<script setup lang="ts">  
import { RouterView } from 'vue-router'  
</script>  
<template>  
 <RouterView />  
</template>  
<style scoped>  
</style>  
@import "tailwindcss/base";  
@import "tailwindcss/components";  
@import "tailwindcss/utilities";  
<script lang="ts">  
export type { CountUp as ICountUp, CountUpOptions } from 'countup.js'  
export default {  
 name: 'CountUp'  
}  
</script>  
<script setup lang="ts">  
import { onMounted, onUnmounted, ref, watch } from 'vue'  
import { CountUp } from 'countup.js'  
import type { CountUpOptions } from 'countup.js'  
const props = withDefaults(  
 defineProps<{  
 endVal: number | string  
 startVal?: number | string  
 duration?: number | string  
 autoplay?: boolean  
 loop?: boolean | number | string  
 delay?: number  
 options?: CountUpOptions  
 }>(),  
 {  
 startVal: 0,  
 duration: 2.5,  
 autoplay: true,  
 loop: false,  
 delay: 0,  
 options: undefined  
 }  
)  
const emits = defineEmits<{  
 (event: 'init', countup: CountUp): void  
 (event: 'finished'): void  
}>()  
let elRef = ref<HTMLElement>()  
let countUp = ref<CountUp>()  
const initCountUp = () => {  
 if (!elRef.value) return  
 const startVal = Number(props.startVal)  
 const endVal = Number(props.endVal)  
 const duration = Number(props.duration)  
 countUp.value = new CountUp(elRef.value, endVal, {  
 startVal,  
 duration,  
 ...props.options  
 })  
 if (countUp.value.error) {  
 console.error(countUp.value.error)  
 return  
 }  
 emits('init', countUp.value)  
}  
const startAnim = (cb?: () => void) => {  
 countUp.value?.start(cb)  
}  
watch(  
 () => props.endVal,  
 (value) => {  
 if (props.autoplay) {  
 countUp.value?.update(value)  
 }  
 }  
)  
const finished = ref(false)  
let loopCount = 0  
const loopAnim = () => {  
 loopCount++  
 startAnim(() => {  
 const isTruely = typeof props.loop === 'boolean' && props.loop  
 if (isTruely || props.loop > loopCount) {  
 delay(() => {  
 countUp.value?.reset()  
 loopAnim()  
 }, props.delay)  
 } else {  
 finished.value = true  
 }  
 })  
}  
watch(finished, (flag) => {  
 if (flag) {  
 emits('finished')  
 }  
})  
onMounted(() => {  
 initCountUp()  
 if (props.autoplay) {  
 loopAnim()  
 }  
})  
onUnmounted(() => {  
 cancelAnimationFrame(dalayRafId)  
 countUp.value?.reset()  
})  
let dalayRafId: number  
const delay = (cb: () => unknown, seconds = 1) => {  
 let startTime: number  
 function count(timestamp: number) {  
 if (!startTime) startTime = timestamp  
 const diff = timestamp - startTime  
 if (diff < seconds \* 1000) {  
 dalayRafId = requestAnimationFrame(count)  
 } else {  
 cb()  
 }  
 }  
 dalayRafId = requestAnimationFrame(count)  
}  
const restart = () => {  
 initCountUp()  
 startAnim()  
}  
defineExpose({  
 init: initCountUp,  
 restart  
})  
</script>  
<template>  
 <div class="countup-wrap">  
 <slot name="prefix"></slot>  
 <span ref="elRef"> </span>  
 <slot name="suffix"></slot>  
 </div>  
</template>  
<script setup lang="ts">  
import { computed, ref ,onBeforeUpdate, nextTick} from "vue";  
import merge from "lodash/merge";  
import { useElementSize } from "@vueuse/core";  
import type { PropType } from "vue";  
const props = defineProps({  
 color: {  
 type: Array as unknown as PropType<[string, string]>,  
 default: () => [],  
 },  
 backgroundColor: {  
 type: String,  
 default: "transparent",  
 },  
});  
const defaultColor = ["#6586ec", "#2cf7fe"];  
const domRef = ref(null);  
const { width, height } = useElementSize(domRef,{width:0,height:0}, { box: 'border-box' });  
const mergedColor = computed<[string, string]>(() => {  
 return merge(defaultColor, props.color);  
});  
</script>  
<template>  
 <div class="dv-border-box-13 dv-border-box" ref="domRef">  
 <svg :width="width" :height="height" class="dv-border-svg-container">  
 <path  
 :fill="backgroundColor"  
 :stroke="mergedColor[0]"  
 :d="`  
 M 5 20 L 5 10 L 12 3 L 60 3 L 68 10  
 L ${width - 20} 10 L ${width - 5} 25  
 L ${width - 5} ${height - 5} L 20 ${height - 5}  
 L 5 ${height - 20} L 5 20  
 `"  
 />  
 <path  
 fill="transparent"  
 stroke-width="3"  
 stroke-linecap="round"  
 stroke-dasharray="10, 5"  
 :stroke="mergedColor[0]"  
 :d="`M 16 9 L 61 9`"  
 />  
 <path  
 fill="transparent"  
 stroke="{mergedColor[1]}"  
 :d="`M 5 20 L 5 10 L 12 3 L 60 3 L 68 10`"  
 />  
 <path  
 fill="transparent"  
 :stroke="mergedColor[1]"  
 :d="`M ${width - 5} ${height - 30} L ${width - 5} ${height - 5} L ${  
 width - 30  
 } ${height - 5}`"  
 />  
 </svg>  
 <div class="dv-border-box-content">  
 <slot></slot>  
 </div>  
 </div>  
</template>  
<style scoped lang="scss">  
.dv-border-box {  
 position: relative;  
 box-sizing: border-box;  
 width: 100%;  
 height: 100%;  
}  
.dv-border-svg-container {  
 position: absolute;  
 width: 100%;  
 height: 100%;  
 top: 0px;  
 left: 0px;  
 display: block;  
}  
.dv-border-box-content {  
 position: relative;  
 width: 100%;  
 height: 100%;  
}  
</style>  
<script setup lang="ts">  
import { onMounted, reactive, ref, watch } from "vue";  
import type { DefaultConfigType } from "./index.d";  
import cloneDeep from "lodash/cloneDeep";  
import merge from "lodash/merge";  
const mergedConfig = ref<any>(null);  
const capsuleLength = ref<any>([]);  
const capsuleValue = ref<any>([]);  
const labelData = ref<any>([]);  
const defaultConfig = reactive<DefaultConfigType>({  
 colors: [  
 "#37a2da",  
 "#32c5e9",  
 "#67e0e3",  
 "#9fe6b8",  
 "#ffdb5c",  
 "#ff9f7f",  
 "#fb7293",  
 ],  
 unit: "",  
 showValue: false, // Show item value  
});  
const props = withDefaults(  
 defineProps<{  
 config: object | any;  
 data: Array<{  
 name: string;  
 value: string | number;  
 }>;  
 }>(),  
 {  
 config: () => { },  
 data: () => [],  
 }  
);  
const calcData = () => {  
 mergeConfig();  
 calcCapsuleLengthAndLabelData();  
};  
const mergeConfig = () => {  
 mergedConfig.value = merge(cloneDeep(defaultConfig), props.config || {});  
};  
const calcCapsuleLengthAndLabelData = () => {  
 if (!props.data.length) return;  
 const newcapsuleValue = props.data.map((item: any) => item.value);  
 const maxValue = Math.max(...newcapsuleValue);  
 capsuleValue.value = newcapsuleValue;  
 capsuleLength.value = newcapsuleValue.map((v: any) =>  
 maxValue ? v / maxValue : 0  
 );  
 const oneFifth = maxValue / 5;  
 const newlabelData = Array.from(  
 new Set(new Array(6).fill(0).map((v, i) => Math.ceil(i \* oneFifth)))  
 );  
 labelData.value = newlabelData;  
};  
watch(  
 () => props.data,  
 (newval: any) => {  
 calcData();  
 },  
);  
watch(  
 () => props.config,  
 (newval: any) => {  
 calcData();  
 },  
);  
onMounted(() => {  
 calcData();  
});  
</script>  
<template>  
 <div class="dv-capsule-chart">  
 <template v-if="mergedConfig">  
 <div class="label-column">  
 <div v-for="item in data" :key="item.name">  
 {{ item.name }}  
 </div>  
 <div>&nbsp;</div>  
 </div>  
 <div class="capsule-container">  
 <div class="capsule-item" v-for="(capsule, index) in capsuleLength" :key="index">  
 <div class="capsule-item-column" :style="`width: ${capsule \* 100}%; background-color: ${mergedConfig.colors[index % mergedConfig.colors.length]  
 };`">  
 <div v-if="mergedConfig.showValue" class="capsule-item-value">  
 {{ capsuleValue[index] }}  
 </div>  
 </div>  
 </div>  
 <div class="unit-label">  
 <div v-for="(label, index) in labelData" :key="label + index">  
 {{ label }}  
 </div>  
 </div>  
 </div>  
 <div class="unit-text" v-if="mergedConfig.unit">  
 {{ mergedConfig.unit }}  
 </div>  
 </template>  
 </div>  
</template>  
<style scoped lang="scss">  
.dv-capsule-chart {  
 position: relative;  
 display: flex;  
 flex-direction: row;  
 box-sizing: border-box;  
 padding: 10px;  
 color: #fff;  
 .label-column {  
 display: flex;  
 flex-direction: column;  
 justify-content: space-between;  
 box-sizing: border-box;  
 padding-right: 10px;  
 text-align: right;  
 font-size: 12px;  
 div {  
 height: 20px;  
 line-height: 20px;  
 }  
 }  
 .capsule-container {  
 flex: 1;  
 display: flex;  
 flex-direction: column;  
 justify-content: space-between;  
 }  
 .capsule-item {  
 box-shadow: 0 0 3px #999;  
 height: 10px;  
 margin: 5px 0px;  
 border-radius: 5px;  
 .capsule-item-column {  
 position: relative;  
 height: 8px;  
 margin-top: 1px;  
 border-radius: 5px;  
 transition: all 0.3s;  
 display: flex;  
 justify-content: flex-end;  
 align-items: center;  
 .capsule-item-value {  
 font-size: 12px;  
 transform: translateX(100%);  
 }  
 }  
 }  
 .unit-label {  
 height: 20px;  
 font-size: 12px;  
 position: relative;  
 display: flex;  
 justify-content: space-between;  
 align-items: center;  
 }  
 .unit-text {  
 text-align: right;  
 display: flex;  
 align-items: flex-end;  
 font-size: 12px;  
 line-height: 20px;  
 margin-left: 10px;  
 }  
}  
</style>  
<script setup lang="ts"></script>  
<template>  
 <div>  
