来自 a group of elements with one parent

Same as first one, but use this method if you are creating a popup from a list of elements in one container. Note that **this method does not enable gallery mode**, it just reduces the number of click event handlers; each item will be opened as a single popup. **If you wish to enable gallery, add the gallery:{enabled:true}**option.

<div class="parent-container">

<a href="path-to-image-1.jpg">Open popup 1</a>

<a href="path-to-image-2.jpg">Open popup 2</a>

<a href="path-to-image-3.jpg">Open popup 3</a>

</div>

$('.parent-container').magnificPopup({

delegate**:** 'a', *// child items selector, by clicking on it popup will open*

type**:** 'image'

*// other options*

});

3）来自 the ‘items’ option

The items option **defines data for the popup item(s)** and **makes Magnific Popup ignore all attributes on the target DOM element.** The value for items can be a single object or an array of objects.

items选项初始化的方法，将定义popup的item数据，且使得popup忽略目标DOM element的自有属性。

*// Example with single object*

$('#some-button').magnificPopup({

items**:** {

src**:** 'path-to-image-1.jpg'

},

type**:** 'image' *// this is default type*

});

*// Example with multiple objects*

$('#some-button').magnificPopup({

items**:** [

{

src**:** 'path-to-image-1.jpg'

},

{

src**:** 'http://vimeo.com/123123',

type**:** 'iframe' *// this overrides default type*

},

{

src**:** $('<div>Dynamically created element</div>'), *// Dynamically created element*

type**:** 'inline'

},

{

src**:** '<div>HTML string</div>',

type**:** 'inline'

},

{

src**:** '#my-popup', *// CSS selector of an element on page that should be used as a popup*

type**:** 'inline'

}

],

gallery**:** {

enabled**:** **true**

},

type**:** 'image' *// this is default type*

});

**3.内容类型**

**Content Types**

**四大类型：**

By default, Magnific Popup has four types of content: image, iframe, inline, and ajax. There is no any “auto-detection” of type based on URL, so you should define it manually.

**内容类型的两种声明方式：**

The type of a popup can be defined in a two ways:

1. Using the type option. E.g.: $('.image-link').magnificPopup({type:'image'}).
2. Using the mfp-TYPE CSS class (where TYPE is the desired content type). For example: <a class="mfp-image image-link">Open image</a>, $('.image-link').magnificPopup().

The second option always overrides the first, so you may initialize popups with multiple content types from one call.

inline is the default content type (from v0.8.4), so you may skip its definition.

inline是默认内容类型，使用它时可以忽略定义

**内容源的几种声明方式：**

**The source of the the popup content** (for example, a path to an image, a path to an HTML file, a path to a video page) can be defined in a few ways:

Method #1: From the href attribute:

<a href="image-for-popup.jpg">Open image</a>

Method #2: From the data-mfp-src attribute (overrides the first method):

<a href="some-image.jpg" data-mfp-src="image-for-popup.jpg">Open image</a>

Method #3: From the items option

$.magnificPopup.open({

items**:** {

src**:** 'some-image.jpg'

},

type**:** 'image'

});

If you want to modify how the source is parsed, you may hook into the elementParse callback. For example:

$('.image-link').magnificPopup({

type**:**'image',

callbacks**:** {

elementParse**:** **function**(item) {

*// Function will fire for each target element*

*// "item.el" is a target DOM element (if present)*

*// "item.src" is a source that you may modify*

console.log(item); *// Do whatever you want with "item" object*

}

}

});

下面就是四大内容类型：

## 4.图片类型Image Type

The path to the image must be set as the main source if you selected this type. If your popup doesn’t have an image source and doesn’t have an image that shouldn’t be preloaded (and retina-ized, etc.), use the inlinetype.

image**:** {

markup**:** '<div class="mfp-figure">'**+**

'<div class="mfp-close"></div>'**+**

'<div class="mfp-img"></div>'**+**

'<div class="mfp-bottom-bar">'**+**

'<div class="mfp-title"></div>'**+**

'<div class="mfp-counter"></div>'**+**

'</div>'**+**

'</div>', *// Popup HTML markup. `.mfp-img` div will be replaced with img tag, `.mfp-close` by close button*

cursor**:** 'mfp-zoom-out-cur', *// Class that adds zoom cursor, will be added to body. Set to null to disable zoom out cursor.*

titleSrc**:** 'title', *// Attribute of the target element that contains caption for the slide.*

*// Or the function that should return the title. For example:*

*// titleSrc: function(item) {*

*// return item.el.attr('title') + '<small>by Marsel Van Oosten</small>';*

*// }*

verticalFit**:** **true**, *// Fits image in area vertically*

tError**:** '<a href="%url%">The image</a> could not be loaded.' *// Error message*

}

## 5.框架类型Iframe Type

By default Magnific Popup supports only one type of URL for each service:

*// YouTube*

"http://www.youtube.com/watch?v=7HKoqNJtMTQ"

*// Vimeo*

"http://vimeo.com/123123"

*// Google Maps*

"https://maps.google.com/maps?q=221B+Baker+Street,+London,+United+Kingdom&hl=en&t=v&hnear=221B+Baker+St,+London+NW1+6XE,+United+Kingdom"

