vue-cropper.vue  
  
<template> <div class="vue-cropper" ref="cropper" @mouseover="scaleImg" @mouseout="cancelScale"> <div class="cropper-box" v-if="imgs"> <div class="cropper-box-canvas" v-show="!loading" :style="{'width': trueWidth + 'px','height': trueHeight + 'px','transform': 'scale(' + scale + ',' + scale + ') ' + 'translate3d('+ x / scale + 'px,' + y / scale + 'px,' + '0)'+ 'rotateZ('+ rotate \* 90 +'deg)'}" > <img :src="imgs" alt="cropper-img" ref="cropperImg"> </div> </div> <div class="cropper-drag-box" :class="{'cropper-move': move && !crop, 'cropper-crop': crop, 'cropper-modal': cropping}" @mousedown="startMove" @touchstart="startMove" ></div> <div v-show="cropping" class="cropper-crop-box" :style="{'width': cropW + 'px','height': cropH + 'px','transform': 'translate3d('+ cropOffsertX + 'px,' + cropOffsertY + 'px,' + '0)'}" > <span class="cropper-view-box"> <img :style="{'width': trueWidth + 'px','height': trueHeight + 'px','transform': 'scale(' + scale + ',' + scale + ') ' + 'translate3d('+ (x - cropOffsertX) / scale + 'px,' + (y - cropOffsertY) / scale + 'px,' + '0)'+ 'rotateZ('+ rotate \* 90 +'deg)'}" :src="imgs" alt="cropper-img" > </span> <span class="cropper-face cropper-move" @mousedown="cropMove" @touchstart="cropMove"></span> <span class="crop-info" v-if="info" :style="{'top': cropInfo.top}" >{{ cropInfo.width }} × {{ cropInfo.height }}</span> <span v-if="!fixedBox"> <span class="crop-line line-w" @mousedown="changeCropSize($event, false, true, 0, 1)" @touchstart="changeCropSize($event, false, true, 0, 1)" ></span> <span class="crop-line line-a" @mousedown="changeCropSize($event, true, false, 1, 0)" @touchstart="changeCropSize($event, true, false, 1, 0)" ></span> <span class="crop-line line-s" @mousedown="changeCropSize($event, false, true, 0, 2)" @touchstart="changeCropSize($event, false, true, 0, 2)" ></span> <span class="crop-line line-d" @mousedown="changeCropSize($event, true, false, 2, 0)" @touchstart="changeCropSize($event, true, false, 2, 0)" ></span> <span class="crop-point point1" @mousedown="changeCropSize($event, true, true, 1, 1)" @touchstart="changeCropSize($event, true, true, 1, 1)" ></span> <span class="crop-point point2" @mousedown="changeCropSize($event, false, true, 0, 1)" @touchstart="changeCropSize($event, false, true, 0, 1)" ></span> <span class="crop-point point3" @mousedown="changeCropSize($event, true, true, 2, 1)" @touchstart="changeCropSize($event, true, true, 2, 1)" ></span> <span class="crop-point point4" @mousedown="changeCropSize($event, true, false, 1, 0)" @touchstart="changeCropSize($event, true, false, 1, 0)" ></span> <span class="crop-point point5" @mousedown="changeCropSize($event, true, false, 2, 0)" @touchstart="changeCropSize($event, true, false, 2, 0)" ></span> <span class="crop-point point6" @mousedown="changeCropSize($event, true, true, 1, 2)" @touchstart="changeCropSize($event, true, true, 1, 2)" ></span> <span class="crop-point point7" @mousedown="changeCropSize($event, false, true, 0, 2)" @touchstart="changeCropSize($event, false, true, 0, 2)" ></span> <span class="crop-point point8" @mousedown="changeCropSize($event, true, true, 2, 2)" @touchstart="changeCropSize($event, true, true, 2, 2)" ></span> </span> </div> </div></template><script>import { defineComponent } from 'vue'import exifmin from "./exif-js-min";export default defineComponent({ data: function() { return { w: 0, h: 0, scale: 1, x: 0, y: 0, loading: true, trueWidth: 0, trueHeight: 0, move: true, moveX: 0, moveY: 0, crop: false, cropping: false, cropW: 0, cropH: 0, cropOldW: 0, cropOldH: 0, canChangeX: false, canChangeY: false, changeCropTypeX: 1, changeCropTypeY: 1, cropX: 0, cropY: 0, cropChangeX: 0, cropChangeY: 0, cropOffsertX: 0, cropOffsertY: 0, support: "", touches: [], touchNow: false, rotate: 0, isIos: false, orientation: 0, imgs: "", coe: 0.2, scaling: false, scalingSet: "", coeStatus: "", isCanShow: true }; }, props: { img: { type: [String, Blob, null, File], default: "" }, outputSize: { type: Number, default: 1 }, outputType: { type: String, default: "jpeg" }, info: { type: Boolean, default: true }, canScale: { type: Boolean, default: true }, autoCrop: { type: Boolean, default: false }, autoCropWidth: { type: [Number, String], default: 0 }, autoCropHeight: { type: [Number, String], default: 0 }, fixed: { type: Boolean, default: false }, fixedNumber: { type: Array, default: () => { return [1, 1]; } }, fixedBox: { type: Boolean, default: false }, full: { type: Boolean, default: false }, canMove: { type: Boolean, default: true }, canMoveBox: { type: Boolean, default: true }, original: { type: Boolean, default: false }, centerBox: { type: Boolean, default: false }, high: { type: Boolean, default: true }, infoTrue: { type: Boolean, default: false }, maxImgSize: { type: [Number, String], default: 2000 }, enlarge: { type: [Number, String], default: 1 }, preW: { type: [Number, String], default: 0 }, 图片布局方式 mode 实现和css背景一样的效果 contain 居中布局 默认不会缩放 保证图片在容器里面 mode: 'contain' cover 拉伸布局 填充整个容器 mode: 'cover' 如果仅有一个数值被给定，这个数值将作为宽度值大小，高度值将被设定为auto。 mode: '50px' 如果有两个数值被给定，第一个将作为宽度值大小，第二个作为高度值大小。 