# Cropper一个简单方便的图片裁剪插件

https://segmentfault.com/a/1190000012344970

**插件介绍**

这是一个我在写以前的项目的途中发现的一个国人写的jQuery图像裁剪插件，当时想实现用户资料的头像上传功能，并且能够预览图片，和对图片进行简单的裁剪、旋转，花了不少时间才看到了这个插件，感觉功能挺全面，代码实现起来也挺简单，再加上用的是Bootstrap，对移动端操作也有适配，于是就用了。现在稍微有点时间就记录一下，方便以后再用的时候查阅。另外也有对应的js版本。

**官方文档（英文）**

* jQuery
  + [GitHub项目地址](https://github.com/fengyuanchen/cropper)
  + [官方示例](https://fengyuanchen.github.io/cropper/)
* js
  + [GitHub项目地址](https://github.com/fengyuanchen/cropperjs)

**兼容性**

兼容所有支持了Canvas的浏览器（IE9+），一小部分功能例外，具体请查看官方文档。

**参数**

**viewMode**

* Type: Number
* Default: 0
* Options: 0,1,2,3

这个具体每个值对应的效果我也不是很清楚，推荐在上面的官方示例里都试一试，我都是比较喜欢2。

**dragMode**

* Type: String
* Default: 'crop'
* Options:
  + 'crop'： 在裁剪框外拖动鼠标会生成一个新的裁剪框。
  + 'move'： 在裁剪框外拖动鼠标会移动原图。
  + 'none'： 在裁剪框外拖动鼠标则什么也不做。

**aspectRatio**

* Type: Number
* Default: NaN

这个是裁剪框的纵横比，默认是不限制的。例如1:1的头像就写1,16:9可写成16 / 9。

**data**

* Type: Object
* Default: null

The previous cropped data if you had stored, will be passed to setData method automatically.

（没怎么用过，都是直接用setData方法）

**preview**

* Type: String (**jQuery selector**)
* Default: ''

预览图的位置，用jQuery选择器表示。

**responsive**

* Type: Boolean
* Default: true

在更改窗口大小后是否重新渲染cropper。

**restore**

* Type: Boolean
* Default: true

在更改窗口大小后是否恢复裁剪区域。

**checkCrossOrigin**

* Type: Boolean
* Default: true

检查图像是否是跨域图像。（具体查看官方文档）

**checkOrientation**

* Type: Boolean
* Default: true

（具体查看官方文档）

**modal**

* Type: Boolean
* Default: true

非裁剪区域是否用黑罩遮盖。

**guides**

* Type: Boolean
* Default: true

裁剪区域是否显示虚线。

**center**

* Type: Boolean
* Default: true

裁剪区域正中央是否显示**+**号。

**highlight**

* Type: Boolean
* Default: true

裁剪区域是否高亮显示。

**background**

* Type: Boolean
* Default: true

是否显示背景的黑白方格（类似PS里透明图层的显示方式）。

**autoCrop**

* Type: Boolean
* Default: true

cropper初始化完成后是否自动显示裁剪框

**autoCropArea**

* Type: Number
* Default: 0.8 (80% of the image)