 <slot></slot>  
 </div>  
</template>  
<style scoped lang="scss"></style>  
<script setup lang="ts">  
import BorderBox13 from "@/components/datav/border-box-13";  
const props = withDefaults(  
 defineProps<{  
 title: number | string;  
 }>(),  
 {  
 title: "",  
 }  
);  
</script>  
<template>  
 <BorderBox13>  
 <div class="item\_title" v-if="title !== ''">  
 <div class="zuo"></div>  
 <span class="title-inner"> &nbsp;&nbsp;{{ title }}&nbsp;&nbsp; </span>  
 <div class="you"></div>  
 </div>  
 <div  
 :class="title !== '' ? 'item\_title\_content' : 'item\_title\_content\_def'"  
 >  
 <slot></slot></div  
 ></BorderBox13>  
</template>  
<style scoped lang="scss">  
$item-title-height: 38px;  
$item\_title\_content-height: calc(100% - 38px);  
.item\_title {  
 height: $item-title-height;  
 line-height: $item-title-height;  
 width: 100%;  
 color: #31abe3;  
 text-align: center;  
 position: relative;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 .zuo,  
 .you {  
 width: 58px;  
 height: 14px;  
 background-image: url("@/assets/img/titles/zuo.png");  
 }  
 .you {  
 transform: rotate(180deg);  
 }  
 .title-inner {  
 font-weight: 900;  
 letter-spacing: 2px;  
 background: linear-gradient(  
 92deg,  
 #0072ff 0%,  
 #00eaff 48.8525390625%,  
 #01aaff 100%  
 );  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 }  
}  
:deep(.dv-border-box-content) {  
 box-sizing: border-box;  
 padding: 6px 16px 0px;  
 }  
.item\_title\_content {  
 height: $item\_title\_content-height;  
}  
.item\_title\_content\_def {  
 width: 100%;  
 height: 100%;  
}  
</style>  
<template></template>  
<script lang="ts" setup>  
import { ElMessage } from 'element-plus'  
window['$message'] = ElMessage  
</script>  
<template>  
 <section  
 :style="{ ...styles.box, ...boxStyle }"  
 class="v-screen-box"  
 ref="box"  
 >  
 <div  
 :style="{ ...styles.wrapper, ...wrapperStyle }"  
 class="screen-wrapper"  
 ref="screenWrapper"  
 >  
 <slot></slot>  
 </div>  
 </section>  
</template>  
<script lang="ts" setup>  
import { nextTick, onMounted, onUnmounted, reactive, ref, watch } from "vue";  
import type { CSSProperties, PropType } from "vue";  
function debounce(fn: Function, delay: number): () => void {  
 let timer: any;  
 return function (...args: any[]): void {  
 if (timer) clearTimeout(timer);  
 timer = setTimeout(  
 () => {  
 typeof fn === "function" && fn.apply(null, args);  
 clearTimeout(timer);  
 },  
 delay > 0 ? delay : 100  
 );  
 };  
}  
interface IState {  
 originalWidth: string | number;  
 originalHeight: string | number;  
 width?: string | number;  
 height?: string | number;  
 observer: null | MutationObserver;  
}  
type IAutoScale =  
 | boolean  
 | {  
 x?: boolean;  
 y?: boolean;  
 };  
const props = defineProps({  
 width: {  
 type: [String, Number] as PropType<string | number>,  
 default: 1920,  
 },  
 height: {  
 type: [String, Number] as PropType<string | number>,  
 default: 1080,  
 },  
 fullScreen: {  
 type: Boolean as PropType<boolean>,  
 default: false,  
 },  
 autoScale: {  
 type: [Object, Boolean] as PropType<IAutoScale>,  
 default: true,  
 },  
 delay: {  
 type: Number as PropType<number>,  
 default: 500,  
 },  
 boxStyle: {  
 type: Object as PropType<CSSProperties>,  
 default: () => ({}),  
 },  
 wrapperStyle: {  
 type: Object as PropType<CSSProperties>,  
 default: () => ({}),  
 },  
});  
const state = reactive<IState>({  
 width: 0,  
 height: 0,  
 originalWidth: 0,  
 originalHeight: 0,  
 observer: null,  
});  
const styles: Record<string, CSSProperties> = {  
 box: {  
 overflow: "hidden",  
 backgroundSize: `100% 100%`,  
 background: `#000`,  
 width: `100vw`,  
 height: `100vh`,  
 },  
 wrapper: {  
 transitionProperty: `all`,  
 transitionTimingFunction: `cubic-bezier(0.4, 0, 0.2, 1)`,  
 transitionDuration: `500ms`,  
 position: `relative`,  
 overflow: `hidden`,  
 zIndex: 100,  
 transformOrigin: `left top`,  
 },  
};  
const screenWrapper = ref<HTMLElement>();  
const box = ref<HTMLElement>();  
watch(  
 () => props.autoScale,  
 async (newVal: any) => {  
 if (newVal) {  
 onResize();  
 addListener();  
 } else {  
 clearListener();  
 clearScreenWrapperStyle();  
 }  
 }  
);  
const initSize = () => {  
 return new Promise<void>((resolve) => {  
 box.value!.scrollLeft = 0;  
 box.value!.scrollTop = 0;  
 nextTick(() => {  
 if (props.width && props.height) {  
 state.width = props.width;  
 state.height = props.height;  
 } else {  
 state.width = screenWrapper.value?.clientWidth;  
 state.height = screenWrapper.value?.clientHeight;  
 }  
 if (!state.originalHeight || !state.originalWidth) {  
 state.originalWidth = window.screen.width;  
 state.originalHeight = window.screen.height;  
 }  
 resolve();  
 });  
 });  
};  
const updateSize = () => {  
 if (state.width && state.height) {  
 screenWrapper.value!.style.width = `${state.width}px`;  
 screenWrapper.value!.style.height = `${state.height}px`;  
 } else {  
 screenWrapper.value!.style.width = `${state.originalWidth}px`;  
 screenWrapper.value!.style.height = `${state.originalHeight}px`;  
 }  
};  
const clearScreenWrapperStyle = () => {  
 screenWrapper.value!.style.transform = "";  
 screenWrapper.value!.style.margin = "";  
};  
const autoScale = (scale: number) => {  
 if (!props.autoScale) {  
 return;  
 }  
 const domWidth = screenWrapper.value!.clientWidth;  
 const domHeight = screenWrapper.value!.clientHeight;  
 const currentWidth = document.body.clientWidth;  
 const currentHeight = document.body.clientHeight;  
 screenWrapper.value!.style.transform = `scale(${scale},${scale})`;  
 let mx = Math.max((currentWidth - domWidth \* scale) / 2, 0);  
 let my = Math.max((currentHeight - domHeight \* scale) / 2, 0);  
 if (typeof props.autoScale === "object") {  
 !props.autoScale.x && (mx = 0);  
 !props.autoScale.y && (my = 0);  
 }  
 screenWrapper.value!.style.margin = `${my}px ${mx}px`;  
};  
const updateScale = () => {  
 const currentWidth = document.body.clientWidth;  
 const currentHeight = document.body.clientHeight;  
 const realWidth = state.width || state.originalWidth;  
 const realHeight = state.height || state.originalHeight;  
 const widthScale = currentWidth / +realWidth;  
 const heightScale = currentHeight / +realHeight;  
 if (props.fullScreen) {  
 screenWrapper.value!.style.transform = `scale(${widthScale},${heightScale})`;  
 return false;  
 }  
 const scale = Math.min(widthScale, heightScale);  
 autoScale(scale);  
};  
const onResize = debounce(async () => {  
 await initSize();  
 updateSize();  
 updateScale();  
}, props.delay);  
const initMutationObserver = () => {  
 const observer = (state.observer = new MutationObserver(() => {  
 onResize();  
 }));  
 observer.observe(screenWrapper.value!, {  
 attributes: true,  
 attributeFilter: ["style"],  
 attributeOldValue: true,  
 });  
};  
const clearListener = () => {  
 window.removeEventListener("resize", onResize);  
};  
const addListener = () => {  
 window.addEventListener("resize", onResize);  
};  
onMounted(() => {  
 nextTick(async () => {  
 await initSize();  
 updateSize();  
 updateScale();  
 addListener();  
 });  
});  
onUnmounted(() => {  
 clearListener();  
});  
</script>  
<script setup lang="ts">  
import {  
 computed,  
 defineComponent,  
 onBeforeMount,  
 onMounted,  
 ref,  
 watch,  
 nextTick,  
} from "vue";  
import type { CSSProperties } from "vue";  
import throttle from "lodash/throttle";  
type propsType = {  
 modelValue?: boolean;  
 list: Array<any>;  
 step?: number;  
 limitScrollNum?: number;  
 hover?: boolean;  
 direction?: string;  
 singleHeight?: number;  
 singleWidth?: number;  
 singleWaitTime?: number;  
 isRemUnit?: boolean;  
 isWatch?: boolean;  
 delay?: number;  
 ease?: any;  
 count?: number;  
 copyNum?: number;  
 wheel?: boolean;  
 singleLine?: boolean;  
};  
const props = withDefaults(defineProps<propsType>(), {  
 modelValue: true,  
 list: () => [],  
 step: 1,  
 limitScrollNum: 3,  
 hover: false,  
 direction: "up",  
 singleHeight: 0,  
 singleWidth: 0,  
 singleWaitTime: 1000,  
 isRemUnit: false,  
 isWatch: true,  
 delay: 0,  
 ease: "ease-in",  
 count: -1,  
 copyNum: 1,  
 wheel: false,  
 singleLine: false,  
});  
interface Emits {  
 (event: "count", \_count: number): void;  
 (event: "stop", \_count: number): void;  
}  
const emit = defineEmits<Emits>();  
const scrollRef = ref(null);  
const slotListRef = ref<HTMLDivElement | null>(null);  
const realBoxRef = ref<HTMLDivElement | null>(null);  
const reqFrame = ref<number | null>(null);  
const singleWaitTimeout = ref<TimeProp | null>(null);  
const realBoxWidth = ref(0);  
const realBoxHeight = ref(0);  
const xPos = ref(0);  
const yPos = ref(0);  
const isHover = ref(false);  
const \_count = ref(0);  
const isScroll = computed(() =>  
 props.list ? props.list.length >= props.limitScrollNum : false  
);  