mode: '50px 60px' mode: { type: String, default: "contain" }, limitMinSize: { type: [Number, Array, String], default: () => { return 10; } }, }, computed: { cropInfo() { let obj = {}; obj.top = this.cropOffsertY > 21 ? "-21px" : "0px"; obj.width = this.cropW > 0 ? this.cropW : 0; obj.height = this.cropH > 0 ? this.cropH : 0; if (this.infoTrue) { let dpr = 1; if (this.high && !this.full) { dpr = window.devicePixelRatio; } if ((this.enlarge !== 1) & !this.full) { dpr = Math.abs(Number(this.enlarge)); } obj.width = obj.width \* dpr; obj.height = obj.height \* dpr; if (this.full) { obj.width = obj.width / this.scale; obj.height = obj.height / this.scale; } } obj.width = obj.width.toFixed(0); obj.height = obj.height.toFixed(0); return obj; }, isIE() { var userAgent = navigator.userAgent; //取得浏览器的userAgent字符串 const isIE = !!window.ActiveXObject || 'ActiveXObject' in window; //判断是否IE浏览器 return isIE; }, passive () { return this.isIE ? null : { passive: false } } }, watch: { img() { this.checkedImg(); }, imgs(val) { if (val === "") { return; } this.reload(); }, cropW() { this.showPreview(); }, cropH() { this.showPreview(); }, cropOffsertX() { this.showPreview(); }, cropOffsertY() { this.showPreview(); }, scale(val, oldVal) { this.showPreview(); }, x() { this.showPreview(); }, y() { this.showPreview(); }, autoCrop(val) { if (val) { this.goAutoCrop(); } }, autoCropWidth() { if (this.autoCrop) { this.goAutoCrop(); } }, autoCropHeight() { if (this.autoCrop) { this.goAutoCrop(); } }, mode() { this.checkedImg(); }, rotate() { this.showPreview(); if (this.autoCrop) { this.goAutoCrop(this.cropW, this.cropH); } else { if (this.cropW > 0 || this.cropH > 0) { this.goAutoCrop(this.cropW, this.cropH); } } } }, methods: { getVersion (name) { var arr = navigator.userAgent.split(' '); var chromeVersion = ''; let result = 0; const reg = new RegExp(name, 'i') for(var i=0;i < arr.length;i++){ if(reg.test(arr[i])) chromeVersion = arr[i] } if(chromeVersion){ result = chromeVersion.split('/')[1].split('.'); } else { result = ['0', '0', '0']; } return result }, checkOrientationImage(img, orientation, width, height) { if (this.getVersion('chrome')[0] >= 81) { orientation = -1 } else { if (this.getVersion('safari')[0] >= 605 ) { const safariVersion = this.getVersion('version') if (safariVersion[0] > 13 && safariVersion[1] > 1) { orientation = -1 } } else { const isIos = navigator.userAgent.toLowerCase().match(/cpu iphone os (.\*?) like mac os/) if (isIos) { let version = isIos[1] version = version.split('\_') if (version[0] > 13 || (version[0] >= 13 && version[1] >= 4)) { orientation = -1 } } } } let canvas = document.createElement("canvas"); let ctx = canvas.getContext("2d"); ctx.save(); switch (orientation) { case 2: canvas.width = width; canvas.height = height; ctx.translate(width, 0); ctx.scale(-1, 1); break; case 3: canvas.width = width; canvas.height = height; ctx.translate(width / 2, height / 2); ctx.rotate((180 \* Math.PI) / 180); ctx.translate(-width / 2, -height / 2); break; case 4: canvas.width = width; canvas.height = height; ctx.translate(0, height); ctx.scale(1, -1); break; case 5: canvas.height = width; canvas.width = height; ctx.rotate(0.5 \* Math.PI); ctx.scale(1, -1); break; case 6: canvas.width = height; canvas.height = width; ctx.translate(height / 2, width / 2); ctx.rotate((90 \* Math.PI) / 180); ctx.translate(-width / 2, -height / 2); break; case 7: canvas.height = width; canvas.width = height; ctx.rotate(0.5 \* Math.PI); ctx.translate(width, -height); ctx.scale(-1, 1); break; case 8: canvas.height = width; canvas.width = height; ctx.translate(height / 2, width / 2); ctx.rotate((-90 \* Math.PI) / 180); ctx.translate(-width / 2, -height / 2); break; default: canvas.width = width; canvas.height = height; } ctx.drawImage(img, 0, 0, width, height); ctx.restore(); canvas.toBlob( blob => { let data = URL.createObjectURL(blob); URL.revokeObjectURL(this.imgs) this.imgs = data; }, "image/" + this.outputType, 1 ); }, checkedImg() { if (this.img === null || this.img === '') { this.imgs = '' this.clearCrop() return } this.loading = true; this.scale = 1; this.rotate = 0; this.clearCrop(); let img = new Image(); img.onload = () => { if (this.img === "") { this.$emit("img-load", "error"); return false; } let width = img.width; let height = img.height; exifmin.getData(img).then(data => { this.orientation = data.orientation || 1; let max = Number(this.maxImgSize); if (!this.orientation && (width < max) & (height < max)) { this.imgs = this.img; return; } if (width > max) { height = (height / width) \* max; width = max; } if (height > max) { width = (width / height) \* max; height = max; } this.checkOrientationImage(img, this.orientation, width, height); }); }; img.onerror = () => { this.$emit("img-load", "error"); }; if (this.img.substr(0, 4) !