自动显示的裁剪框的大小。因此，数字应当在0~1之间。

**movable**

* Type: Boolean
* Default: true

是否允许移动原图。（如果这里填false那么尽管**dragMode**的值是move，在裁剪框外拖动也不会移动原图）

**rotatable**

* Type: Boolean
* Default: true

是否可以旋转原图。

**scalable**

* Type: Boolean
* Default: true

是否可以对原图进行纵横拉伸。

例如把原图宽度拉长为原来的2倍或者拉长为原来的-1倍（即水平翻转）。

**zoomable**

* Type: Boolean
* Default: true

是否可以对原图进行缩小放大。

**zoomOnTouch**

* Type: Boolean
* Default: true

是否允许在移动端上使用双指触摸缩放原图。

**zoomOnWheel**

* Type: Boolean
* Default: true

是否允许使用鼠标滚轮缩放原图。

**wheelZoomRatio**

* Type: Number
* Default: 0.1

当使用鼠标滚轮缩放时的比例。

**cropBoxMovable**

* Type: Boolean
* Default: true

是否允许移动裁剪框。

**cropBoxResizable**

* Type: Boolean
* Default: true

是否允许通过拖动裁剪框的边框来调整裁剪框的大小。

**toggleDragModeOnDblclick**

* Type: Boolean
* Default: true

是否允许通过双击来在crop和move之间切换**dragMode**。

**minContainerWidth**

* Type: Number
* Default: 200

容器宽度最小值。

**minContainerHeight**

* Type: Number
* Default: 100

容器高度最小值。

**minCanvasWidth**

* Type: Number
* Default: 0

canvas（原图）宽度最小值。

**minCanvasHeight**

* Type: Number
* Default: 0

canvas（原图）高度最小值。

**minCropBoxWidth**

* Type: Number
* Default: 0

剪切框宽度最小值。

**Note:** This size is relative to the page, not the image.

**minCropBoxHeight**

* Type: Number
* Default: 0

剪切框高度最小值。

**Note:** This size is relative to the page, not the image.

**ready**

* Type: Function
* Default: null

A shortcut of the "ready" event.

**cropstart**

* Type: Function
* Default: null

A shortcut of the "cropstart" event.

**cropmove**

* Type: Function
* Default: null

A shortcut of the "cropmove" event.

**cropend**

* Type: Function
* Default: null

A shortcut of the "cropend" event.

**crop**

* Type: Function
* Default: null

A shortcut of the "crop" event.

**zoom**

* Type: Function
* Default: null

A shortcut of the "zoom" event.

**常用方法**

除了"setAspectRatio","replace"和"destroy"以外，所有的方法都要在ready后才能使用。这里只介绍几个常用的方法，全部的方法请到官方文档查阅。

方法的使用格式为

$().cropper('method',arg0,arg1,arg2,...);

**crop()**

手动显示裁剪框。

$().cropper({

autoCrop: false,

ready: **function** () {

*// Do something here*

*// ...*

*// And then*

$(**this**).cropper('crop');

}

});

**reset()**

恢复全部到初始状态。

**replace(url[, onlyColorChanged])**

* **url**:
  + Type: String
  + A new image url.
* **onlyColorChanged** (optional):
  + Type: Boolean
  + If only change the color, not the size, then the cropper only need to change the srcs of all related images, not need to rebuild the cropper. This can be used for applying filters.
  + If not present, its default value is false.

替换cropper中的图像文件，通常第二个参数不管。

**destroy()**

销毁cropper，并且会移除img标签的src属性的值。

**getCroppedCanvas([options])**

* **options** (optional):
  + Type: Object
  + Properties:
    - width: the destination width of the output canvas.
    - height: the destination height of the output canvas.
    - minWidth: the minimum destination width of the output canvas, the default value is 0.
    - minHeight: the minimum destination height of the output canvas, the default value is 0.
    - maxWidth: the maximum destination width of the output canvas, the default value is Infinity.
    - maxHeight: the maximum destination height of the output canvas, the default value is Infinity.
    - fillColor: a color to fill any alpha values in the output canvas, the default value is transparent.
    - [imageSmoothingEnabled](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/imageSmoothingEnabled): set to change if images are smoothed (true, default) or not (false).
    - [imageSmoothingQuality](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/imageSmoothingQuality): set the quality of image smoothing, one of "low" (default), "medium", or "high".
* (return value):
  + Type: HTMLCanvasElement
  + A canvas drawn the cropped image.
* Notes:
  + 输出的canvas的纵横比会自动适应于裁剪框的纵横比.
  + 如果打算得到JPEG图像，那么应该先设置fillColor参数，否则裁剪后的透明部分默认会由黑色填充。
* Browser support:
  + Basic image: requires [Canvas](http://caniuse.com/canvas) support (IE 9+).
  + Rotated image: requires [CSS3 2D Transforms](http://caniuse.com/transforms2d) support (IE 9+).
  + Cross-origin image: requires HTML5 [CORS settings attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_settings_attributes) support (IE 11+).

得到裁剪到的图像的canvas，如果没有裁剪，那么就返回的是整个原图图像的canvas。

这是最重要的一个方法，通过这个方法就可以得到裁剪后的图像，再使用toDataURL()得到base64 dataURL（不指定格式的话会是png格式）或者toBlob()得到Blob,然后就可以很轻松地将图片上传至服务器上或者显示在某个img标签中了。例如：

*// 转换为png格式的dataURL*

**var** dataURL = $().cropper('getCroppedCanvas', {

width:100,

height:100

}).toDataURL('image/png');

*// 转换为Blob后显示在img标签中*

**var** URL = window.URL || window.webkitURL;

$().cropper('getCroppedCanvas', {

width:100,

height:100

}).toBlob(**function** (blob) {

$().attr('src',URL.createObjectURL(blob));

});