const realBoxStyle = computed(() => {  
 return {  
 width: realBoxWidth.value ? `${realBoxWidth.value}px` : "auto",  
 transform: `translate(${xPos.value}px,${yPos.value}px)`,  
 transition: `all ${  
 typeof props.ease === "string"  
 ? props.ease  
 : "cubic-bezier(" +  
 props.ease.x1 +  
 "," +  
 props.ease.y1 +  
 "," +  
 props.ease.x2 +  
 "," +  
 props.ease.y2 +  
 ")"  
 } ${props.delay}ms`,  
 overflow: "hidden",  
 display: props.singleLine ? "flex" : "block",  
 };  
});  
const isHorizontal = computed(  
 () => props.direction == "left" || props.direction == "right"  
);  
function dataWarm(list: any) {  
 if (list && typeof list !== "boolean" && list.length > 100) {  
 console.warn(  
 `数据达到了${list.length}条有点多哦~,可能会造成部分老旧浏览器卡顿。`  
 );  
 }  
}  
const floatStyle = computed<CSSProperties>(() => {  
 return isHorizontal.value  
 ? {  
 float: "left",  
 overflow: "hidden",  
 display: props.singleLine ? "flex" : "block",  
 flexShrink: props.singleLine ? 0 : 1,  
 }  
 : { overflow: "hidden" };  
});  
const baseFontSize = computed(() => {  
 return props.isRemUnit  
 ? parseInt(  
 globalThis.window.getComputedStyle(  
 globalThis.document.documentElement,  
 null  
 ).fontSize  
 )  
 : 1;  
});  
const realSingleStopWidth = computed(  
 () => props.singleWidth \* baseFontSize.value  
);  
const realSingleStopHeight = computed(  
 () => props.singleHeight \* baseFontSize.value  
);  
const step = computed(() => {  
 let singleStep: number;  
 let \_step = props.step;  
 if (isHorizontal.value) {  
 singleStep = realSingleStopWidth.value;  
 } else {  
 singleStep = realSingleStopHeight.value;  
 }  
 if (singleStep > 0 && singleStep % \_step > 0) {  
 console.error(  
 "如果设置了单步滚动，step 需是单步大小的约数，否则无法保证单步滚动结束的位置是否准确。~~~~~"  
 );  
 }  
 return \_step;  
});  
const cancle = () => {  
 cancelAnimationFrame(reqFrame.value as number);  
 reqFrame.value = null;  
};  
const animation = (  
 \_direction: "up" | "down" | "left" | "right",  
 \_step: number,  
 isWheel?: boolean  
) => {  
 reqFrame.value = requestAnimationFrame(function () {  
 const h = realBoxHeight.value / 2;  
 const w = realBoxWidth.value / 2;  
 if (\_direction === "up") {  
 if (Math.abs(yPos.value) >= h) {  
 yPos.value = 0;  
 \_count.value += 1;  
 emit("count", \_count.value);  
 }  
 yPos.value -= \_step;  
 } else if (\_direction === "down") {  
 if (yPos.value >= 0) {  
 yPos.value = h \* -1;  
 \_count.value += 1;  
 emit("count", \_count.value);  
 }  
 yPos.value += \_step;  
 } else if (\_direction === "left") {  
 if (Math.abs(xPos.value) >= w) {  
 xPos.value = 0;  
 \_count.value += 1;  
 emit("count", \_count.value);  
 }  
 xPos.value -= \_step;  
 } else if (\_direction === "right") {  
 if (xPos.value >= 0) {  
 xPos.value = w \* -1;  
 \_count.value += 1;  
 emit("count", \_count.value);  
 }  
 xPos.value += \_step;  
 }  
 if (isWheel) {  
 return;  
 }  
 let { singleWaitTime } = props;  
 if (singleWaitTimeout.value) {  
 clearTimeout(singleWaitTimeout.value);  
 }  
 if (!!realSingleStopHeight.value) {  
 if (Math.abs(yPos.value) % realSingleStopHeight.value < \_step) {  
 singleWaitTimeout.value = setTimeout(() => {  
 move();  
 }, singleWaitTime);  
 } else {  
 move();  
 }  
 } else if (!!realSingleStopWidth.value) {  
 if (Math.abs(xPos.value) % realSingleStopWidth.value < \_step) {  
 singleWaitTimeout.value = setTimeout(() => {  
 move();  
 }, singleWaitTime);  
 } else {  
 move();  
 }  
 } else {  
 move();  
 }  
 });  
};  
const move = () => {  
 cancle();  
 if (isHover.value || !isScroll.value || \_count.value === props.count) {  
 emit("stop", \_count.value);  
 \_count.value = 0;  
 return;  
 }  
 animation(  
 props.direction as "up" | "down" | "left" | "right",  
 step.value,  
 false  
 );  
};  
const initMove = () => {  
 dataWarm(props.list);  
 if (isHorizontal.value) {  
 let slotListWidth = (slotListRef.value as HTMLDivElement).offsetWidth;  
 slotListWidth = slotListWidth \* 2 + 1;  
 realBoxWidth.value = slotListWidth;  
 }  
 if (isScroll.value) {  
 realBoxHeight.value = (realBoxRef.value as HTMLDivElement).offsetHeight;  
 if (props.modelValue) {  
 move();  
 }  
 } else {  
 cancle();  
 yPos.value = xPos.value = 0;  
 }  
};  
const startMove = () => {  
 isHover.value = false;  
 move();  
};  
const stopMove = () => {  
 isHover.value = true;  
 if (singleWaitTimeout.value) {  
 clearTimeout(singleWaitTimeout.value);  
 }  
 cancle();  
};  
const hoverStop = computed(  
 () => props.hover && props.modelValue && isScroll.value  
);  
const throttleFunc = throttle((e: WheelEvent) => {  
 cancle();  
 const singleHeight = !!realSingleStopHeight.value  
 ? realSingleStopHeight.value  
 : 15;  
 if (e.deltaY < 0) {  
 animation("down", singleHeight, true);  
 }  
 if (e.deltaY > 0) {  
 animation("up", singleHeight, true);  
 }  
}, 30);  
const onWheel = (e: WheelEvent) => {  
 throttleFunc(e);  
};  
const reset = () => {  
 cancle();  
 isHover.value = false;  
 initMove();  
};  
const Reset = () => {  
 reset();  
};  
defineExpose({  
 Reset,  
});  
watch(  
 () => props.list,  
 () => {  
 if (props.isWatch) {  
 nextTick(() => {  
 reset();  
 });  
 }  
 },  
 {  
 deep: true,  
 }  
);  
watch(  
 () => props.modelValue,  
 (newValue) => {  
 if (newValue) {  
 startMove();  
 } else {  
 stopMove();  
 }  
 }  
);  
watch(  
 () => props.count,  
 (newValue) => {  
 if (newValue !== 0) {  
 startMove();  
 }  
 }  
);  
onBeforeMount(() => {  
 cancle();  
 clearTimeout(singleWaitTimeout.value as unknown as number);  
});  
onMounted(() => {  
 if (isScroll.value) {  
 initMove();  
 }  
});  
</script>  
<template>  
 <div  
 v-if="props.wheel && props.hover"  
 ref="realBoxRef"  
 :style="realBoxStyle"  
 @mouseenter="  
 () => {  
 hoverStop && stopMove();  
 }  
 "  
 @mouseleave="  
 () => {  
 hoverStop && startMove();  
 }  
 "  
 @wheel="  
 (e) => {  
 hoverStop && onWheel(e);  
 }  
 "  
 >  
 <div ref="slotListRef" :style="floatStyle">  
 <slot></slot>  
 </div>  
 <div :style="floatStyle">  
 <slot></slot>  
 </div>  
 </div>  
 <div  
 v-else  
 :style="realBoxStyle"  
 ref="realBoxRef"  
 @mouseenter="  
 () => {  
 hoverStop && stopMove();  
 }  
 "  
 @mouseleave="  
 () => {  
 hoverStop && startMove();  
 }  
 "  
 >  
 <div ref="slotListRef" :style="floatStyle">  
 <slot></slot>  
 </div>  
 <div :style="floatStyle">  
 <slot></slot>  
 </div>  
 </div>  
</template>  
<style scoped lang="scss"></style>  
<script setup lang="ts">  
import { reactive } from "vue";  
import dayjs from 'dayjs';  
import type {DateDataType} from "./index.d"  
import {useSettingStore} from "@/stores/index"  
const dateData = reactive<DateDataType>({  
 dateDay: "",  
 dateYear: "",  
 dateWeek: "",  
 timing:null  
});  
const { setSettingShow} =useSettingStore()  
const weekday= ["周日", "周一", "周二", "周三", "周四", "周五", "周六"]  
const timeFn = () => {  
 dateData.timing = setInterval(() => {  
 dateData.dateDay = dayjs().format("YYYY-MM-DD hh : mm : ss");  
 dateData.dateWeek = weekday[dayjs().day()];  
 }, 1000);  
};  
timeFn()  
</script>  
<template>  
 <div class="d-flex jc-center title\_wrap">  
 <div class="zuojuxing"></div>  
 <div class="youjuxing"></div>  
 <div class="guang"></div>  
 <div class="d-flex jc-center">  
 <div class="title">  
 <span class="title-text">FADEC</span>  
 </div>  
 </div>  
 <div class="timers">  
 {{ dateData.dateYear }} {{ dateData.dateWeek }} {{ dateData.dateDay }}  
 <div class="setting\_icon" @click="setSettingShow(true)">  
 <img src="@/assets/img/headers/setting.png" alt="设置">  
 </div>  
 </div>  
 </div>  
</template>  
<style scoped lang="scss">  
.title\_wrap {  
 height: 60px;  
 background-image: url("../assets/img/top.png");  
 background-size: cover;  
 background-position: center center;  
 position: relative;  
 margin-bottom: 4px;  
 .guang {  
 position: absolute;  
 bottom: -26px;  
 background-image: url("../assets/img/guang.png");  
 background-position: 80px center;  
 width: 100%;  
 height: 56px;  
 }  
 .zuojuxing,  
 .youjuxing {  
 position: absolute;  
 top: -2px;  
 width: 140px;  
 height: 6px;  
 background-image: url("../assets/img/headers/juxing1.png");  
 }  
 .zuojuxing {  
 left: 11%;  
 }  
 .youjuxing {  
 right: 11%;  
 transform: rotate(180deg);  
 }  
 .timers {  
 position: absolute;  
 right: 0;  
 top: 30px;  
 font-size: 18px;  
 display: flex;  
 align-items: center;  
 .setting\_icon {  
 width: 20px;  
 height: 20px;  
 cursor: pointer;  
 margin-left: 12px;  
 img{  
 width: 100%;  
 height: 100%;  
 }  
 }  
 }  
}  
.title {  
 position: relative;  
 text-align: center;  
 background-size: cover;  
 color: transparent;  
 height: 60px;  
 line-height: 46px;  
 .title-text {  
 font-size: 38px;  
 font-weight: 900;  
 letter-spacing: 6px;  
 width: 100%;  
 background: linear-gradient(  
 92deg,  
 #0072ff 0%,  
 #00eaff 48.8525390625%,  
 #01aaff 100%  
