[下载插件](http://down.htmleaf.com/1504/201504212053.zip)

 (5)

简要教程

cropper是一款使用简单且功能强大的图片剪裁[jQuery插件](http://www.htmleaf.com/jQuery/)。该图片剪裁插件支持图片放大缩小，支持鼠标滚轮操作，支持图片旋转，支持触摸屏设备，支持[canvas](http://www.htmleaf.com/html5/html5-canvas/)，并且支持跨浏览器使用。

cropper提供了大量的参数、方法和事件供图片的剪裁操作。

 安装

可以通过[Bower](http://bower.io/)或[NPM](http://npmjs.org/)来安装该插件。

|  |
| --- |
| **bower install cropper**  **npm install cropper** |

 使用方法

使用该图片剪裁插件首先要引入必要的js和css文件。

|  |
| --- |
| **<script src="/path/to/jquery.js"></script><!-- jQuery is required -->**  **<link  href="/path/to/cropper.css" rel="stylesheet">**  **<script src="/path/to/cropper.js"></script>** |

**HTML结构**

可以将图片或canvas直接包裹到一个块级元素中。

|  |
| --- |
| **<!-- Wrap the image or canvas with a block element -->**  **<div class="container">**  **<img src="picture.jpg">**  **</div>** |

**调用插件**

可以使用$.fn.cropper方法来初始化该图片剪裁插件。

|  |
| --- |
| **$('.container > img').cropper({**  **aspectRatio: 16 / 9,**  **crop: function(data) {**  **// Output the result data for cropping image.**  **}**  **});** |

**注意事项**：

* 注意：剪裁区域的尺寸继承自图片的父容器（包裹容器），所以要确保包裹图片的是一个可见的块级元素。
* 输出的剪裁数据基于原始的图片尺寸，你可以使用这些数据直接剪裁图片。
* 如果你要使用跨源图片来作为剪裁图片，请确保你的浏览器支持HTML5 [CORS settings attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_settings_attributes)，并且你的图片服务器支持Access-Control-Allow-Origin属性。

 配置参数

你可以通过$().cropper(options)方法来设置参数。如果你想改变全局默认参数，可以使用$.fn.cropper.setDefaults(options)方法。

* aspectRatio：类型：Number，默认值NaN。设置剪裁容器的比例。
* crop：类型：Function，默认值null。当改变剪裁容器或图片时的事件函数。
* preview：类型：String(jQuery选择器)，默认值''。添加额外的元素（容器）的预览。注意：
  + 最大宽度是剪裁容器的初始化宽度
  + 最大高度是剪裁容器的初始化高度
  + 如果你设置了aspectRatio参数，确保预览容器具有相同的比例
* strict：类型：Boolean，默认值true。在strict模式中，canvas不能小于容器，剪裁容器不能再canvas之外。
* responsive：类型：Boolean，默认值true。是否在窗口尺寸改变的时候重置cropper。
* checkImageOrigin：类型：Boolean，默认值true。默认情况下，插件会检测图片的源，如果是跨域图片，图片元素会被添加crossOrigin class，并会为图片的url添加一个时间戳来使getCroppedCanvas变为可用。添加时间戳会使图片重新加载，以使跨域图片能够使用getCroppedCanvas。在图片上添加crossOrigin class会阻止在图片url上添加时间戳，及图片的重新加载。
* background：类型：Boolean，默认值true。是否在容器上显示网格背景。
* modal：类型：Boolean，默认值true。是否在剪裁框上显示黑色的模态窗口。
* guides：类型：Boolean，默认值true。是否在剪裁框上显示虚线。
* highlight：类型：Boolean，默认值true。是否在剪裁框上显示白色的模态窗口。
* autoCrop：类型：Boolean，默认值true。是否在初始化时允许自动剪裁图片。
* autoCropArea：类型：Number，默认值0.8（图片的80%）。0-1之间的数值，定义自动剪裁区域的大小。
* dragCrop：类型：Boolean，默认值true。是否允许移除当前的剪裁框，并通过拖动来新建一个剪裁框区域。
* movable：类型：Boolean，默认值true。是否允许移动剪裁框。
* resizable：类型：Boolean，默认值true。是否允许改变剪裁框的大小。
* zoomable：类型：Boolean，默认值true。是否允许放大缩小图片。
* mouseWheelZoom：类型：Boolean，默认值true。是否允许通过鼠标滚轮来缩放图片。
* touchDragZoom：类型：Boolean，默认值true。是否允许通过触摸移动来缩放图片。
* rotatable：类型：Boolean，默认值true。是否允许旋转图片。
* minContainerWidth：类型：Number，默认值200。容器的最小宽度。
* minContainerHeight：类型：Number，默认值100。容器的最小高度。
* minCanvasWidth：类型：Number，默认值0。canvas 的最小宽度（image wrapper）。
* minCanvasHeight：类型：Number，默认值0。canvas 的最小高度（image wrapper）。
* build：类型：Function，默认值null。build.cropper事件的简写方式。
* built：类型：Function，默认值null。built.cropper事件的简写方式。
* dragstart：类型：Function，默认值null。dragstart.cropper事件的简写方式。
* dragmove：类型：Function，默认值null。dragmove.cropper事件的简写方式。
* dragend：类型：Function，默认值null。dragend.cropper事件的简写方式。
* zoomin：类型：Function，默认值null。zoomin.cropper事件的简写方式。
* zoomout：类型：Function，默认值null。zoomout.cropper事件的简写方式。