== "data") { img.crossOrigin = ""; } if (this.isIE) { var xhr = new XMLHttpRequest(); xhr.onload = function() { var url = URL.createObjectURL(this.response); img.src = url; }; xhr.open("GET", this.img, true); xhr.responseType = "blob"; xhr.send(); } else { img.src = this.img; } }, startMove(e) { e.preventDefault(); if (this.move && !this.crop) { if (!this.canMove) { return false; } this.moveX = ('clientX' in e ? e.clientX : e.touches[0].clientX) - this.x; this.moveY = ('clientY' in e ? e.clientY : e.touches[0].clientY) - this.y; if (e.touches) { window.addEventListener("touchmove", this.moveImg); window.addEventListener("touchend", this.leaveImg); if (e.touches.length == 2) { this.touches = e.touches; window.addEventListener("touchmove", this.touchScale); window.addEventListener("touchend", this.cancelTouchScale); } } else { window.addEventListener("mousemove", this.moveImg); window.addEventListener("mouseup", this.leaveImg); } this.$emit("imgMoving", { moving: true, axis: this.getImgAxis() }); this.$emit("img-moving", { moving: true, axis: this.getImgAxis() }); } else { this.cropping = true; window.addEventListener("mousemove", this.createCrop); window.addEventListener("mouseup", this.endCrop); window.addEventListener("touchmove", this.createCrop); window.addEventListener("touchend", this.endCrop); this.cropOffsertX = e.offsetX ? e.offsetX : e.touches[0].pageX - this.$refs.cropper.offsetLeft; this.cropOffsertY = e.offsetY ? e.offsetY : e.touches[0].pageY - this.$refs.cropper.offsetTop; this.cropX = 'clientX' in e ? e.clientX : e.touches[0].clientX; this.cropY = 'clientY' in e ? e.clientY : e.touches[0].clientY; this.cropChangeX = this.cropOffsertX; this.cropChangeY = this.cropOffsertY; this.cropW = 0; this.cropH = 0; } }, touchScale(e) { e.preventDefault(); let scale = this.scale; var oldTouch1 = { x: this.touches[0].clientX, y: this.touches[0].clientY }; var newTouch1 = { x: e.touches[0].clientX, y: e.touches[0].clientY }; var oldTouch2 = { x: this.touches[1].clientX, y: this.touches[1].clientY }; var newTouch2 = { x: e.touches[1].clientX, y: e.touches[1].clientY }; var oldL = Math.sqrt( Math.pow(oldTouch1.x - oldTouch2.x, 2) + Math.pow(oldTouch1.y - oldTouch2.y, 2) ); var newL = Math.sqrt( Math.pow(newTouch1.x - newTouch2.x, 2) + Math.pow(newTouch1.y - newTouch2.y, 2) ); var cha = newL - oldL; var coe = 1; coe = coe / this.trueWidth > coe / this.trueHeight ? coe / this.trueHeight : coe / this.trueWidth; coe = coe > 0.1 ? 0.1 : coe; var num = coe \* cha; if (!this.touchNow) { this.touchNow = true; if (cha > 0) { scale += Math.abs(num); } else if (cha < 0) { scale > Math.abs(num) ? (scale -= Math.abs(num)) : scale; } this.touches = e.touches; setTimeout(() => { this.touchNow = false; }, 8); if (!this.checkoutImgAxis(this.x, this.y, scale)) { return false; } this.scale = scale; } }, cancelTouchScale(e) { window.removeEventListener("touchmove", this.touchScale); }, moveImg(e) { e.preventDefault(); if (e.touches && e.touches.length === 2) { this.touches = e.touches; window.addEventListener("touchmove", this.touchScale); window.addEventListener("touchend", this.cancelTouchScale); window.removeEventListener("touchmove", this.moveImg); return false; } let nowX = 'clientX' in e ? e.clientX : e.touches[0].clientX; let nowY = 'clientY' in e ? e.clientY : e.touches[0].clientY; let changeX, changeY; changeX = nowX - this.moveX; changeY = nowY - this.moveY; this.$nextTick(() => { if (this.centerBox) { let axis = this.getImgAxis(changeX, changeY, this.scale); let cropAxis = this.getCropAxis(); let imgW = this.trueHeight \* this.scale; let imgH = this.trueWidth \* this.scale; let maxLeft, maxTop, maxRight, maxBottom; switch (this.rotate) { case 1: case -1: case 3: case -3: maxLeft = this.cropOffsertX - (this.trueWidth \* (1 - this.scale)) / 2 + (imgW - imgH) / 2; maxTop = this.cropOffsertY - (this.trueHeight \* (1 - this.scale)) / 2 + (imgH - imgW) / 2; maxRight = maxLeft - imgW + this.cropW; maxBottom = maxTop - imgH + this.cropH; break; default: maxLeft = this.cropOffsertX - (this.trueWidth \* (1 - this.scale)) / 2; maxTop = this.cropOffsertY - (this.trueHeight \* (1 - this.scale)) / 2; maxRight = maxLeft - imgH + this.cropW; maxBottom = maxTop - imgW + this.cropH; break; } if (axis.x1 >= cropAxis.x1) { changeX = maxLeft; } if (axis.y1 >= cropAxis.y1) { changeY = maxTop; } if (axis.x2 <= cropAxis.x2) { changeX = maxRight; } if (axis.y2 <= cropAxis.y2) { changeY = maxBottom; } } this.x = changeX; this.y = changeY; this.$emit("imgMoving", { moving: true, axis: this.getImgAxis() }); this.$emit("img-moving", { moving: true, axis: this.getImgAxis() }); }); }, leaveImg(e) { window.removeEventListener("mousemove", this.moveImg); window.removeEventListener("touchmove", this.moveImg); window.removeEventListener("mouseup", this.leaveImg); window.removeEventListener("touchend", this.leaveImg); this.$emit("imgMoving", { moving: false, axis: this.getImgAxis() }); this.$emit("img-moving", { moving: false, axis: this.getImgAxis() }); }, scaleImg() { if (this.canScale) { window.addEventListener(this.support, this.changeSize, this.passive); } }, cancelScale() { if (this.canScale) { window.removeEventListener(this.support, this.changeSize); } }, changeSize(e) { e.preventDefault(); let scale = this.scale; var change = e.deltaY || e.wheelDelta; var isFirefox = navigator.userAgent.indexOf("Firefox"); change = isFirefox > 0 ? change \* 30 : change; if (this.isIE) { change = -change; } var coe = this.coe; coe = coe / this.trueWidth > coe / this.trueHeight ? coe / this.trueHeight : coe / this.trueWidth; var num = coe \* change; num < 0 ? (scale += Math.abs(num)) : scale > Math.abs(num) ? (scale -= Math.abs(num)) : scale; let status = num < 0 ? "add" : "reduce"; if (status !