 );  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 }  
}  
</style>  
<script setup lang="ts">  
import { ref } from "vue";  
import { RouterView } from "vue-router";  
import ScaleScreen from "@/components/scale-screen";  
import Headers from "./header.vue";  
import Setting from "./setting.vue";  
import { useSettingStore } from "@/stores/index";  
import { storeToRefs } from "pinia";  
import MessageContent from "@/components/Plugins/MessageContent";  
const settingStore = useSettingStore();  
const { isScale } = storeToRefs(settingStore);  
const wrapperStyle = {};  
</script>  
<template>  
 <scale-screen  
 width="1920"  
 height="1080"  
 :delay="500"  
 :fullScreen="false"  
 :boxStyle="{  
 background: '#03050C',  
 overflow: isScale ? 'hidden' : 'auto',  
 }"  
 :wrapperStyle="wrapperStyle"  
 :autoScale="isScale"  
 >  
 <div class="content\_wrap">  
 <Headers />  
 <RouterView />  
 <MessageContent />  
 </div>  
 </scale-screen>  
 <Setting />  
</template>  
<style lang="scss" scoped>  
.content\_wrap {  
 width: 100%;  
 height: 100%;  
 padding: 16px 16px 16px 16px;  
 box-sizing: border-box;  
 background-image: url("@/assets/img/pageBg.png");  
 background-size: cover;  
 background-position: center center;  
}  
</style>  
<script setup lang="ts">  
import { useSettingStore } from "@/stores/index";  
import { ref } from "vue";  
import {storeToRefs} from "pinia"  
const isScaleRadio = ref(false);  
const leftBottomRadio=ref(true)  
const rightBottomRadio=ref(true)  
const modelRotate=ref(true) // 模型是否旋转  
const model=ref(true) // 模型是否旋转  
const settingStore = useSettingStore();  
const {indexConfig}=storeToRefs(settingStore)  
const init = () => {  
 settingStore.initSetting();  
 isScaleRadio.value = settingStore.isScale;  
 leftBottomRadio.value=indexConfig.value.leftBottomSwiper  
 rightBottomRadio.value=indexConfig.value.rightBottomSwiper  
 modelRotate.value=indexConfig.value.modelRotateSpeed  
};  
init();  
const handleClose = () => {};  
const cancelClick = () => {  
 settingStore.setSettingShow(false);  
};  
const confirmClick = () => {};  
const isScaleChange = (flag: boolean) => {  
 settingStore.setIsScale(flag);  
};  
const radiochange = (blag: boolean) => {  
 settingStore.setIsScale(blag);  
};  
const indexRadioChange=(flag: boolean)=>{  
 settingStore.setIndexConfig({  
 leftBottomSwiper: leftBottomRadio.value,//左轮播  
 rightBottomSwiper:rightBottomRadio.value,//右下轮播  
 modelRotateSpeed: modelRotate.value,//模型旋转速度  
 });  
}  
</script>  
<template>  
 <el-drawer v-model="settingStore.settingShow" direction="rtl" size="360px">  
 <template #header>  
 <h2 class="setting-title">设置</h2>  
 </template>  
 <template #default>  
 <div class="left\_shu">全局设置</div>  
 <div class="setting\_item">  
 <span class="setting\_label">  
 是否进行自动适配<span class="setting\_label\_tip"  
 >(默认分辨率1920\*1080)</span  
 >:  
 </span>  
 <div class="setting\_content">  
 <el-radio-group v-model="isScaleRadio" @change="(flag)=>isScaleChange(flag as boolean)">  
 <el-radio :label="true">是</el-radio>  
 <el-radio :label="false">否</el-radio>  
 </el-radio-group>  
 </div>  
 </div>  
 <div class="left\_shu">实时监测</div>  
 <div class="setting\_item">  
 <span class="setting\_label">  
 传感器设备自动轮询: <span class="setting\_label\_tip"></span>  
 </span>  
 <div class="setting\_content">  
 <el-radio-group  
 v-model="leftBottomRadio"  
 @change="(flag)=>indexRadioChange(flag as boolean)"  
 >  
 <el-radio :label="true">是</el-radio>  
 <el-radio :label="false">否</el-radio>  
 </el-radio-group>  
 </div>  
 </div>  
 <div class="setting\_item">  
 <span class="setting\_label"> 执行器设备列表轮播: </span>  
 <div class="setting\_content">  
 <el-radio-group  
 v-model="rightBottomRadio"  
 @change="(flag)=>indexRadioChange(flag as boolean)"  
 >  
 <el-radio :label="true">是</el-radio>  
 <el-radio :label="false">否</el-radio>  
 </el-radio-group>  
 </div>  
 </div>  
 <div class="left\_shu">模型菜单</div>  
 <div class="setting\_item">  
 <span class="setting\_label">  
 模型是否旋转: <span class="setting\_label\_tip"></span>  
 </span>  
 <div class="setting\_content">  
 <el-radio-group  
 v-model="modelRotate"  
 @change="(flag)=>indexRadioChange(flag as boolean)"  
 >  
 <el-radio :label="true">是</el-radio>  
 <el-radio :label="false">否</el-radio>  
 </el-radio-group>  
 </div>  
 </div>  
 <div class="setting\_item">  
 <span class="setting\_label">  
 模型材质: <span class="setting\_label\_tip"></span>  
 </span>  
 <div class="setting\_content">  
 <el-radio-group  
 v-model="modelRotate"  
 @change="(flag)=>indexRadioChange(flag as boolean)"  
 >  
 <el-radio :label="true">是</el-radio>  
 <el-radio :label="false">否</el-radio>  
 </el-radio-group>  
 </div>  
 </div>  
 </template>  
<script setup lang="ts">  
import { ref, reactive, onMounted, nextTick, onUnmounted } from "vue";  
import { installationPlan } from "@/api";  
import { graphic } from "echarts/core";  
import { ElMessage } from "element-plus";  
const option = ref({});  
const msg = ref({  
 "category": [  
 "08-01", "08-02", "08-03", "08-04", "08-05", "08-06",  
 "08-07", "08-08", "08-09", "08-10", "08-11", "08-12", "08-13",  
 "08-14", "08-15", "08-16", "08-17", "08-18", "08-19", "08-20",  
 "08-21", "08-22", "08-23", "08-24", "08-25", "08-26"  
 ],  
 "barData": [  
 89, 93, 82, 95, 54, 12, 72,  
 93, 13, 14, 54, 68, 30, 80,  
 23, 15, 27, 20, 61, 30, 81,  
 37, 79, 64, 64, 49, 36, 62,  
 88, 49, 57, 57  
 ],  
 "lineData": [  
 179, 170, 101, 173, 128, 104, 96, 176,  
 70, 111, 81, 155, 35, 149, 89, 111, 58,  
 78, 149, 53, 47, 121, 141, 70, 138, 95, 52,  
 95, 114, 69, 106, 120  
 ],  
 "rateData": [  
 "50", "55", "81", "55", "42",  
 "12", "75", "53", "19", "13",  
 "67", "44", "86", "54", "26",  
 "14", "47", "26", "41", "57",  
 "55", "31", "56", "91", "46",  
 "52", "69", "65", "77", "71",  
 "54", "48"  
 ]  
});  
const updateData = () => {  
 const now = new Date();  
 const newDate = now.toLocaleDateString('default', { month: '2-digit', day: '2-digit' });  
 const lastTime = new Date(`2024-08-${msg.value.category[msg.value.category.length - 1].replace('-', '')}`);  
 if (now < lastTime) {  
 setTimeout(updateData, 24 \* 60 \* 60 \* 1000 - (now.getTime() - lastTime.getTime()));  
 return;  
 }  
 msg.value.category.push(newDate);  
 msg.value.category.shift(); // 移除第一个元素以保持数组长度不变  
 const barData = msg.value.barData.map(num => Math.round(num \* (1 + (Math.random() \* 20 - 10) / 100)));  
 const lineData = msg.value.lineData.map(num => Math.round(num \* (1 + (Math.random() \* 20 - 10) / 100)));  
 const rateData = msg.value.rateData.map(num => Math.round(num \* (1 + (Math.random() \* 20 - 10) / 100)));  
 msg.value.barData = barData;  
 msg.value.lineData = lineData;  
 msg.value.rateData = rateData;  
};  
const getData = () => {  
 installationPlan()  
 .then((res) => {  
 console.log("中下--风险演化分析", res);  
 if (res.success) {  
 setOption(msg.value);  
 } else {  
 ElMessage({  
 message: res.msg,  
 type: "warning",  
 });  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const setOption = async (newData: any) => {  
 option.value = {  
 tooltip: {  
 trigger: "axis",  
 backgroundColor: "rgba(0,0,0,.6)",  
 borderColor: "rgba(147, 235, 248, .8)",  
 textStyle: {  
 color: "#FFF",  
 },  
 formatter: function (params: any) {  
 var result = params[0].name + "<br>";  
 params.forEach(function (item: any) {  
 if (item.value) {  
 if (item.seriesName == "指标3") {  
 result += item.marker + " " + item.seriesName + " : " + item.value + "%</br>";  
 } else {  
 result += item.marker + " " + item.seriesName + " : " + item.value + "个</br>";  
 }  
 } else {  
 result += item.marker + " " + item.seriesName + " : - </br>";  
 }  
 });  
 return result;  
 },  
 },  
 legend: {  
 data: ["指标1", "指标2", "指标3"],  
 textStyle: {  
 color: "#B4B4B4",  
 },  
 top: "0",  
 },  
 grid: {  
 left: "50px",  
 right: "40px",  
 bottom: "30px",  
 top: "20px",  
 },  
 xAxis: {  
 data: newData.category,  
 axisLine: {  
 lineStyle: {  
 color: "#B4B4B4",  
 },  
 },  
 axisTick: {  
 show: false,  
 },  
 },  
 yAxis: [  
 {  
 splitLine: { show: false },  
 axisLine: {  
 lineStyle: {  
 color: "#B4B4B4",  
 },  
 },  
 axisLabel: {  
 formatter: "{value}",  
 },  
 },  
 {  
 splitLine: { show: false },  
 axisLine: {  
 lineStyle: {  
 color: "#B4B4B4",  
 },  
 },  
 axisLabel: {  
 formatter: "{value}% ",  
 },  
 },  
 ],  
 series: [  
 {  
 name: "指标1",  
 type: "bar",  
 barWidth: 10,  
 itemStyle: {  
 borderRadius: 5,  
 color: new graphic.LinearGradient(0, 0, 0, 1, [  
 { offset: 0, color: "#956FD4" },  
 { offset: 1, color: "#3EACE5" },  
 ]),  
 },  
 data: newData.barData,  
 },  
 {  
 name: "指标2",  
 type: "bar",  
 barGap: "-100%",  
 barWidth: 10,  
 itemStyle: {  
 borderRadius: 5,  
 color: new graphic.LinearGradient(0, 0, 0, 1, [  
 { offset: 0, color: "rgba(156,107,211,0.8)" },  
 { offset: 0.2, color: "rgba(156,107,211,0.5)" },  