 方法

因为图片是异步加载的，所以你需要在built之后才能调用下面的方法，setAspectRatio、replace和destroy方法例外。

|  |
| --- |
| **$().cropper({**  **built: function () {**  **$().cropper('method', argument1, , argument2, ..., argumentN)**  **}**  **}** |

* move(offsetX, offsetY)：
  + offsetX：类型：Number，水平方向上移动的大小，单位像素。
  + offsetY：类型：Number，垂直方向上移动的大小，单位像素。

移动一幅图片：

|  |
| --- |
| **$().cropper('move', 1, 0)**  **$().cropper('move', 0, -1)** |

* zoom(ratio)：

ratio：

* + 类型：Number
  + Zoom in：需要一个正数（ratio > 0）
  + Zoom out：需要一个负数（ratio < 0）

缩放一幅图片：

|  |
| --- |
| **$().cropper('zoom', 0.1)**  **$().cropper('zoom', -0.1)** |

* rotate(degree)：

ratio：

* + 类型：Number
  + Rotate right：需要一个正数（degree > 0）
  + Rotate left：需要一个负数（degree < 0）

旋转一幅图片，需要[CSS3 Transforms3d](http://caniuse.com/transforms3d)的支持（IE10+）：

|  |
| --- |
| **$().cropper('rotate', 90)**  **$().cropper('rotate', -90)** |

* enable()：使cropper可用。
* disable()：冻结cropper。
* reset()：重置剪裁区域的图片到初始状态。
* clear()：清空剪裁区域。
* replace(url)：

url：

* + 类型：String
  + 一个新的图片URL

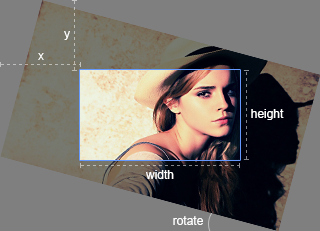
替换图片的URL重建cropper。

* getData()：

返回值：

* + 类型：Object
  + 属性：
    - x：剪裁区域左侧的偏移。
    - y：剪裁区域距上部的偏移。
    - width：剪裁区域的宽度。
    - height：剪裁区域的高度。
    - rotate：图片的旋转角度。

看下图演示从原始图片中返回的剪裁区域数据。



* getContainerData()：

返回值：

* + 类型：Object
  + 属性：
    - width：容器的当前宽度。
    - height：容器的当前高度。

输出容器尺寸数据。

* getImageData()：

返回值：

* + 类型：Object
  + 属性：
    - left：图片的左侧偏移。
    - top：图片的上部偏移。
    - width：图片的宽度。
    - height：图片的高度。

输出图片的位置和尺寸大小。

* getCanvasData()：

返回值：

* + 类型：Object
  + 属性：
    - left：canvas的左侧偏移。
    - top：canvas的上部偏移。
    - width：canvas的宽度。
    - height：canvas的高度。

输出canvas（图片容器）的位置和尺寸大小。

* setCanvasData(data)：

返回值：

* + 类型：Object
  + 属性：
    - left：canvas的距离左侧新的偏移。
    - top：canvas的距离上部新的偏移。
    - width：canvas新的宽度。
    - height：canvas新的高度。

改变canvas（图片容器）的位置和尺寸大小。

* getCropBoxData()：

返回值：

* + 类型：Object
  + 属性：
    - left：剪裁区域距离左侧的偏移。
    - top：剪裁区域距离上部的偏移。
    - width：剪裁区域的宽度。
    - height：剪裁区域的高度。