**简单实例**

**在页面直接使用cropper**

接下来只是实现一个简单的功能：网页中可以上传图片，然后对图片进行裁剪，点击确定后会显示出裁剪后的图片。

代码如下：

**<!DOCTYPE html>**

<html lang="zh-cn">

<head>

<meta charset="UTF-8">

<title>裁剪图片</title>

<link href="https://cdn.bootcss.com/cropper/3.1.3/cropper.min.css" rel="stylesheet">

<link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

<style>

.row{

margin-bottom: 5px;

}

**#photo** {

max-width: 100%;

}

.img-preview {

width: 100px;

height: 100px;

overflow: hidden;

}

**button** {

margin-top:10px;

}

**#result** {

width: 150px;

height: 150px;

}

</style>

</head>

<body>

<div class="container">

<div class="row">

<div class="col-sm-12 text-center">

<label for="input" class="btn btn-danger" id="">

<span>选择图片</span>

<input type="file" id="input" class="sr-only">

</label>

</div>

</div>

<div class="row">

<div class="col-sm-6 col-sm-offset-2">

<img src="" id="photo">

</div>

<div class="col-sm-2">

<div>

<p>

预览(100\*100)：

</p>

<div class="img-preview">

</div>

</div>

<button class="btn btn-primary" onclick="crop()">裁剪图片</button>

<div>

<br/>

<p>

结果：

</p>

<img src="" alt="裁剪结果" id="result">

</div>

</div>

</div>

</div>

*<!-- Scripts -->*

<script src="https://cdn.bootcss.com/jquery/3.2.1/jquery.min.js"></script>

<script src="https://cdn.bootcss.com/cropper/3.1.3/cropper.min.js"></script>

<script src="https://cdn.bootcss.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script>

*// 修改自官方demo的js*

**var** initCropper = **function** (img, input){

**var** $image = img;

**var** options = {

aspectRatio: 1, *// 纵横比*

viewMode: 2,

preview: '.img-preview' *// 预览图的class名*

};

$image.cropper(options);

**var** $inputImage = input;

**var** uploadedImageURL;

**if** (URL) {

*// 给input添加监听*

$inputImage.change(**function** () {

**var** files = **this**.files;

**var** file;

**if** (!$image.data('cropper')) {

**return**;

}

**if** (files && files.length) {

file = files[0];

*// 判断是否是图像文件*

**if** (/^image\/\w+$/.test(file.type)) {

*// 如果URL已存在就先释放*

**if** (uploadedImageURL) {

URL.revokeObjectURL(uploadedImageURL);

}

uploadedImageURL = URL.createObjectURL(file);

*// 销毁cropper后更改src属性再重新创建cropper*

$image.cropper('destroy').attr('src', uploadedImageURL).cropper(options);

$inputImage.val('');

} **else** {

window.alert('请选择一个图像文件！');

}

}

});

} **else** {

$inputImage.prop('disabled', true).addClass('disabled');

}

}

**var** crop = **function**(){

**var** $image = $('#photo');

**var** $target = $('#result');

$image.cropper('getCroppedCanvas',{

width:300, *// 裁剪后的长宽*

height:300

}).toBlob(**function**(blob){

*// 裁剪后将图片放到指定标签*

$target.attr('src', URL.createObjectURL(blob));

});

}

$(**function**(){

initCropper($('#photo'),$('#input'));

});

</script>

</body>

</html>



**在bootstrap模态框中使用cropper**

虽然在模态框中可以像上面一样使用cropper，甚至我以前写的项目也是跟上面一样，但是这次整理的时候突然发现了一个bug：当隐藏模态框后调整浏览器大小（甚至按f12），再打开模态框后cropper的容器会改变，导致难以使用。于是，我在GitHub中翻找了issue，在官方的example中找到了对应的解决方法。但其实这个解决方法也是一种暴力解法，即模态框隐藏后销毁cropper，打开后重新创建cropper，可能会有别的方法，因为不确定会不会有别的bug，所以暂时还是用官方的方法比较好。

代码如下：

**<!DOCTYPE html>**

<html lang="zh-cn">

<head>

<meta charset="UTF-8">

<title>上传头像</title>

<link href="https://cdn.bootcss.com/cropper/3.1.3/cropper.min.css" rel="stylesheet">

<link href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

<style type="text/css">

body{

text-align: center;

}

#user-photo {

width:300px;

height:300px;

margin-top: 10px;

}

#photo {

max-width:100%;

max-height:350px;

}

.img-preview-box {

text-align: center;

}

.img-preview-box > div {

display: inline-block;;

margin-right: 10px;