 { offset: 1, color: "rgba(156,107,211,0.2)" },  
 ]),  
 },  
 z: -12,  
 data: newData.lineData,  
 },  
 {  
 name: "指标3",  
 type: "line",  
 smooth: true,  
 showAllSymbol: true,  
 symbol: "emptyCircle",  
 symbolSize: 8,  
 yAxisIndex: 1,  
 itemStyle: {  
 color: "#F02FC2",  
 },  
 data: newData.rateData,  
 },  
 ],  
 };  
};  
onMounted(() => {  
 const timer = setInterval(updateData, 2000); // 每隔2秒更新数据  
 onUnmounted(() => clearInterval(timer));  
 getData();  
});  
</script>  
<template>  
 <v-chart class="chart" :option="option" v-if="JSON.stringify(option) != '{}'" />  
</template>  
<style scoped lang="scss"></style>  
<script setup lang="ts">  
import { ref, reactive, nextTick } from "vue";  
import { centerMap, GETNOBASE } from "@/api";  
import { registerMap, getMap } from "echarts/core";  
import { optionHandle, regionCodes } from "./center.map";  
import BorderBox13 from "@/components/datav/border-box-13";  
import { ElMessage } from "element-plus";  
import type { MapdataType } from "./center.map";  
const option = ref({});  
const code = ref("china");

withDefaults(  
 defineProps<{  
 title: number | string;  
 }>(),  
 {  
 title: "地图",  
 }  
);  
const dataSetHandle = async (regionCode: string, list: object[]) => {  
 const geojson: any = await getGeojson(regionCode);  
 let cityCenter: any = {};  
 let mapData: MapdataType[] = [];  
 geojson.features.forEach((element: any) => {  
 cityCenter[element.properties.name] = element.properties.centroid || element.properties.center;  
 });  
 list.forEach((item: any) => {  
 if (cityCenter[item.name]) {  
 mapData.push({  
 name: item.name,  
 value: cityCenter[item.name].concat(item.value),  
 });  
 }  
 });  
 await nextTick();  
 option.value = optionHandle(regionCode, list, mapData);  
};  
const getData = async (regionCode: string) => {  
 centerMap({ regionCode: regionCode })  
 .then((res) => {  
 console.log("中上--传感器分布图", res);  
 if (res.success) {  
 dataSetHandle(res.data.regionCode, res.data.dataList);  
 } else {  
 ElMessage.error(res.msg);  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const getGeojson = (regionCode: string) => {  
 return new Promise<boolean>(async (resolve) => {  
 let mapjson = getMap(regionCode);  
 if (mapjson) {  
 mapjson = mapjson.geoJSON;  
 resolve(mapjson);  
 } else {  
 mapjson = await GETNOBASE(`./map-geojson/${regionCode}.json`).then((data) => data);  
 code.value = regionCode;  
 registerMap(regionCode, {  
 geoJSON: mapjson as any,  
 specialAreas: {},  
 });  
 resolve(mapjson);  
 }  
 });  
};  
getData(code.value);  
const mapClick = (params: any) => {  
 let xzqData = regionCodes[params.name];  
 if (xzqData) {  
 getData(xzqData.adcode);  
 } else {  
 window["$message"].warning("暂无下级地市");  
 }  
};  
</script>  
<template>  
 <div class="centermap">  
 <div class="maptitle">  
 <div class="zuo"></div>  
 <span class="titletext">{{ title }}</span>  
 <div class="you"></div>  
 </div>  
 <div class="mapwrap">  
 <BorderBox13>  
 <div class="quanguo" @click="getData('china')" v-if="code !== 'china'">中国</div>  
 <v-chart  
 class="chart"  
 :option="option"  
 ref="centerMapRef"  
 @click="mapClick"  
 v-if="JSON.stringify(option) != '{}'"  
 />  
 </BorderBox13>  
 </div>  
 </div>  
</template>  
<style scoped lang="scss">  
.centermap {  
 margin-bottom: 30px;  
 .maptitle {  
 height: 60px;  
 display: flex;  
 justify-content: center;  
 padding-top: 10px;  
 box-sizing: border-box;  
 .titletext {  
 font-size: 28px;  
 font-weight: 900;  
 letter-spacing: 6px;  
 background: linear-gradient(92deg, #0072ff 0%, #00eaff 48.8525390625%, #01aaff 100%);  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 margin: 0 10px;  
 }  
 .zuo,  
 .you {  
 background-size: 100% 100%;  
 width: 29px;  
 height: 20px;  
 margin-top: 8px;  
 }  
 .zuo {  
 background: url("@/assets/img/xiezuo.png") no-repeat;  
 }  
 .you {  
 background: url("@/assets/img/xieyou.png") no-repeat;  
 }  
 }  
 .mapwrap {  
 height: 580px;  
 width: 100%;  
 box-sizing: border-box;  
 position: relative;  
 .quanguo {  
 position: absolute;  
 right: 20px;  
 top: -46px;  
 width: 80px;  
 height: 28px;  
 border: 1px solid #00eded;  
 border-radius: 10px;  
 color: #00f7f6;  
 text-align: center;  
 line-height: 26px;  
 letter-spacing: 6px;  
 cursor: pointer;  
 box-shadow: 0 2px 4px rgba(0, 237, 237, 0.5), 0 0 6px rgba(0, 237, 237, 0.4);  
 z-index: 10;  
 }  
 }  
}  
</style>  
<script setup lang="ts">  
import ItemWrap from "@/components/item-wrap";  
import LeftTop from "./left-top.vue";  
import LeftCenter from "./left-center.vue";  
import LeftBottom from "./left-bottom.vue";  
import CenterMap from "./center-map.vue";  
import CenterModel from "@/views/index/center-model/index.vue"  
import CenterBottom from "./center-bottom.vue";  
import RightTop from "./right-top.vue";  
import RightCenter from "./right-center.vue";  
import RightBottom from "./right-bottom.vue";  
</script>  
<template>  
 <div class="index-box">  
 <div class="contetn\_left">  
<script setup lang="ts">  
import { leftBottom } from "@/api";  
import SeamlessScroll from "@/components/seamless-scroll";  
import { computed, onMounted, reactive } from "vue";  
import { useSettingStore } from "@/stores";  
import { storeToRefs } from "pinia";  
import EmptyCom from "@/components/empty-com";  
import { ElMessage } from "element-plus";  
const settingStore = useSettingStore();  
const { defaultOption, indexConfig } = storeToRefs(settingStore);  
const state = reactive<any>({  
 list: [],  
 defaultOption: {  
 ...defaultOption.value,  
 singleHeight: 256,  
 limitScrollNum: 4,  
 },  
 scroll: true,  
});  
const getData = () => {  
 leftBottom( { limitNum: 20 })  
 .then((res) => {  
 console.log("左下--设备提醒", res);  
 if (res.success) {  
 state.list = res.data.list;  
 } else {  
 ElMessage({  
 message: res.msg,  
 type: "warning",  
 });  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const addressHandle = (item: any) => {  
 let name = item.provinceName;  
 if (item.cityName) {  
 name += "/" + item.cityName;  
 if (item.countyName) {  
 name += "/" + item.countyName;  
 }  
 }  
 return name;  
};  
const comName = computed(() => {  
 if (indexConfig.value.leftBottomSwiper) {  
 return SeamlessScroll;  
 } else {  
 return EmptyCom;  
 }  
});  
onMounted(() => {  
 getData();  
});  
</script>  
<template>  
 <div class="left\_boottom\_wrap beautify-scroll-def" :class="{ 'overflow-y-auto': !indexConfig.leftBottomSwiper }">  
 <component  
 :is="comName"  
 :list="state.list"  
 v-model="state.scroll"  
 :singleHeight="state.defaultOption.singleHeight"  
 :step="state.defaultOption.step"  
 :limitScrollNum="state.defaultOption.limitScrollNum"  
 :hover="state.defaultOption.hover"  
 :singleWaitTime="state.defaultOption.singleWaitTime"  
 :wheel="state.defaultOption.wheel"  
 >  
 <ul class="left\_boottom">  
 <li class="left\_boottom\_item" v-for="(item, i) in state.list" :key="i">  
 <span class="orderNum doudong">{{ i + 1 }}</span>  
 <div class="inner\_right">  
 <div class="dibu"></div>  
 <div class="flex">  
 <div class="info">  
 <span class="labels">设备ID：</span>  
 <span class="text-content zhuyao doudong wangguan"> {{ item.gatewayno }}</span>  
 </div>  
 <div class="info">  
 <span class="labels">时间：</span>  
 <span class="text-content" style="font-size: 12px"> {{ item.createTime }}</span>  
 </div>  
 </div>  
 <span  
 class="types doudong"  
 :class="{  
 typeRed: item.onlineState == 0,  
 typeGreen: item.onlineState == 1,  
 }"  
 >{{ item.onlineState == 1 ? "上线" : "下线" }}</span  
 >  
 <div class="info addresswrap">  
 <span class="labels">设备位置：</span>  
 <span class="text-content ciyao" style="font-size: 12px"> 设备位置 </span>  
 </div>  
 </div>  
 </li>  
 </ul>  
 </component>  
 </div>  
</template>  
<style scoped lang="scss">  
.left\_boottom\_wrap {  
 overflow: hidden;  
 width: 100%;  
 height: 100%;  
}  
.doudong {  
 overflow: hidden;  
 backface-visibility: hidden;  
}  
.overflow-y-auto {  
 overflow-y: auto;  
}  
.left\_boottom {  
 width: 100%;  
 height: 100%;  
 .left\_boottom\_item {  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 padding: 8px;  
 font-size: 14px;  
 margin: 10px 0;  
 .orderNum {  
 margin: 0 16px 0 -20px;  
 }  
 .info {  
 margin-right: 10px;  
 display: flex;  
 align-items: center;  
 color: #fff;  
 .labels {  
 flex-shrink: 0;  
 font-size: 12px;  
 color: rgba(255, 255, 255, 0.6);  
 }  
 .zhuyao {  
 color: $primary-color;  
 font-size: 15px;  
 }  
 .ciyao {  
 color: rgba(255, 255, 255, 0.8);  
 }  
 .warning {  
 color: #e6a23c;  
 font-size: 15px;  
 }  
 }  
 .inner\_right {  
 position: relative;  
 height: 100%;  
 width: 380px;  
 flex-shrink: 0;  
 line-height: 1;  
 display: flex;  
 align-items: center;  
 justify-content: space-between;  
 flex-wrap: wrap;  
 .dibu {  
 position: absolute;  
 height: 2px;  