== this.coeStatus) { this.coeStatus = status; this.coe = 0.2; } if (!this.scaling) { this.scalingSet = setTimeout(() => { this.scaling = false; this.coe = this.coe += 0.01; }, 50); } this.scaling = true; if (!this.checkoutImgAxis(this.x, this.y, scale)) { return false; } this.scale = scale; }, changeScale(num) { let scale = this.scale; num = num || 1; var coe = 20; coe = coe / this.trueWidth > coe / this.trueHeight ? coe / this.trueHeight : coe / this.trueWidth; num = num \* coe; num > 0 ? (scale += Math.abs(num)) : scale > Math.abs(num) ? (scale -= Math.abs(num)) : scale; if (!this.checkoutImgAxis(this.x, this.y, scale)) { return false; } this.scale = scale; }, createCrop(e) { e.preventDefault(); var nowX = 'clientX' in e ? e.clientX : e.touches ? e.touches[0].clientX : 0; var nowY = 'clientY' in e ? e.clientY : e.touches ? e.touches[0].clientY : 0; this.$nextTick(() => { var fw = nowX - this.cropX; var fh = nowY - this.cropY; if (fw > 0) { this.cropW = fw + this.cropChangeX > this.w ? this.w - this.cropChangeX : fw; this.cropOffsertX = this.cropChangeX; } else { this.cropW = this.w - this.cropChangeX + Math.abs(fw) > this.w ? this.cropChangeX : Math.abs(fw); this.cropOffsertX = this.cropChangeX + fw > 0 ? this.cropChangeX + fw : 0; } if (!this.fixed) { if (fh > 0) { this.cropH = fh + this.cropChangeY > this.h ? this.h - this.cropChangeY : fh; this.cropOffsertY = this.cropChangeY; } else { this.cropH = this.h - this.cropChangeY + Math.abs(fh) > this.h ? this.cropChangeY : Math.abs(fh); this.cropOffsertY = this.cropChangeY + fh > 0 ? this.cropChangeY + fh : 0; } } else { var fixedHeight = (this.cropW / this.fixedNumber[0]) \* this.fixedNumber[1]; if (fixedHeight + this.cropOffsertY > this.h) { this.cropH = this.h - this.cropOffsertY; this.cropW = (this.cropH / this.fixedNumber[1]) \* this.fixedNumber[0]; if (fw > 0) { this.cropOffsertX = this.cropChangeX; } else { this.cropOffsertX = this.cropChangeX - this.cropW; } } else { this.cropH = fixedHeight; } this.cropOffsertY = this.cropOffsertY; } }); }, changeCropSize(e, w, h, typeW, typeH) { e.preventDefault(); window.addEventListener("mousemove", this.changeCropNow); window.addEventListener("mouseup", this.changeCropEnd); window.addEventListener("touchmove", this.changeCropNow); window.addEventListener("touchend", this.changeCropEnd); this.canChangeX = w; this.canChangeY = h; this.changeCropTypeX = typeW; this.changeCropTypeY = typeH; this.cropX = 'clientX' in e ? e.clientX : e.touches[0].clientX; this.cropY = 'clientY' in e ? e.clientY : e.touches[0].clientY; this.cropOldW = this.cropW; this.cropOldH = this.cropH; this.cropChangeX = this.cropOffsertX; this.cropChangeY = this.cropOffsertY; if (this.fixed) { if (this.canChangeX && this.canChangeY) { this.canChangeY = 0; } } this.$emit('change-crop-size', { width: this.cropW, height: this.cropH }) }, changeCropNow(e) { e.preventDefault(); var nowX = 'clientX' in e ? e.clientX : e.touches ? e.touches[0].clientX : 0; var nowY = 'clientY' in e ? e.clientY : e.touches ? e.touches[0].clientY : 0; let wrapperW = this.w; let wrapperH = this.h; let minX = 0; let minY = 0; if (this.centerBox) { let axis = this.getImgAxis(); let imgW = axis.x2; let imgH = axis.y2; minX = axis.x1 > 0 ? axis.x1 : 0; minY = axis.y1 > 0 ? axis.y1 : 0; if (wrapperW > imgW) { wrapperW = imgW; } if (wrapperH > imgH) { wrapperH = imgH; } } this.$nextTick(() => { var fw = nowX - this.cropX; var fh = nowY - this.cropY; if (this.canChangeX) { if (this.changeCropTypeX === 1) { if (this.cropOldW - fw > 0) { this.cropW = wrapperW - this.cropChangeX - fw <= wrapperW - minX ? this.cropOldW - fw : this.cropOldW + this.cropChangeX - minX; this.cropOffsertX = wrapperW - this.cropChangeX - fw <= wrapperW - minX ? this.cropChangeX + fw : minX; } else { this.cropW = Math.abs(fw) + this.cropChangeX <= wrapperW ? Math.abs(fw) - this.cropOldW : wrapperW - this.cropOldW - this.cropChangeX; this.cropOffsertX = this.cropChangeX + this.cropOldW; } } else if (this.changeCropTypeX === 2) { if (this.cropOldW + fw > 0) { this.cropW = this.cropOldW + fw + this.cropOffsertX <= wrapperW ? this.cropOldW + fw : wrapperW - this.cropOffsertX; this.cropOffsertX = this.cropChangeX; } else { this.cropW = wrapperW - this.cropChangeX + Math.abs(fw + this.cropOldW) <= wrapperW - minX ? Math.abs(fw + this.cropOldW) : this.cropChangeX - minX; this.cropOffsertX = wrapperW - this.cropChangeX + Math.abs(fw + this.cropOldW) <= wrapperW - minX ? this.cropChangeX - Math.abs(fw + this.cropOldW) : minX; } } } if (this.canChangeY) { if (this.changeCropTypeY === 1) { if (this.cropOldH - fh > 0) { this.cropH = wrapperH - this.cropChangeY - fh <= wrapperH - minY ? this.cropOldH - fh : this.cropOldH + this.cropChangeY - minY; this.cropOffsertY = wrapperH - this.cropChangeY - fh <= wrapperH - minY ? this.cropChangeY + fh : minY; } else { this.cropH = Math.abs(fh) + this.cropChangeY <= wrapperH ? Math.abs(fh) - this.cropOldH : wrapperH - this.cropOldH - this.cropChangeY; this.cropOffsertY = this.cropChangeY + this.cropOldH; } } else if (this.changeCropTypeY === 2) { if (this.cropOldH + fh > 0) { this.cropH = this.cropOldH + fh + this.cropOffsertY <= wrapperH ? this.cropOldH + fh : wrapperH - this.cropOffsertY; this.cropOffsertY = this.cropChangeY; } else { this.cropH = wrapperH - this.cropChangeY + Math.abs(fh + this.cropOldH) <= wrapperH - minY ? Math.abs(fh + this.cropOldH) : this.cropChangeY - minY; this.cropOffsertY = wrapperH - this.cropChangeY + Math.abs(fh + this.cropOldH) <= wrapperH - minY ? this.cropChangeY - Math.abs(fh + this.cropOldH) : minY; } } } if (this.canChangeX && this.fixed) { var fixedHeight = (this.cropW / this.fixedNumber[0]) \* this.fixedNumber[1]; if (fixedHeight + this.cropOffsertY > wrapperH) { this.cropH = wrapperH - this.cropOffsertY; this.cropW = (this.cropH / this.fixedNumber[1]) \* this.fixedNumber[0]; } else { this.cropH = fixedHeight; } } if (this.canChangeY && this.fixed) { var fixedWidth = (this.cropH / this.fixedNumber[1]) \* this.fixedNumber[0]; if (fixedWidth + this.cropOffsertX > wrapperW) { this.cropW = wrapperW - this.cropOffsertX; this.cropH = (this.cropW / this.fixedNumber[0]) \* this.fixedNumber[1]; } else { this.cropW = fixedWidth; } } }); }, checkCropLimitSize () { let { cropW, cropH, limitMinSize } = this; let limitMinNum = new Array; if (!Array.isArray[limitMinSize]) { limitMinNum = [limitMinSize, limitMinSize] } else { limitMinNum = limitMinSize } cropW = parseFloat(limitMinNum[0]) cropH = parseFloat(limitMinNum[1]) return [cropW, cropH] }, changeCropEnd(e) { window.removeEventListener("mousemove", this.changeCropNow); window.removeEventListener("mouseup", this.changeCropEnd); window.removeEventListener("touchmove", this.changeCropNow); window.removeEventListener("touchend", this.changeCropEnd); }, endCrop() { if (this.cropW === 0 && this.cropH === 0) { this.cropping = false; } window.removeEventListener("mousemove", this.createCrop); window.removeEventListener("mouseup", this.endCrop); window.removeEventListener("touchmove", this.createCrop); window.removeEventListener("touchend", this.endCrop); }, startCrop() { this.crop = true; }, stopCrop() { this.crop = false; }, clearCrop() { this.cropping = false; this.cropW = 0; this.cropH = 0; }, cropMove(e) { e.preventDefault(); if (!this.canMoveBox) { this.crop = false; this.startMove(e); return false; } if (e.touches && e.touches.length === 2) { this.crop = false; this.startMove(e); this.leaveCrop(); return false; } window.addEventListener("mousemove", this.moveCrop); window.addEventListener("mouseup", this.leaveCrop); window.addEventListener("touchmove", this.moveCrop); window.addEventListener("touchend", this.leaveCrop); let x = 'clientX' in e ? e.clientX : e.touches[0].clientX; let y = 'clientY' in e ? e.clientY : e.touches[0].clientY; let newX, newY; newX = x - this.cropOffsertX; newY = y - this.cropOffsertY; this.cropX = newX; this.cropY = newY; this.$emit("cropMoving", { moving: true, axis: this.getCropAxis() }); this.$emit("crop-moving", { moving: true, axis: this.getCropAxis() }); }, moveCrop(e, isMove) { let nowX = 0; let nowY = 0; if (e) { e.preventDefault(); nowX = 'clientX' in e ? e.clientX : e.touches[0].clientX; nowY = 'clientY' in e ? e.clientY : e.touches[0].clientY; } this.$nextTick(() => { let cx, cy; let fw = nowX - this.cropX; let fh = nowY - this.cropY; if (isMove) { fw = this.cropOffsertX; fh = this.cropOffsertY; } if (fw <= 0) { cx = 0; } else if (fw + this.cropW > this.w) { cx = this.w - this.cropW; } else { cx = fw; } if (fh <= 0) { cy = 0; } else if (fh + this.cropH > this.h) { cy = this.h - this.cropH; } else { cy = fh; } if (this.centerBox) { let axis = this.getImgAxis(); if (cx <= axis.x1) { cx = axis.x1; } if (cx + this.cropW > axis.x2) { cx = axis.x2 - this.cropW; } if (cy <= axis.y1) { cy = axis.y1; } if (cy + this.cropH > axis.y2) { cy = axis.y2 - this.cropH; } } this.cropOffsertX = cx; this.cropOffsertY = cy; this.$emit("cropMoving", { moving: true, axis: this.getCropAxis() }); this.$emit("crop-moving", { moving: true, axis: this.getCropAxis() }); }); }, getImgAxis(x, y, scale) { x = x || this.x; y = y || this.y; scale = scale || this.scale; let obj = { x1: 0, x2: 0, y1: 0, y2: 0 }; let imgW = this.trueWidth \* scale; let imgH = this.trueHeight \* scale; switch (this.rotate) { case 0: obj.x1 = x + (this.trueWidth \* (1 - scale)) / 2; obj.x2 = obj.x1 + this.trueWidth \* scale; obj.y1 = y + (this.trueHeight \* (1 - scale)) / 2; obj.y2 = obj.y1 + this.trueHeight \* scale; break; case 1: case -1: case 3: case -3: obj.x1 = x + (this.trueWidth \* (1 - scale)) / 2 + (imgW - imgH) / 2; obj.x2 = obj.x1 + this.trueHeight \* scale; obj.y1 = y + (this.trueHeight \* (1 - scale)) / 2 + (imgH - imgW) / 2; obj.y2 = obj.y1 + this.trueWidth \* scale; break; default: obj.x1 = x + (this.trueWidth \* (1 - scale)) / 2; obj.x2 = obj.x1 + this.trueWidth \* scale; obj.y1 = y + (this.trueHeight \* (1 - scale)) / 2; obj.y2 = obj.y1 + this.trueHeight \* scale; break; } return obj; }, getCropAxis() { let obj = { x1: 0, x2: 0, y1: 0, y2: 0 }; obj.x1 = this.cropOffsertX; obj.x2 = obj.x1 + this.cropW; obj.y1 = this.cropOffsertY; obj.y2 = obj.y1 + this.cropH; return obj; }, leaveCrop(e) { window.removeEventListener("mousemove", this.moveCrop); window.removeEventListener("mouseup", this.leaveCrop); window.removeEventListener("touchmove", this.moveCrop); window.removeEventListener("touchend", this.leaveCrop); this.