输出剪裁区域的位置和尺寸大小。

* setCropBoxData(data)：

返回值：

* + 类型：Object
  + 属性：
    - left：剪裁区域距离左侧新的偏移。
    - top：剪裁区域距离上部新的偏移。
    - width：剪裁区域新的宽度。
    - height：剪裁区域新的高度。

改变剪裁区域的位置和尺寸大小。

* getCroppedCanvas([options])：

参数（默认值）：

* + 类型：Object
  + 属性：
    - width：输出的canvas目标宽度。
    - height：输出的canvas目标高度。
    - fillColor：canvas中填充的颜色。

返回值：

* + 类型：HTMLCanvasElement
  + 一个画有剪裁图片的canvas。

浏览器支持：

* + 基本图像：需要[Canvas](http://caniuse.com/canvas)的支持（IE9+）
  + 旋转图像：需要[CSS3 animations](http://www.htmleaf.com/css3/animation/) [Transforms3d](http://caniuse.com/transforms3d)的支持（IE10+）
  + 跨域图像：需要[HTML5](http://www.htmleaf.com/html5/) [CORS settings attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/CORS_settings_attributes)的支持（IE11+）

获取canvas绘制的剪裁图像。

在这之后，你可以直接将canvas作为图片显示，或使用[canvas.toDataURL](https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/toDataURL)方法获取图像的数据链接，或者使用[canvas.toBlob](https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/toBlob)方法获取一个blob，并通过[FormData](https://developer.mozilla.org/en/XMLHttpRequest/FormData)方法将它更新到服务器上（如果浏览器支持这些API）。

|  |
| --- |
| **$().cropper('getCroppedCanvas')**  **$().cropper('getCroppedCanvas', {**  **width: 160,**  **height: 90**  **});** |

* setAspectRatio(aspectRatio)：

aspectRatio：

* + 类型：Number
  + 需要一个正整数。

改变剪裁区域的比例。

* setDragMode([mode])：

mode（可选项）：

* + 类型：String
  + 默认值：''
  + 选项：'crop'或'move'

改变拖拽模式。提示：你可以通过双击来改变"crop" 和 "move"模式。

 事件

 build.cropper：当cropper对象开始加载图片的时候触发该事件。

 built.cropper：当cropper对象构建完成时触发该事件。

 dragstart.cropper：

event.dragType：

* "crop"：创建一个新的剪裁区域。
* "move"：移动canvas。
* "zoom"：通过触摸移动来缩放canvas。
* "e"：调整剪裁区域东边的尺寸大小。
* "w"：调整剪裁区域西边的尺寸大小。
* "s"：调整剪裁区域南边的尺寸大小。
* "n"：调整剪裁区域北边的尺寸大小。
* "se"：调整剪裁区域东南边的尺寸大小。
* "sw"：调整剪裁区域西南边的尺寸大小。
* "ne"：调整剪裁区域东北边的尺寸大小。
* "nw"：调整剪裁区域西北边的尺寸大小。
* "all"：移动剪裁区域。

该事件在剪裁区域发生改变时触发。

相关的原始事件：mousedown、touchstart。

|  |
| --- |
| **$('img').on('dragstart.cropper', function (e) {**  **console.log(e.type); // dragstart**  **console.log(e.namespace); // cropper**  **console.log(e.dragType); // ...**  **});** |

 dragmove.cropper：

event.dragType：和dragstart.cropper相同。

当剪裁区域发生改变时触发。

相关的原始事件：mousemove、touchmove。

 dragend.cropper：

event.dragType：和dragstart.cropper相同。

当剪裁区域改变结束时触发。

相关的原始事件：mouseup、mouseleave、touchend、touchleave和touchcancel。

 zoomin.cropper：当cropper对象开始放大canvas时触发。

 zoomout.cropper：当cropper对象开始缩小canvas时触发。

 No conflict：如果你使用了相同名称空间的其它插件，可以通过$.fn.cropper.noConflict方法来修改名称空间。

|  |
| --- |
| **<script src="other-plugin.js"></script>**  **<script src="cropper.js"></script>**  **<script>**  **$.fn.cropper.noConflict();**  **// Code that uses other plugin's "$().cropper" can follow here.**  **</script>** |

 浏览器兼容

* Chrome 38+
* Firefox 33+
* Internet Explorer 8+
* Opera 25+
* Safari 5.1+

作为[jQuery插件](http://www.htmleaf.com/jQuery/)，你还需要了解[浏览器对jQuery的支持](http://jquery.com/browser-support/)。

本文版权属于jQuery之家，转载请注明出处：<http://www.htmleaf.com/jQuery/Image-Effects/201504211716.html>