 width: 104%;  
 background-image: url("@/assets/img/zuo\_xuxian.png");  
 bottom: -10px;  
 left: -2%;  
 background-size: cover;  
 }  
 .addresswrap {  
 width: 100%;  
 display: flex;  
 margin-top: 8px;  
 }  
 }  
 .wangguan {  
 color: #1890ff;  
 font-weight: 900;  
 font-size: 15px;  
 width: 80px;  
 flex-shrink: 0;  
 }  
 .time {  
 font-size: 12px;  
 color: #fff;  
 }  
 .address {  
 font-size: 12px;  
 cursor: pointer;  
 }  
 .types {  
 width: 30px;  
 flex-shrink: 0;  
 }  
 .typeRed {  
 color: #fc1a1a;  
 }  
 .typeGreen {  
 color: #29fc29;  
 }  
 }  
}  
</style>  
<script setup lang="ts">  
import { ref, reactive } from "vue";  
import { graphic } from "echarts/core";  
import { countUserNum } from "@/api";  
import {ElMessage} from "element-plus"  
let colors = ["#0BFC7F", "#A0A0A0", "#F48C02", "#F4023C"];  
const option = ref({});  
const state = reactive({  
 lockNum: 0,  
 offlineNum: 0,  
 onlineNum: 0,  
 alarmNum: 0,  
 totalNum: 0,  
});  
const echartsGraphic = (colors: string[]) => {  
 return new graphic.LinearGradient(1, 0, 0, 0, [  
 { offset: 0, color: colors[0] },  
 { offset: 1, color: colors[1] },  
 ]);  
};  
const getData = () => {  
 countUserNum().then((res) => {  
 console.log("左中--用户总览",res);  
 if (res.success) {  
 state.lockNum = res.data.lockNum;  
 state.offlineNum = res.data.offlineNum;  
 state.onlineNum = res.data.onlineNum;  
 state.totalNum = res.data.totalNum;  
 state.alarmNum = res.data.alarmNum;  
 setOption();  
 }else{  
 ElMessage.error(res.msg)  
 }  
 }).catch(err=>{  
 ElMessage.error(err)  
 });  
};  
getData();  
const setOption = () => {  
 option.value = {  
 title: {  
 top: "center",  
 left: "center",  
 text: [`{value|${state.totalNum}}`, "{name|总数}"].join("\n"),  
 textStyle: {  
 rich: {  
 value: {  
 color: "#ffffff",  
 fontSize: 24,  
 fontWeight: "bold",  
 lineHeight: 20,  
 padding:[4,0,4,0]  
 },  
 name: {  
 color: "#ffffff",  
 lineHeight: 20,  
 },  
 },  
 },  
 },  
 tooltip: {  
 trigger: "item",  
 backgroundColor: "rgba(0,0,0,.6)",  
 borderColor: "rgba(147, 235, 248, .8)",  
 textStyle: {  
 color: "#FFF",  
 },  
 },  
 series: [  
 {  
 name: "用户总览",  
 type: "pie",  
 radius: ["40%", "70%"],  
 itemStyle: {  
 borderRadius: 6,  
 borderColor: "rgba(255,255,255,0)",  
 borderWidth: 2,  
 },  
 color: colors,  
 label: {  
 show: true,  
 formatter: " {b|{b}} \n {c|{c}个} {per|{d}%} ",  
 rich: {  
 b: {  
 color: "#fff",  
 fontSize: 12,  
 lineHeight: 26,  
 },  
 c: {  
 color: "#31ABE3",  
 fontSize: 14,  
 },  
 per: {  
 color: "#31ABE3",  
 fontSize: 14,  
 },  
 },  
 },  
 emphasis: {  
 show: false,  
 },  
 legend: {  
 show: false,  
 },  
 tooltip: { show: true },  
 labelLine: {  
 show: true,  
 length: 20, // 第一段线 长度  
 length2: 36, // 第二段线 长度  
 smooth: 0.2,  
 lineStyle: {},  
 },  
 data: [  
 {  
 value: state.onlineNum,  
 name: "在线",  
 itemStyle: {  
 color: echartsGraphic(["#0BFC7F", "#A3FDE0"]),  
 },  
 },  
 {  
 value: state.offlineNum,  
 name: "离线",  
 itemStyle: {  
 color: echartsGraphic(["#A0A0A0", "#DBDFDD"]),  
 },  
 },  
 {  
 value: state.lockNum,  
 name: "锁定",  
 itemStyle: {  
 color: echartsGraphic(["#F48C02", "#FDDB7D"]),  
 },  
 },  
 {  
 value: state.alarmNum,  
 name: "异常",  
 itemStyle: {  
 color: echartsGraphic(["#F4023C", "#FB6CB7"]),  
 },  
 },  
 ],  
 },  
 ],  
 };  
};  
</script>  
<template>  
 <v-chart class="chart" :option="option" />  
</template>  
<style scoped lang="scss"></style>  
<script setup lang="ts">  
import { reactive, ref, onMounted,onUnmounted } from "vue";  
import { countDeviceNum } from "@/api";  
import CountUp from "@/components/count-up";  
import {ElMessage} from "element-plus"  
const duration = ref(2);  
const state = reactive({  
 alarmNum: 0,  
 offlineNum: 0,  
 onlineNum: 0,  
 totalNum: 0,  
});  
const getData = () => {  
 countDeviceNum().then((res) => {  
 console.log("左上--设备总览",res);  
 if (res.success) {  
 state.alarmNum = res.data.alarmNum;  
 state.offlineNum = res.data.offlineNum;  
 state.onlineNum = res.data.onlineNum;  
 state.totalNum = res.data.totalNum;  
 }else{  
 ElMessage.error(res.msg)  
 }  
 }).catch(err=>{  
 ElMessage.error(err)  
 });  
};  
getData()  
</script>  
<template>  
 <ul class="user\_Overview flex">  
 <li class="user\_Overview-item" style="color: #00fdfa">  
 <div class="user\_Overview\_nums allnum">  
 <CountUp :endVal="state.totalNum" :duration="duration" />  
 </div>  
 <p style="text-indent: 30px;">总设备数</p>  
 </li>  
 <li class="user\_Overview-item" style="color: #07f7a8">  
 <div class="user\_Overview\_nums online">  
 <CountUp :endVal="state.onlineNum" :duration="duration" />  
 </div>  
 <p style="text-indent: 35px;">在线数</p>  
 </li>  
 <li class="user\_Overview-item" style="color: #e3b337">  
 <div class="user\_Overview\_nums offline">  
 <CountUp :endVal="state.offlineNum" :duration="duration" />  
 </div>  
 <p style="text-indent: 35px;">掉线数</p>  
 </li>  
 <li class="user\_Overview-item" style="color: #f5023d">  
 <div class="user\_Overview\_nums laramnum">  
 <CountUp :endVal="state.alarmNum" :duration="duration" />  
 </div>  
 <p style="text-indent: 30px;">告警次数</p>  
 </li>  
 </ul>  
</template>  
<style scoped lang="scss">  
.left-top {  
 width: 100%;  
 height: 100%;  
}  
.user\_Overview {  
 li {  
 flex: 1;  
 p {  
 height: 16px;  
 font-size: 16px;  
 }  
 .user\_Overview\_nums {  
 width: 100px;  
 height: 100px;  
 text-align: center;  
 line-height: 100px;  
 font-size: 22px;  
 margin: 50px auto 30px; text-align: center;  
 background-size: cover;  
 background-position: center center;  
 position: relative;  
 &::before {  
 content: "";  
 position: absolute;  
 width: 100%;  
 height: 100%;  
 top: 0;  
 left: 0;  
 }  
 &.bgdonghua::before {  
 animation: rotating 14s linear infinite;  
 }  
 }  
 .allnum {  
 &::before {  
 background-image: url("@/assets/img/left\_top\_lan.png");  
 }  
 }  
 .online {  
 &::before {  
 background-image: url("@/assets/img/left\_top\_lv.png");  
 }  
 }  
 .offline {  
 &::before {  
 background-image: url("@/assets/img/left\_top\_huang.png");  
 }  
 }  
 .laramnum {  
 &::before {  
 background-image: url("@/assets/img/left\_top\_hong.png");  
 }  
 }  
 }  
}  
</style>  
<script setup lang="ts">  
import { rightBottom } from "@/api";  
import SeamlessScroll from "@/components/seamless-scroll";  
import { computed, onMounted, reactive } from "vue";  
import { useSettingStore } from "@/stores";  
import { storeToRefs } from "pinia";  
import EmptyCom from "@/components/empty-com";  
import { ElMessage } from "element-plus";  
const settingStore = useSettingStore();  
const { defaultOption, indexConfig } = storeToRefs(settingStore);  
const state = reactive<any>({  
 list: [],  
 defaultOption: {  
 ...defaultOption.value,  
 singleHeight: 252,  
 limitScrollNum: 3,  
 },  
 scroll: true,  
});  
const getData = () => {  
 rightBottom({ limitNum: 20 })  
 .then((res) => {  
 console.log("右下", res);  
 if (res.success) {  
 state.list = res.data.list;  
 } else {  
 ElMessage({  
 message: res.msg,  
 type: "warning",  
 });  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const comName = computed(() => {  
 if (indexConfig.value.rightBottomSwiper) {  
 return SeamlessScroll;  
 } else {  
 return EmptyCom;  
 }  
});  
function montionFilter(val: any) {  
 return val ? Number(val).toFixed(2) : "--";  
}  
const handleAddress = (item: any) => {  
 return `${item.provinceName}/${item.cityName}/${item.countyName}`;  
};  
onMounted(() => {  
 getData();  
});  
</script>  
<template>  
 <div class="right\_bottom\_wrap beautify-scroll-def" :class="{ 'overflow-y-auto': !indexConfig.rightBottomSwiper }">  
 <component  
 :is="comName"  
 :list="state.list"  
 v-model="state.scroll"  
 :singleHeight="state.defaultOption.singleHeight"  
 :step="state.defaultOption.step"  
 :limitScrollNum="state.defaultOption.limitScrollNum"  
 :hover="state.defaultOption.hover"  
 :singleWaitTime="state.defaultOption.singleWaitTime"  
 :wheel="state.defaultOption.wheel"  
 >  
 <ul class="right\_bottom">  
 <li class="right\_center\_item" v-for="(item, i) in state.list" :key="i">  
 <span class="orderNum">{{ i + 1 }}</span>  
 <div class="inner\_right">  
 <div class="dibu"></div>  
 <div class="flex">  
 <div class="info">  
 <span class="labels">设备ID：</span>  
 <span class="text-content zhuyao"> {{ item.gatewayno }}</span>  
 </div>  
 <div class="info">  
 <span class="labels">型号：</span>  
 <span class="text-content"> {{ item.terminalno }}</span>  
 </div>  
 <div class="info">  
 <span class="labels">告警值：</span>  
 <span class="text-content warning"> {{ montionFilter(item.alertvalue) }}</span>  
 </div>  
 </div>  
 <div class="flex">  
 <div class="info">  
 <span class="labels shrink-0"> 设备位置：</span>  
 <span class="ciyao truncate" style="font-size: 12px; width: 220px" :title="handleAddress(item)">  
 设备位置</span  