}

.img-preview {

overflow: hidden;

}

.img-preview-box .img-preview-lg {

width: 150px;

height: 150px;

}

.img-preview-box .img-preview-md {

width: 100px;

height: 100px;

}

.img-preview-box .img-preview-sm {

width: 50px;

height: 50px;

border-radius: 50%;

}

</style>

</head>

<body>

<button class="btn btn-primary" data-target="#changeModal" data-toggle="modal">打开</button><br/>

<div class="user-photo-box">

<img id="user-photo" src="">

</div>

</div>

<div class="modal fade" id="changeModal" tabindex="-1" role="dialog" aria-hidden="true">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title text-primary">

<i class="fa fa-pencil"></i>

更换头像

</h4>

</div>

<div class="modal-body">

<p class="tip-info text-center">

未选择图片

</p>

<div class="img-container hidden">

<img src="" alt="" id="photo">

</div>

<div class="img-preview-box hidden">

<hr>

<span>150\*150:</span>

<div class="img-preview img-preview-lg">

</div>

<span>100\*100:</span>

<div class="img-preview img-preview-md">

</div>

<span>30\*30:</span>

<div class="img-preview img-preview-sm">

</div>

</div>

</div>

<div class="modal-footer">

<label class="btn btn-danger pull-left" for="photoInput">

<input type="file" class="sr-only" id="photoInput" accept="image/\*">

<span>打开图片</span>

</label>

<button class="btn btn-primary disabled" disabled="true" onclick="sendPhoto();">提交</button>

<button class="btn btn-close" aria-hidden="true" data-dismiss="modal">取消</button>

</div>

</div>

</div>

</div>

<script src="https://cdn.bootcss.com/jquery/3.2.1/jquery.min.js"></script>

<script src="https://cdn.bootcss.com/cropper/3.1.3/cropper.min.js"></script>

<script src="https://cdn.bootcss.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script type="text/javascript">

**var** initCropperInModal = **function**(img, input, modal){

**var** $image = img;

**var** $inputImage = input;

**var** $modal = modal;

**var** options = {

aspectRatio: 1, *// 纵横比*

viewMode: 2,

preview: '.img-preview' *// 预览图的class名*

};

*// 模态框隐藏后需要保存的数据对象*

**var** saveData = {};

**var** URL = window.URL || window.webkitURL;

**var** blobURL;

$modal.on('show.bs.modal',**function** () {

*// 如果打开模态框时没有选择文件就点击“打开图片”按钮*

**if**(!$inputImage.val()){

$inputImage.click();

}

}).on('shown.bs.modal', **function** () {

*// 重新创建*

$image.cropper( $.extend(options, {

ready: **function** () {

*// 当剪切界面就绪后，恢复数据*

**if**(saveData.canvasData){

$image.cropper('setCanvasData', saveData.canvasData);

$image.cropper('setCropBoxData', saveData.cropBoxData);

}

}

}));

}).on('hidden.bs.modal', **function** () {

*// 保存相关数据*

saveData.cropBoxData = $image.cropper('getCropBoxData');

saveData.canvasData = $image.cropper('getCanvasData');

*// 销毁并将图片保存在img标签*

$image.cropper('destroy').attr('src',blobURL);

});

**if** (URL) {

$inputImage.change(**function**() {

**var** files = **this**.files;

**var** file;

**if** (!$image.data('cropper')) {

**return**;

}

**if** (files && files.length) {

file = files[0];

**if** (/^image\/\w+$/.test(file.type)) {

**if**(blobURL) {

URL.revokeObjectURL(blobURL);

}

blobURL = URL.createObjectURL(file);

*// 重置cropper，将图像替换*

$image.cropper('reset').cropper('replace', blobURL);

*// 选择文件后，显示和隐藏相关内容*

$('.img-container').removeClass('hidden');

$('.img-preview-box').removeClass('hidden');

$('#changeModal .disabled').removeAttr('disabled').removeClass('disabled');

$('#changeModal .tip-info').addClass('hidden');

} **else** {

window.alert('请选择一个图像文件！');

}

}

});

} **else** {

$inputImage.prop('disabled', true).addClass('disabled');

}

}

**var** sendPhoto = **function**(){

$('#photo').cropper('getCroppedCanvas',{

width:300,

height:300

}).toBlob(**function**(blob){

*// 转化为blob后更改src属性，隐藏模态框*

$('#user-photo').attr('src',URL.createObjectURL(blob));

$('#changeModal').modal('hide');

});