$emit("cropMoving", { moving: false, axis: this.getCropAxis() }); this.$emit("crop-moving", { moving: false, axis: this.getCropAxis() }); }, getCropChecked(cb) { let canvas = document.createElement("canvas"); let img = new Image(); let rotate = this.rotate; let trueWidth = this.trueWidth; let trueHeight = this.trueHeight; let cropOffsertX = this.cropOffsertX; let cropOffsertY = this.cropOffsertY; img.onload = () => { if (this.cropW !== 0) { let ctx = canvas.getContext("2d"); let dpr = 1; if (this.high & !this.full) { dpr = window.devicePixelRatio; } if ((this.enlarge !== 1) & !this.full) { dpr = Math.abs(Number(this.enlarge)); } let width = this.cropW \* dpr; let height = this.cropH \* dpr; let imgW = trueWidth \* this.scale \* dpr; let imgH = trueHeight \* this.scale \* dpr; let dx = (this.x - cropOffsertX + (this.trueWidth \* (1 - this.scale)) / 2) \* dpr; let dy = (this.y - cropOffsertY + (this.trueHeight \* (1 - this.scale)) / 2) \* dpr; setCanvasSize(width, height); ctx.save(); switch (rotate) { case 0: if (!this.full) { ctx.drawImage(img, dx, dy, imgW, imgH); } else { setCanvasSize(width / this.scale, height / this.scale); ctx.drawImage( img, dx / this.scale, dy / this.scale, imgW / this.scale, imgH / this.scale ); } break; case 1: case -3: if (!this.full) { dx = dx + (imgW - imgH) / 2; dy = dy + (imgH - imgW) / 2; ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, dy, -dx - imgH, imgW, imgH); } else { setCanvasSize(width / this.scale, height / this.scale); dx = dx / this.scale + (imgW / this.scale - imgH / this.scale) / 2; dy = dy / this.scale + (imgH / this.scale - imgW / this.scale) / 2; ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage( img, dy, -dx - imgH / this.scale, imgW / this.scale, imgH / this.scale ); } break; case 2: case -2: if (!this.full) { ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, -dx - imgW, -dy - imgH, imgW, imgH); } else { setCanvasSize(width / this.scale, height / this.scale); ctx.rotate((rotate \* 90 \* Math.PI) / 180); dx = dx / this.scale; dy = dy / this.scale; ctx.drawImage( img, -dx - imgW / this.scale, -dy - imgH / this.scale, imgW / this.scale, imgH / this.scale ); } break; case 3: case -1: if (!this.full) { dx = dx + (imgW - imgH) / 2; dy = dy + (imgH - imgW) / 2; ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, -dy - imgW, dx, imgW, imgH); } else { setCanvasSize(width / this.scale, height / this.scale); dx = dx / this.scale + (imgW / this.scale - imgH / this.scale) / 2; dy = dy / this.scale + (imgH / this.scale - imgW / this.scale) / 2; ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage( img, -dy - imgW / this.scale, dx, imgW / this.scale, imgH / this.scale ); } break; default: if (!this.full) { ctx.drawImage(img, dx, dy, imgW, imgH); } else { setCanvasSize(width / this.scale, height / this.scale); ctx.drawImage( img, dx / this.scale, dy / this.scale, imgW / this.scale, imgH / this.scale ); } } ctx.restore(); } else { let width = trueWidth \* this.scale; let height = trueHeight \* this.scale; let ctx = canvas.getContext("2d"); ctx.save(); switch (rotate) { case 0: setCanvasSize(width, height); ctx.drawImage(img, 0, 0, width, height); break; case 1: case -3: setCanvasSize(height, width); ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, 0, -height, width, height); break; case 2: case -2: setCanvasSize(width, height); ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, -width, -height, width, height); break; case 3: case -1: setCanvasSize(height, width); ctx.rotate((rotate \* 90 \* Math.PI) / 180); ctx.drawImage(img, -width, 0, width, height); break; default: setCanvasSize(width, height); ctx.drawImage(img, 0, 0, width, height); } ctx.restore(); } cb(canvas); }; var s = this.img.substr(0, 4); if (s !== "data") { img.crossOrigin = "Anonymous"; } img.src = this.imgs; function setCanvasSize(width, height) { canvas.width = Math.round(width); canvas.height = Math.round(height); } }, getCropData(cb) { this.getCropChecked(data => { cb(data.toDataURL("image/" + this.outputType, this.outputSize)); }); }, getCropBlob(cb) { this.getCropChecked(data => { data.toBlob( blob => cb(blob), "image/" + this.outputType, this.outputSize ); }); }, showPreview() { if (this.isCanShow) { this.isCanShow = false; setTimeout(() => { this.isCanShow = true; }, 16); } else { return false; } let w = this.cropW; let h = this.cropH; let scale = this.scale; var obj = {}; obj.div = { width: `${w}px`, height: `${h}px` }; let transformX = (this.x - this.cropOffsertX) / scale; let transformY = (this.y - this.cropOffsertY) / scale; let transformZ = 0; obj.w = w; obj.h = h; obj.url = this.imgs; obj.img = { width: `${this.trueWidth}px`, height: `${this.trueHeight}px`, transform: `scale(${scale})translate3d(${transformX}px, ${transformY}px, ${transformZ}px)rotateZ(${this .rotate \* 90}deg)` }; obj.html = ` <div class="show-preview" style="width: ${obj.w}px; height: ${ obj.h }px,; overflow: hidden"> <div style="width: ${w}px; height: ${h}px"> <img src=${obj.url} style="width: ${this.trueWidth}px; height: ${ this.trueHeight }px; transform: scale(${scale})translate3d(${transformX}px, ${transformY}px, ${transformZ}px)rotateZ(${this .rotate \* 90}deg)"> </div> </div>`; this.