 >  
 </div>  
 <div class="info time shrink-0">  
 <span class="labels">时间：</span>  
 <span class="text-content" style="font-size: 12px"> {{ item.createtime }}</span>  
 </div>  
 </div>  
 <div class="flex">  
 <div class="info">  
 <span class="labels">报警内容：</span>  
 <span class="text-content ciyao" :class="{ warning: item.alertdetail }">  
 报警内容</span  
 >  
 </div>  
 </div>  
 </div>  
 </li>  
 </ul>  
 </component>  
 </div>  
</template>  
<style scoped lang="scss">  
.right\_bottom {  
 width: 100%;  
 height: 100%;  
 .right\_center\_item {  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 height: auto;  
 padding: 10px;  
 font-size: 14px;  
 color: #fff;  
 .orderNum {  
 margin: 0 20px 0 -20px;  
 }  
 .inner\_right {  
 position: relative;  
 height: 100%;  
 width: 400px;  
 flex-shrink: 0;  
 line-height: 1.5;  
 .dibu {  
 position: absolute;  
 height: 2px;  
 width: 104%;  
 background-image: url("@/assets/img/zuo\_xuxian.png");  
 bottom: -12px;  
 left: -2%;  
 background-size: cover;  
 }  
 }  
 .info {  
 margin-right: 10px;  
 display: flex;  
 align-items: center;  
 .labels {  
 flex-shrink: 0;  
 font-size: 12px;  
 color: rgba(255, 255, 255, 0.6);  
 }  
 .zhuyao {  
 color: $primary-color;  
 font-size: 15px;  
 }  
 .ciyao {  
 color: rgba(255, 255, 255, 0.8);  
 }  
 .warning {  
 color: #e6a23c;  
 font-size: 15px;  
 }  
 }  
 }  
}  
.right\_bottom\_wrap {  
 overflow: hidden;  
 width: 100%;  
 height: 252px;  
}  
.overflow-y-auto {  
 overflow-y: auto;  
}  
</style>  
<script setup lang="ts">  
import { ref, reactive } from "vue";  
import CapsuleChart from "@/components/datav/capsule-chart";  
import { ranking } from "@/api";  
import { ElMessage } from "element-plus";  
const msg = ref([  
 {  
 "value": 94.1,  
 "name": "传感器1"  
 },  
 {  
 "value": 92.5,  
 "name": "传感器2"  
 },  
 {  
 "value": 91.8,  
 "name": "传感器3"  
 },  
 {  
 "value": 75.2,  
 "name": "传感器4"  
 },  
 {  
 "value": 45.7,  
 "name": "传感器5"  
 },  
 {  
 "value": 41.7,  
 "name": "传感器6"  
 },  
 {  
 "value": 24.0,  
 "name": "传感器7"  
 },  
 {  
 "value": 20.2,  
 "name": "传感器8"  
 }  
]);  
const config = ref({  
 showValue: true,  
 unit: "%",  
});  
const data = ref([]);  
const getData = () => {  
 ranking()  
 .then((res) => {  
 console.log("右中--风险预测", res);  
 if (res.success) {  
 data.value = msg.value; // 这里使用我们的假数据  
 } else {  
 ElMessage({  
 message: res.msg,  
 type: "warning",  
 });  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const generateNewData = (currentData:any) => {  
 return currentData.map(sensor => {  
 const newValue = parseFloat((sensor.value + (Math.random() - 0.5) \* 2).toFixed(1));  
 const clampedValue = parseFloat(Math.min(100, Math.max(0, newValue)).toFixed(1));  
 return {  
 ...sensor,  
 value: clampedValue  
 };  
 });  
};  
getData();  
const updateInterval = setInterval(() => {  
 data.value = generateNewData(data.value);  
}, 2000);  
import { onUnmounted } from 'vue';  
onUnmounted(() => {  
 clearInterval(updateInterval);  
});  
</script>  
<template>  
 <div class="right\_bottom">  
 <CapsuleChart :config="config" style="width: 100%; height: 260px" :data="data" />  
 </div>  
</template>  
<style scoped lang="scss">  
.right\_bottom {  
 box-sizing: border-box;  
 padding: 0 16px;  
}  
</style>  
<script setup lang="ts">  
import { ref, onMounted, onUnmounted } from "vue";  
import { alarmNum } from "@/api";  
import { graphic } from "echarts/core";  
import { ElMessage } from "element-plus";  
const option = ref({});  
const msg = ref({  
 dateList: ['10:41', '10:42', '10:43', '10:44', '10:45', '10:46'],  
 numList: [436, 589, 689, 789, 657, 590],  
 numList2: [34, 94, 134, 234, 334, 234]  
});  
const updateData = () => {  
 const now = new Date();  
 const newDate = now.toLocaleTimeString('default', { hour: '2-digit', minute: '2-digit' });  
 const lastTime = new Date(`2024-08-03T${msg.value.dateList[msg.value.dateList.length - 1]}`);  
 if (now < lastTime) {  
 setTimeout(updateData, 60000 - (now.getTime() - lastTime.getTime()));  
 return;  
 }  
 msg.value.dateList.push(newDate);  
 msg.value.dateList.shift(); // 移除第一个元素以保持数组长度不变  
 const numList = msg.value.numList.map(num => Math.round(num \* (1 + (Math.random() \* 20 - 10) / 100)));  
 const numList2 = msg.value.numList2.map(num => Math.round(num \* (1 + (Math.random() \* 20 - 10) / 100)));  
 msg.value.numList = numList;  
 msg.value.numList2 = numList2;  
 setOption(msg.value.dateList, msg.value.numList, msg.value.numList2);  
};  
const getData = () => {  
 alarmNum()  
 .then((res) => {  
 console.log("右上--实时监测信息 ", res);  
 if (res.success) {  
 setOption(msg.value.dateList, msg.value.numList, msg.value.numList2);  
 } else {  
 ElMessage({  
 message: res.msg,  
 type: "warning",  
 });  
 }  
 })  
 .catch((err) => {  
 ElMessage.error(err);  
 });  
};  
const setOption = (xData: any[], yData: any[], yData2: any[]) => {  
 option.value = {  
 xAxis: {  
 type: 'category',  
 data: xData,  
 boundaryGap: false, // 不留白，从原点开始  
 name: '时间', // 添加横轴名称  
 splitLine: {  
 show: true,  
 lineStyle: {  
 color: 'rgba(31,99,163,.2)',  
 },  
 },  
 axisLine: {  
 lineStyle: {  
 color: 'rgba(31,99,163,.1)',  
 },  
 },  
 axisLabel: {  
 color: '#7EB7FD',  
 fontWeight: '500',  
 },  
 },  
 yAxis: {  
 type: 'value',  
 name: '功率', // 添加纵轴名称  
 splitLine: {  
 show: true,  
 lineStyle: {  
 color: 'rgba(31,99,163,.2)',  
 },  
 },  
 axisLine: {  
 lineStyle: {  
 color: 'rgba(31,99,163,.1)',  
 },  
 },  
 axisLabel: {  
 color: '#7EB7FD',  
 fontWeight: '500',  
 },  
 },  
 tooltip: {  
 trigger: 'axis',  
 backgroundColor: 'rgba(0,0,0,.6)',  
 borderColor: 'rgba(147, 235, 248, .8)',  
 textStyle: {  
 color: '#FFF',  
 },  
 },  
 grid: {  
 show: true,  
 left: '10px',  
 right: '30px',  
 bottom: '10px',  
 top: '32px',  
 containLabel: true,  
 borderColor: '#1F63A3',  
 },  
 series: [  
 {  
 data: yData,  
 type: 'line',  
 smooth: true,  
 symbol: 'none', //去除点  
 name: '报警1功率',  
 color: 'rgba(252,144,16,.7)',  
 areaStyle: {  
 color: new graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [  
 {  
 offset: 0,  
 color: 'rgba(252,144,16,.7)',  
 },  
 {  
 offset: 1,  
 color: 'rgba(252,144,16,.0)',  
 },  
 ],  
 false  
 ),  
 },  
 markPoint: {  
 data: [  
 {  
 name: '最大值',  
 type: 'max',  
 valueDim: 'y',  
 symbol: 'rect',  
 symbolSize: [60, 26],  
 symbolOffset: [0, -20],  
 itemStyle: {  
 color: 'rgba(0,0,0,0)',  
 },  
 label: {  
 color: '#FC9010',  
 backgroundColor: 'rgba(252,144,16,0.1)',  
 borderRadius: 6,  
 padding: [7, 14],  
 borderWidth: 0.5,  
 borderColor: 'rgba(252,144,16,.5)',  
 formatter: '报警1：{c}',  
 },  
 },  
 {  
 name: '最大值',  
 type: 'max',  
 valueDim: 'y',  
 symbol: 'circle',  
 symbolSize: 6,  
 itemStyle: {  
 color: '#FC9010',  
 shadowColor: '#FC9010',  
 shadowBlur: 8,  
 },  
 label: {  
 formatter: '',  
 },  
 },  
 ],  
 },  
 },  
 {  
 data: yData2,  
 type: 'line',  
 smooth: true,  
 symbol: 'none', //去除点  
 name: '报警2功率',  
 color: 'rgba(9,202,243,.7)',  
 areaStyle: {  
 color: new graphic.LinearGradient(  
 0,  
 0,  
 0,  
 1,  
 [  
 {  
 offset: 0,  
 color: 'rgba(9,202,243,.7)',  
 },  
 {  
 offset: 1,  
 color: 'rgba(9,202,243,.0)',  
 },  
 ],  
 false  
 ),  
 },  
 markPoint: {  
 data: [  
 {  
 name: '最大值',  
 type: 'max',  
 valueDim: 'y',  
 symbol: 'rect',  
 symbolSize: [60, 26],  
 symbolOffset: [0, -20],  
 itemStyle: {  
 color: 'rgba(0,0,0,0)',  
 },  
 label: {  
 color: '#09CAF3',  
 backgroundColor: 'rgba(9,202,243,0.1)',  
 borderRadius: 6,  
 borderColor: 'rgba(9,202,243,.5)',  
 padding: [7, 14],  
 formatter: '报警2：{c}',  
 borderWidth: 0.5,  
 },  
 },  
 {  
 name: '最大值',  
 type: 'max',  
 valueDim: 'y',  
 symbol: 'circle',  
 symbolSize: 6,  
 itemStyle: {  
 color: '#09CAF3',  
 shadowColor: '#09CAF3',  
 shadowBlur: 8,  
 },  
 label: {  
 formatter: '',  
 },  
 },  
 ],  
 },  
 },  
 ],  
 };  
};  
onMounted(() => {  
 setOption(msg.value.dateList, msg.value.numList, msg.value.numList2);  
 const timer = setInterval(updateData, 2000); // 每隔10秒更新数据  
 onUnmounted(() => clearInterval(timer));  
 getData();  
});  
</script>  
<template>  
 <v-chart class="chart" :option="option" v-if="JSON.stringify(option) != '{}'" />  
</template>  
<style scoped lang="scss"></style>  
<script setup lang="ts">  
import { ref, reactive, nextTick, watch } from "vue";  
import BorderBox13 from "@/components/datav/border-box-13";  
import \* as THREE from 'three';  
import { onMounted } from "vue";  
import { useSettingStore } from "@/stores";  
import { GLTFLoader } from "three/addons/loaders/GLTFLoader.js"; // gltf模型引入  
import { OrbitControls } from "three/addons/controls/OrbitControls.js"; // 坐标轴控制器  
import { CSS2DRenderer,CSS2DObject } from 'three/addons/renderers/CSS2DRenderer.js'; // CSS2标签  
import { CSS3DRenderer,CSS3DObject } from 'three/addons/renderers/CSS3DRenderer.js';  