}

$(**function**(){

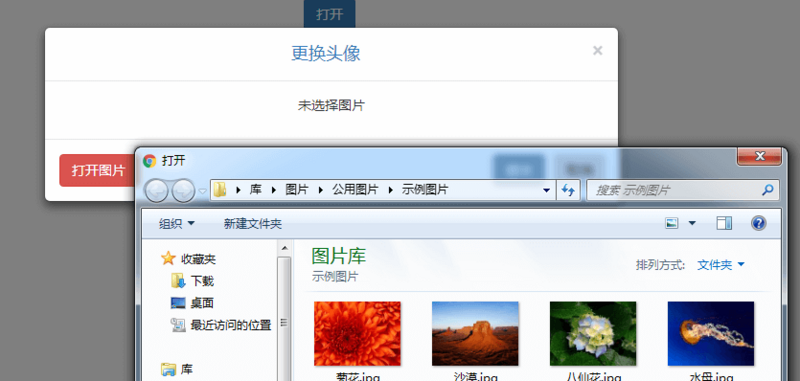
initCropperInModal($('#photo'),$('#photoInput'),$('#changeModal'));

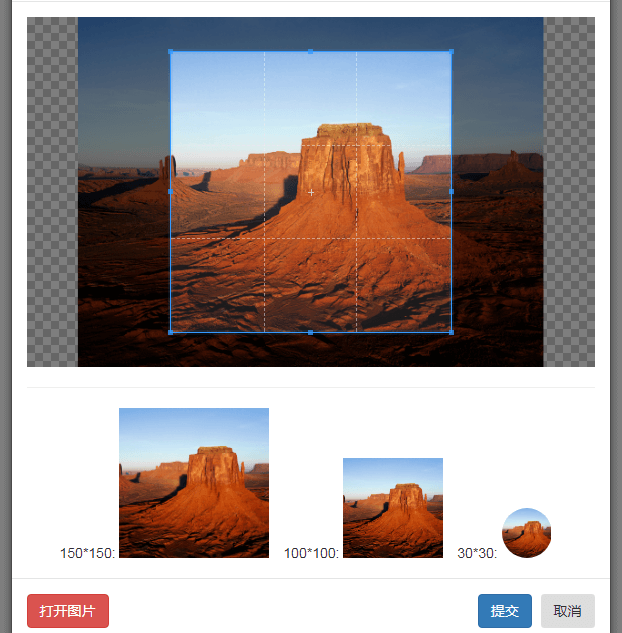
});

</script>

</body>

</html>





**使用cropper来上传图片到服务器**

由于cropper可以得到两种裁剪后图片的数据（即blob和dataURL），所以对应的上传到后台也会有两种方法，在这里我只写一种使用ajax上传base64 dataURL的，另一种方法如果有兴趣，可以自己尝试。

页面中，将上面的sendPhoto方法改为：

**var** sendPhoto = **function** () {

*// 得到PNG格式的dataURL*

**var** photo = $('#photo').cropper('getCroppedCanvas', {

width: 300,

height: 300

}).toDataURL('image/png');

$.ajax({

url: '上传地址', *// 要上传的地址*

type: 'post',

data: {

'imgData': photo

},

dataType: 'json',

success: **function** (data) {

**if** (data.status == 0) {

*// 将上传的头像的地址填入，为保证不载入缓存加个随机数*

$('.user-photo').attr('src', '头像地址?t=' + Math.random());

$('#changeModal').modal('hide');

} **else** {

alert(data.info);

}

}

});

}

后台中，Java的主要代码如下：（使用了jdk8的Base64,，如果是低版本请自行替换）

*/\*\**

*\* 将Base64位编码的图片进行解码，并保存到指定目录*

*\*/*

**public** **static** **void** **decodeBase64DataURLToImage**(String dataURL, String path, String imgName) **throws** IOException {

*// 将dataURL开头的非base64字符删除*

String base64 = dataURL.substring(dataURL.indexOf(",") + 1);

FileOutputStream write = **new** FileOutputStream(**new** File(path + imgName));

**byte**[] decoderBytes = Base64.getDecoder().decode(base64);

write.write(decoderBytes);

write.close();

}

**小结**

cropper能做到的事情还很多，这里只是简单使用了一下，更多功能可以在有想法的再研究下。

这是针对以前项目用的cropper的一个整理，结果因为当初没有看官方例子，途中发现了在模态框中使用的一个bug，以后会注意这方面。另外，整理这部分资料时也参考了不少的网络资料，在这里就不一一记录了。