$emit("realTime", obj); this.$emit("real-time", obj); }, reload() { let img = new Image(); img.onload = () => { this.w = parseFloat(window.getComputedStyle(this.$refs.cropper).width); this.h = parseFloat(window.getComputedStyle(this.$refs.cropper).height); this.trueWidth = img.width; this.trueHeight = img.height; if (!this.original) { this.scale = this.checkedMode(); } else { this.scale = 1; } this.$nextTick(() => { this.x = -(this.trueWidth - this.trueWidth \* this.scale) / 2 + (this.w - this.trueWidth \* this.scale) / 2; this.y = -(this.trueHeight - this.trueHeight \* this.scale) / 2 + (this.h - this.trueHeight \* this.scale) / 2; this.loading = false; if (this.autoCrop) { this.goAutoCrop(); } this.$emit("img-load", "success"); this.$emit("imgLoad", "success"); setTimeout(() => { this.showPreview(); }, 20); }); }; img.onerror = () => { this.$emit("imgLoad", "error"); this.$emit("img-load", "error"); }; img.src = this.imgs; }, checkedMode() { let scale = 1; let imgW = this.trueWidth; let imgH = this.trueHeight; const arr = this.mode.split(" "); switch (arr[0]) { case "contain": if (this.trueWidth > this.w) { scale = this.w / this.trueWidth; } if (this.trueHeight \* scale > this.h) { scale = this.h / this.trueHeight; } break; case "cover": imgW = this.w; scale = imgW / this.trueWidth; imgH = imgH \* scale; if (imgH < this.h) { imgH = this.h; scale = imgH / this.trueHeight; } break; default: try { let str = arr[0]; if (str.search("px") !== -1) { str = str.replace("px", ""); imgW = parseFloat(str); const scaleX = imgW / this.trueWidth; let scaleY = 1; let strH = arr[1]; if (strH.search("px") !== -1) { strH = strH.replace("px", ""); imgH = parseFloat(strH); scaleY = imgH / this.trueHeight; } scale = Math.min(scaleX,scaleY) } if (str.search("%") !== -1) { str = str.replace("%", ""); imgW = (parseFloat(str) / 100) \* this.w; scale = imgW / this.trueWidth; } if (arr.length === 2 && str === "auto") { let str2 = arr[1]; if (str2.search("px") !== -1) { str2 = str2.replace("px", ""); imgH = parseFloat(str2); scale = imgH / this.trueHeight; } if (str2.search("%") !== -1) { str2 = str2.replace("%", ""); imgH = (parseFloat(str2) / 100) \* this.h; scale = imgH / this.trueHeight; } } } catch (error) { scale = 1; } } return scale; }, goAutoCrop(cw, ch) { if (this.imgs === '' || this.imgs === null) return this.clearCrop(); this.cropping = true; let maxWidth = this.w; let maxHeight = this.h; if (this.centerBox) { const switchWH = Math.abs(this.rotate) % 2 > 0 let imgW = (switchWH ? this.trueHeight : this.trueWidth) \* this.scale; let imgH = (switchWH ? this.trueWidth : this.trueHeight) \* this.scale; maxWidth = imgW < maxWidth ? imgW : maxWidth; maxHeight = imgH < maxHeight ? imgH : maxHeight; } var w = cw ? cw : parseFloat(this.autoCropWidth); var h = ch ? ch : parseFloat(this.autoCropHeight); if (w === 0 || h === 0) { w = maxWidth \* 0.8; h = maxHeight \* 0.8; } w = w > maxWidth ? maxWidth : w; h = h > maxHeight ? maxHeight : h; if (this.fixed) { h = (w / this.fixedNumber[0]) \* this.fixedNumber[1]; } if (h > this.h) { h = this.h; w = (h / this.fixedNumber[1]) \* this.fixedNumber[0]; } this.changeCrop(w, h); }, changeCrop(w, h) { if (this.centerBox) { let axis = this.getImgAxis(); if (w > axis.x2 - axis.x1) { w = axis.x2 - axis.x1; h = (w / this.fixedNumber[0]) \* this.fixedNumber[1]; } if (h > axis.y2 - axis.y1) { h = axis.y2 - axis.y1; w = (h / this.fixedNumber[1]) \* this.fixedNumber[0]; } } this.cropW = w; this.cropH = h; this.checkCropLimitSize() this.$nextTick(() => { this.cropOffsertX = (this.w - this.cropW) / 2; this.cropOffsertY = (this.h - this.cropH) / 2; if (this.centerBox) { this.moveCrop(null, true); } }); }, refresh() { let img = this.img; this.imgs = ""; this.scale = 1; this.crop = false; this.rotate = 0; this.w = 0; this.h = 0; this.trueWidth = 0; this.trueHeight = 0; this.clearCrop(); this.$nextTick(() => { this.checkedImg(); }); }, rotateLeft() { this.rotate = this.rotate <= -3 ? 0 : this.rotate - 1; }, rotateRight() { this.rotate = this.rotate >= 3 ? 0 : this.rotate + 1; }, rotateClear() { this.rotate = 0; }, checkoutImgAxis(x, y, scale) { x = x || this.x; y = y || this.y; scale = scale || this.scale; let canGo = true; if (this.centerBox) { let axis = this.getImgAxis(x, y, scale); let cropAxis = this.getCropAxis(); if (axis.x1 >= cropAxis.x1) { canGo = false; } if (axis.x2 <= cropAxis.x2) { canGo = false; } if (axis.y1 >= cropAxis.y1) { canGo = false; } if (axis.y2 <= cropAxis.y2) { canGo = false; } } return canGo; } }, mounted() { this.support = "onwheel" in document.createElement("div") ? "wheel" : document.onmousewheel !== undefined ? "mousewheel" : "DOMMouseScroll"; let that = this; var u = navigator.userAgent; this.isIOS = !!u.match(/\(i[^;]+;( U;)? CPU.