withDefaults(  
 defineProps<{  
 title: number | string;  
 }>(),  
 {  
 title: "地图",  
 }  
);  
const settingStore = useSettingStore();  
const scene = new THREE.Scene(); // 创建一个场景  
const camera = new THREE.PerspectiveCamera(  
 30,  
 window.innerWidth / window.innerHeight,  
 0.1,  
 1000  
);  
let speed = ref(0.005) // 存放旋转速度  
let screenDom = ref(null) // 获取dom  
watch(() => settingStore.indexConfig.modelRotateSpeed, (newVal) => {  
 speed.value = newVal ? 0.005 : 0;  
});  
function initThree() {  
 scene.background = null; // 设置场景背景为透明  
 const threeDemo = document.getElementById("dom");  
 const renderer = new THREE.WebGLRenderer({  
 canvas: threeDemo,  
 antialias: true,  
 alpha: true, // 设置渲染器的 alpha 通道为 true  
 logarithmicDepthBuffer: false, //如果要在单个场景中处理巨大的比例差异，就有必要使用。此处要用false，否则文字标签不显示  
 });  
 camera["position"].z = 30;  
 camera["position"].y = -10;  
 const div = document.getElementById('tag');  
 const tag = new CSS2DObject(div);  
 tag.position.set(3,3,3);  
 const group = new THREE.Group();  
 group.add(tag);  
 function animate() {  
 renderer.render(scene, camera);  
 requestAnimationFrame(animate);  
 }  
 const controls = new OrbitControls(camera, renderer.domElement);  
 controls.update()  
 animate();  
 createAreaPoint();  
 renderScene();  
 function createAreaPoint() {  
 for (let i = 0, length = areas.length; i < length; i++) {  
 let name = areas[i].name  
 let position = createPosition(areas[i].position)  
 createTxt(position,name); // 精灵标签函数  
 }  
 }  
 function createPosition(lnglat:any) {  
 let spherical = new THREE.Spherical  
 spherical.radius = 10;  
 let lng = lnglat[0]  
 let lat = lnglat[1]  
 let theta = (lng + 90) \* (Math.PI / 180)  
 let phi = (90 - lat) \* (Math.PI / 180)  
 spherical.phi = phi; // phi是方位面（水平面）内的角度，范围0~360度  
 spherical.theta = theta; // theta是俯仰面（竖直面）内的角度，范围0~180度  
 let position = new THREE.Vector3()  
 position.setFromSpherical(spherical)  
 return position  
 }  
 function createTxt(position,name){  
 let texture = new THREE.TextureLoader().load('');  
 let material = new THREE.SpriteMaterial({ map: texture });  
 let spriteMesh = new THREE.Sprite(material);  
 spriteMesh.name = name;  
 spriteMesh.scale.x = 1;  
 spriteMesh.scale.y = 1;  
 spriteMesh.position.x = (position.x >0 ? position.x+1 : position.x-1);  
 spriteMesh.position.y = (position.y >0 ? position.y+1 : position.y-1);  
 spriteMesh.position.z = (position.z >0 ? position.z+1 : position.z-1);  
 spriteMesh.lookAt(new THREE.Vector3(0, 0,0));  
 scene.add(spriteMesh);  
 }  
 function renderScene() {  
 requestAnimationFrame(renderScene); //循环调用renderScene函数  
 renderer.render(scene, camera);  
 if(resizeDevicePixel(renderer)) {  
 const canvas = renderer.domElement;  
 camera.aspect = canvas.clientWidth / canvas.clientHeight;  
 camera.updateProjectionMatrix();  
 }  
 }  
}  
const gltfLoader = new GLTFLoader();  
gltfLoader.load("/air\_engine/scene.gltf", (gltf:any) => {  
 let model = gltf.scene;  
 model.position.set(0,0,0)  
 model.scale.set(3.5, 3.5, 3.5);  
 scene.add(model);  
 const ambientLight = new THREE.AmbientLight(0xffffff, 0.5); // 软光  
 const directionalLight = new THREE.DirectionalLight(0xffffff, 0.5);  
 directionalLight["position"].set(1, 1, 1).normalize();  
 scene.add(ambientLight);  
 scene.add(directionalLight);  
 const pointLight = new THREE.PointLight(0xffffff, 1, 100);  
 pointLight["position"].set(0, 50, 50);  
 scene.add(pointLight);  
 const light01 = new THREE.DirectionalLight(0xffffff, 1)  
 light01["position"].set(0, 0, 10)  
 const light02 = new THREE.DirectionalLight(0xffffff, 1)  
 light02["position"].set(0, 0, -10)  
 const light03 = new THREE.DirectionalLight(0xffffff, 1)  
 light03["position"].set(10, 0, 0)  
 const light04 = new THREE.DirectionalLight(0xffffff, 1)  
 light04["position"].set(-10, 0, 0)  
 const light05 = new THREE.DirectionalLight(0xffffff, 1)  
 light05["position"].set(0, 10, 0)  
 const light06 = new THREE.DirectionalLight(0xffffff, 1)  
 light06["position"].set(5, 10, 0)  
 const light07 = new THREE.DirectionalLight(0xffffff, 1)  
 light07["position"].set(0, -10, 0)  
 scene.add(light01)  
 scene.add(light02)  
 scene.add(light03)  
 scene.add(light04)  
 scene.add(light05)  
 scene.add(light06)  
 scene.add(light07)  
})  
scene.background = new THREE.Color(0xff0000);  
setInterval(() => {  
 scene["rotation"].y += speed.value  
}, 1000 / 60)  
onMounted(() => {  
 initThree();  
});  
function clickMouse(event) {  
 console.log("ddd")  
 event.preventDefault();  
 let raycaster = new THREE.Raycaster();  
 let mouse = new THREE.Vector2();  
 let getBoundingClientRect = dom.getBoundingClientRect();  
 mouse.x = ((event.clientX - getBoundingClientRect.left) / dom.offsetWidth) \* 2 - 1;  
 mouse.y = -((event.clientY - getBoundingClientRect.top) / dom.offsetHeight) \* 2 + 1;  
 raycaster.setFromCamera(mouse, camera);  
 let intersects = raycaster.intersectObjects(scene.children);  
 deleteDiv(); //调用清除弹窗函数  
 for (let i = 0; i < intersects.length; i++) {  
 if (intersects[i].object.type == 'Sprite') {  
 let countryName = intersects[i].object.name; // 国家  
 console.log("ming cheng",countryName)  
 let V3 = intersects[i].object.position ; // 三维坐标  
 if(countryName !=''){  
 divPop(countryName, V3); //调用弹窗函数  
 }  
 }  
 }  
 function divPop(countryName, V3){  
 let rs = WorldToScreen(V3.x, V3.y, V3.z); // 调用世界坐标转屏幕坐标函数  
 let div = document.createElement("divCell"); //创建一个div  
 div.id = "divCell"; //设置ID  
 div.innerHTML = countryName; //div的内容  
 div.style.padding = '5px';  
 div.style.position = 'absolute';  
 div.style.backgroundColor = 'rgba(255, 255, 255, 0.8)';  
 div.style.left = rs.x + "px";  
 div.style.top = rs.y + "px";  
 document.body.appendChild(div); //添加到页面  
 }  
 function deleteDiv(){  
 let d = document.getElementById("divCell");  
 if (d != null){  
 d.parentNode.removeChild(d);  
 }  
 }  
 function WorldToScreen(x, y, z) {  
 let worldVector = new THREE.Vector3(x, y, z);  
 let vector = worldVector.project(camera); //世界坐标转标准设备坐标  
 let w = window.innerWidth / 2;  
 let h = window.innerHeight / 2;  
 return {  
 x: Math.round(vector.x \* w + w),  
 y: Math.round(-vector.y \* h + h)  
 }  
 }  
}  
window.addEventListener('click', clickMouse, false);  
function resizeDevicePixel(renderer) {  
 const canvas = renderer.domElement  
 let width = window.innerWidth  
 let height = window.innerHeight  
 let devicePixelWidth = canvas.width / window.devicePixelRatio  
 let devicePixelHeight = canvas.height / window.devicePixelRatio  
 const needResize = devicePixelWidth !== width || devicePixelHeight !== height  
 if (needResize) {  
 renderer.setSize(width, height, false)  
 }  
 return needResize  
}  
</script>  
<template>  
 <div class="centermap">  
 <div class="maptitle">  
 <div class="zuo"></div>  
 <span class="titletext">{{ title }}</span>  
 <div class="you"></div>  
 </div>  
 <div class="mapwrap">  
 <BorderBox13>  
 <canvas id="dom" ref="screenDom">  
 <div id="tag">标签内容</div>  
 </canvas>  
 </BorderBox13>  
 </div>  
 </div>  
</template>  
<style scoped lang="scss">  
.centermap {  
 margin-bottom: 30px;  
 .maptitle {  
 height: 60px;  
 display: flex;  
 justify-content: center;  
 padding-top: 10px;  
 box-sizing: border-box;  
 .titletext {  
 font-size: 28px;  
 font-weight: 900;  
 letter-spacing: 6px;  
 background: linear-gradient(92deg, #0072ff 0%, #00eaff 48.8525390625%, #01aaff 100%);  
 -webkit-background-clip: text;  
 -webkit-text-fill-color: transparent;  
 margin: 0 10px;  
 }  
 .zuo,  
 .you {  
 background-size: 100% 100%;  
 width: 29px;  
 height: 20px;  
 margin-top: 8px;  
 }  
 .zuo {  
 background: url("@/assets/img/xiezuo.png") no-repeat;  
 }  
 .you {  
 background: url("@/assets/img/xieyou.png") no-repeat;  
 }  
 }  
 .mapwrap {  
 height: 580px;  
 width: 100%;  
 box-sizing: border-box;  
 position: relative;  
 .quanguo {  
 position: absolute;  
 right: 20px;  
 top: -46px;  
 width: 80px;  
 height: 28px;  
 border: 1px solid #00eded;  
 border-radius: 10px;  
 color: #00f7f6;  
 text-align: center;  
 line-height: 26px;  
 letter-spacing: 6px;  
 cursor: pointer;  
 box-shadow: 0 2px 4px rgba(0, 237, 237, 0.5), 0 0 6px rgba(0, 237, 237, 0.4);  
 z-index: 10;  
 }  
 }  
}  
#dom {  
 width: 700px;  
 height: 580px;  
 position: absolute;  
 top: 0;  
 left: 0;  
}  
</style>  
<script setup lang="ts">  
import \* as THREE from 'three';  
import { onMounted } from "vue";  
import { useSettingStore } from "@/stores";  
import { GLTFLoader } from "three/addons/loaders/GLTFLoader.js"; // gltf模型引入  
import { OrbitControls } from "three/addons/controls/OrbitControls.js"; // 坐标轴控制器  
import { CSS2DRenderer,CSS2DObject } from 'three/addons/renderers/CSS2DRenderer.js'; // CSS2标签  
import { CSS3DRenderer,CSS3DObject } from 'three/addons/renderers/CSS3DRenderer.js';  
import { ref, reactive, nextTick, watch } from "vue";