But you can extend it and make it support absolutely any URL or any other service (view [example that adds Dailymotion support](http://codepen.io/dimsemenov/pen/jnohA)). Iframe options:

iframe**:** {

markup**:** '<div class="mfp-iframe-scaler">'**+**

'<div class="mfp-close"></div>'**+**

'<iframe class="mfp-iframe" frameborder="0" allowfullscreen></iframe>'**+**

'</div>', *// HTML markup of popup, `mfp-close` will be replaced by the close button*

patterns**:** {

youtube**:** {

index**:** 'youtube.com/', *// String that detects type of video (in this case YouTube). Simply via url.indexOf(index).*

id**:** 'v=', *// String that splits URL in a two parts, second part should be %id%*

*// Or null - full URL will be returned*

*// Or a function that should return %id%, for example:*

*// id: function(url) { return 'parsed id'; }*

src**:** '//www.youtube.com/embed/%id%?autoplay=1' *// URL that will be set as a source for iframe.*

},

vimeo**:** {

index**:** 'vimeo.com/',

id**:** '/',

src**:** '//player.vimeo.com/video/%id%?autoplay=1'

},

gmaps**:** {

index**:** '//maps.google.',

src**:** '%id%&output=embed'

}

*// you may add here more sources*

},

srcAction**:** 'iframe\_src', *// Templating object key. First part defines CSS selector, second attribute. "iframe\_src" means: find "iframe" and set attribute "src".*

}

## 6.行内类型Inline Type

To create popup from inline element you need to:

**1)** Create a HTML element that you wish to display in popup and add it somewhere. Class mfp-hide is required to hide the popup from the page.

创建一个你希望在popup上展示且随处可添加的的**HTML元素**

使用mtf-hide类在页面隐藏popup

<div id="test-popup" class="white-popup mfp-hide">

Popup content

</div>

**2)** Style this element. Magnific Popup by default doesn’t apply any styles to it, except vertical centering (if alignTop:false). Close button will be automatically appended inside (if closeBtnInside:true).

popup默认不添加样式，除了垂直在中心的选项（默认是false）

close按钮会自动添加在popup中

**.white-popup** {

**position:** **relative**;

**background:** #FFF;

**padding:** 20px;

**width:** **auto**;

**max-width:** 500px;

**margin:** 20px **auto**;

}

3) Add button that will open the popup (source must match CSS id of an element (#test-popup in our case).

*<!-- Like so: -->*

<a href="#test-popup" class="open-popup-link">Show inline popup</a>

*<!-- Or like so: -->*

<a href="mobile-friendly-page.html" data-mfp-src="#test-popup" class="open-popup-link">Show inline popup</a>

4) Initialize script.

$('.open-popup-link').magnificPopup({

type**:**'inline',

midClick**:** **true** *// Allow opening popup on middle mouse click. Always set it to true if you don't provide alternative source in href.*

});

Here are some other ways to initialize popup:

*// From HTML string*

$('button').magnificPopup({

items**:** {

src**:** '<div class="white-popup">Dynamically created popup</div>',

type**:** 'inline'

}

});

*// From an element with ID #popup*

$('button').magnificPopup({

items**:** {

src**:** '#popup',

type**:** 'inline'

}

});

*// From jQuery object*

$('button').magnificPopup({

items**:** {

src**:** $('<div class="white-popup">Dynamically created popup</div>'),

type**:** 'inline'

}

});

*// Open directly via API*

$.magnificPopup.open({

items**:** {

src**:** '<div class="white-popup">Dynamically created popup</div>', *// can be a HTML string, jQuery object, or CSS selector*

type**:** 'inline'

}

});

## 7 .Ajax Type

To create such type of popup, first of **define the path to the file** that you wish to display and select ajax type of the popup. Popup itself should be styled in exactly the same way as an [inline popup type](http://dimsemenov.com/plugins/magnific-popup/documentation.html#inline-type).

**Important note!** The contents of the file that you load is already a popup itself, so there must be **only one root element**.

<a href="path-to-file.html" class="ajax-popup-link">Show inline popup</a>

$('.ajax-popup-link').magnificPopup({

type**:** 'ajax'

});

Note that path to the file that will be loaded should have the same origin (e.g. be on the same domain), [learn more](http://stackoverflow.com/questions/3076414/ways-to-circumvent-the-same-origin-policy).

Ajax options:

ajax**:** {

settings**:** **null**, *// Ajax settings object that will extend default one - http://api.jquery.com/jQuery.ajax/#jQuery-ajax-settings*

*// For example:*

*// settings: {cache:false, async:false}*

cursor**:** 'mfp-ajax-cur', *// CSS class that will be added to body during the loading (adds "progress" cursor)*

tError**:** '<a href="%url%">The content</a> could not be loaded.' *// Error message, can contain %curr% and %total% tags if gallery is enabled*

}

To modify content after it’s loaded, or to select and show just specific element from loaded file, there is a parseAjax callback:

callbacks**:** {

parseAjax**:** **function**(mfpResponse) {

*// mfpResponse.data is a "data" object from ajax "success" callback*

*// for simple HTML file, it will be just String*

*// You may modify it to change contents of the popup*

*// For example, to show just #some-element:*

*// mfpResponse.data = $(mfpResponse.data).find('#some-element');*

*// mfpResponse.data must be a String or a DOM (jQuery) element*

console.log('Ajax content loaded:', mfpResponse);

},

ajaxContentAdded**:** **function**() {

*// Ajax content is loaded and appended to DOM*

console.log(**this**.content);

}

}