可视化拖拽编辑器(四)

```
registerConfig.register({
    label: '输入框',
    resize:{
        width:true // 更改更改输入框横向大小
    },
});
```

```
registerConfig.register({
    label: '按钮',
    resize:{
        width:true,
        height:true // 按钮可以更改宽高
    },
})
```

```
const { width, height } = component.resize || {}
return <div class="editor-block" style={blockStyles.value} ref=
{blockRef}>
    {RenderComponent}
    {/* block修改宽高信息 ， component需要知道改的是宽度还是高度*/}
    {props.block.focus && (width || height) &&
        <BlockResize
            block={props.block}
            component={component}>
        </BlockResize>
    }
</div>
```

```
import { defineComponent } from "vue";

export default defineComponent({
    props: {
        block: { type: Object },
        component: { type: Object }
    },
    setup(props) {
        return () => {
            const { width, height } = props.component.resize || {};
            let data = {};
            const onmousedown = (e, direction) => {
                e.stopPropagation();
```

```
                data = {
                    startX: e.clientX,
                    startY: e.clientY,
                    startWidth: props.block.width,
                    startHeight: props.block.height,
                    startLeft: props.block.left,
                    startTop: props.block.top,
                    direction
                }
                document.body.addEventListener('mousemove', mousemove)
                document.body.addEventListener('mouseup', mouseup)
            }

            const mouseup = () => {
                document.body.removeEventListener('mousemove',
mousemove)
                document.body.removeEventListener('mouseup', mouseup)
            }

            const mousemove = (e) => {
                e.stopPropagation()
                const { startX, startY, startWidth, startHeight,
direction, startLeft, startTop } = data;
                let { clientX, clientY } = e;
                if (direction.horizontal == 'center') { // 只能改纵向位置
固定横向
                    clientX = startX
                }
                if (direction.vertical == 'center') { // 只能改横向  固定纵
向
                    clientY = startY
                }
                let durX = clientX - startX;
                let durY = clientY - startY;
                let block = props.block;
                if (direction.vertical === 'start') {
                    durY = -durY
                    block.top = startTop - durY
                }
                if (direction.horizontal === 'start') {
                    durX = -durX
                    block.left = startLeft - durX
                }

                const width = startWidth + durX;
                const height = startHeight + durY
                block.width = width;
                block.height = height;
                block.hasResize = true;
```

```
            }

            return <>
                {height && <>
                    <div class="block-resize block-resize-top"
                        onMousedown={e => onmousedown(e, { horizontal:
'center', vertical: 'start' })}>
                    </div>
                    <div class="block-resize block-resize-bottom"
                        onMousedown={e => onmousedown(e, { horizontal:
'center', vertical: 'end' })}>
                    </div>
                </>}

                {width && <>
                    <div class="block-resize block-resize-left"
                        onMousedown={e => onmousedown(e, { horizontal:
'start', vertical: 'center' })}>
                    </div>
                    <div class="block-resize block-resize-right"
                        onMousedown={e => onmousedown(e, { horizontal:
'end', vertical: 'center' })}>
                    </div>
                </>}

                {(height && width) && <>
                    <div class="block-resize block-resize-top-left"
                        onMousedown={e => onmousedown(e, { horizontal:
'start', vertical: 'start' })}>
                    </div>
                    <div class="block-resize block-resize-top-right"
                        onMousedown={e => onmousedown(e, { horizontal:
'end', vertical: 'start' })}>
                    </div>
                    <div class="block-resize block-resize-bottom-left"
                        onMousedown={e => onmousedown(e, { horizontal:
'start', vertical: 'end' })}>
                    </div>
                    <div class="block-resize block-resize-bottom-right"
                        onMousedown={e => onmousedown(e, { horizontal:
'end', vertical: 'end' })}
                    ></div>
                </>}
            </>
        }
    }
})
```

```
.block-resize {
```

```
    position: absolute;
    width: 8px;
    height: 8px;
    background: rgb(9, 70, 184);
    z-index: 1000;
}
.block-resize-top {
    top: -2px;
    left: calc(50% - 2px);
}
.block-resize-bottom {
    bottom: -2px;
    left: calc(50% - 2px);
}
.block-resize-left {
    top: calc(50% - 2px);
    left: -2px;
}
.block-resize-right {
    top: calc(50% - 2px);
    right: -2px;
}
.block-resize-top-right {
    right: -2px;
    top: -2px;
}
.block-resize-top-left {
    left: -2px;
    top: -2px;
}
.block-resize-bottom-left {
    bottom: -2px;
    left: -2px;
}
.block-resize-bottom-right {
    bottom: -2px;
    right: -2px;
}
.el-button,.el-input{
    transition: none;
}
```

```
const RenderComponent = component.render({
+    size:props.block.hasResize ?
{width:props.block.width,height:props.block.height}:{},
     props: props.block.props,
     model: Object.keys(component.model || {}).reduce((prev, propName)
=> {
```

```
            const modelName = props.block.model[propName]; // 获取对应需要绑定
的字段
            prev[propName] = {
                modelValue: props.formData[modelName], // 进行数据绑定
                'onUpdate:modelValue': (val) => {
                    props.formData[modelName] = val;
                }
            };
            return prev; // 绑定的数据
        }, {})
});
```

```
registerConfig.register({
    label: '按钮',
    resize:{
        width:true,
        height:true
    },
    preview: () => <ElButton>预览按钮</ElButton>,
+   render: ({ props,size }) => <ElButton type={props.type} style={{
height: size.height + 'px', width: size.width + 'px' }}>{props.text ||
'默认按钮'}</ElButton>,
    key: 'button',
    props: {
        text: createInputProp('按钮内容'),
        type: createSelectProp('按钮类型', [
            { label: '基础', value: 'primary' },
            { label: '成功', value: 'success' },
            { label: '警告', value: 'warning' },
            { label: '危险', value: 'danger' },
            { label: '提示', value: 'info' },
            { label: '文本', value: 'text' }
        ]),
        size: createSelectProp('按钮大小', [
            { label: '默认', value: '' },
            { label: '中等', value: 'medium' },
            { label: '小', value: 'small' },
            { label: '极小', value: 'mini' },
        ])
    }
})
registerConfig.register({
    label: '输入框',
    resize:{
        width:true
    },
    preview: () => <ElInput placeholder="预览输入框"></ElInput>,
+   render: ({ model,size }) => <ElInput placeholder="渲染输入框"
{...model.default} style={{ width: `${size.width}px` }}></ElInput>,
```

```
    key: 'input',
    model: {
        default: '绑定字段'
        // default 等会绑定的model = {modelValue,onUpdate:modelValue}
    }
});
```