+Mac OS X/); if (!HTMLCanvasElement.prototype.toBlob) { Object.defineProperty(HTMLCanvasElement.prototype, "toBlob", { value: function(callback, type, quality) { var binStr = atob(this.toDataURL(type, quality).split(",")[1]), len = binStr.length, arr = new Uint8Array(len); for (var i = 0; i < len; i++) { arr[i] = binStr.charCodeAt(i); } callback(new Blob([arr], { type: that.type || "image/png" })); } }); } this.showPreview(); this.checkedImg(); }, unmounted() { window.removeEventListener("mousemove", this.moveCrop); window.removeEventListener("mouseup", this.leaveCrop); window.removeEventListener("touchmove", this.moveCrop); window.removeEventListener("touchend", this.leaveCrop); this.cancelScale() }});</script><style scoped lang="css">.vue-cropper { position: relative; width: 100%; height: 100%; box-sizing: border-box; user-select: none; -webkit-user-select: none; -moz-user-select: none; -ms-user-select: none; direction: ltr; touch-action: none; text-align: left; background-image: url("");}.cropper-box,.cropper-box-canvas,.cropper-drag-box,.cropper-crop-box,.cropper-face { position: absolute; top: 0; right: 0; bottom: 0; left: 0; user-select: none;}.cropper-box-canvas img { position: relative; text-align: left; user-select: none; transform: none; max-width: none; max-height: none;}.cropper-box { overflow: hidden;}.cropper-move { cursor: move;}.cropper-crop { cursor: crosshair;}.cropper-modal { background: rgba(0, 0, 0, 0.5);}.cropper-crop-box {}.cropper-view-box { display: block; overflow: hidden; width: 100%; height: 100%; outline: 1px solid #39f; outline-color: rgba(51, 153, 255, 0.75); user-select: none;}.cropper-view-box img { user-select: none; text-align: left; max-width: none; max-height: none;}.cropper-face { top: 0; left: 0; background-color: #fff; opacity: 0.1;}.crop-info { position: absolute; left: 0px; min-width: 65px; text-align: center; color: white; line-height: 20px; background-color: rgba(0, 0, 0, 0.8); font-size: 12px;}.crop-line { position: absolute; display: block; width: 100%; height: 100%; opacity: 0.1;}.line-w { top: -3px; left: 0; height: 5px; cursor: n-resize;}.line-a { top: 0; left: -3px; width: 5px; cursor: w-resize;}.line-s { bottom: -3px; left: 0; height: 5px; cursor: s-resize;}.line-d { top: 0; right: -3px; width: 5px; cursor: e-resize;}.crop-point { position: absolute; width: 8px; height: 8px; opacity: 0.75; background-color: #39f; border-radius: 100%;}.point1 { top: -4px; left: -4px; cursor: nw-resize;}.point2 { top: -5px; left: 50%; margin-left: -3px; cursor: n-resize;}.point3 { top: -4px; right: -4px; cursor: ne-resize;}.point4 { top: 50%; left: -4px; margin-top: -3px; cursor: w-resize;}.point5 { top: 50%; right: -4px; margin-top: -3px; cursor: e-resize;}.point6 { bottom: -5px; left: -4px; cursor: sw-resize;}.point7 { bottom: -5px; left: 50%; margin-left: -3px; cursor: s-resize;}.point8 { bottom: -5px; right: -4px; cursor: se-resize;}@media screen and (max-width: 500px) { .crop-point { position: absolute; width: 20px; height: 20px; opacity: 0.45; background-color: #39f; border-radius: 100%; } .point1 { top: -10px; left: -10px; } .point2, .point4, .point5, .point7 { display: none; } .point3 { top: -10px; right: -10px; } .point4 { top: 0; left: 0; } .point6 { bottom: -10px; left: -10px; } .point8 { bottom: -10px; right: -10px; }}</style>  
  
Test.java  
  
package me.king;  
public class Test {  
 public static void main(String[] args) {  
 }  
}  
  
MainTest.java  
  
package me.king;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.test.context.junit4.SpringRunner;  
@RunWith(SpringRunner.class)  
@SpringBootTest(classes = Application.class)  
public class MainTest {  
 @Test  
 public void test() {  
 }  
}  
  
DeleteComments.java  
  
package me.king.core.utils;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;  
public class DeleteComments {  
 private static int count = 0;  
   
 public static void clearComment(File file, String charset) {  
 try {  
 if (!file.exists()) {  
 return;  
 }  
 if (file.isDirectory()) {  
 File[] files = file.listFiles();  
 for (File f : files) {  
 clearComment(f, charset); //递归调用  
 }  
 return;  
 } else if (!file.getName().endsWith(".js")) {  
 return;  
 }  
 System.out.println("-----开始处理文件：" + file.getAbsolutePath());  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream(file), charset));  
 StringBuffer content = new StringBuffer();  
 String tmp = null;  
 while ((tmp = reader.readLine()) != null) {  
 content.append(tmp);  
 content.append("\n");  
 }  
 String target = content.toString();  
 if(target.contains("www.xiaonuo.vip")){  
 String s = target.replaceFirst("/\\\*{1,2}[\\s\\S]\*?\\\*/", "");  
 BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file), charset));  
 out.write(s);  
 out.flush();  
 out.close();  
 count++;  
 }  
 System.out.println("-----文件处理完成---" + count);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public static void clearComment(String filePath, String charset) {  
 clearComment(new File(filePath), charset);  
 }  
 public static void clearComment(String filePath) {  
 clearComment(new File(filePath), "UTF-8");  
 }  
 public static void clearComment(File file) {  
 clearComment(file, "UTF-8");  
 }  
 public static void main(String[] args) {  
 clearComment("D:\\Project\\BackCode\\snowy\\easy-antd\\easy-admin-web"); //删除目录下所有java文件注释  
 }  
}