在 js 中使用 ref 的值必须使用 .value 获取，在 Vue 的模板中使用 ref 的值不需要通过 value 获取

在使用ref和reactive时记得要先在script标签下引用

基本数据类型只能用ref()包裹，对象类型推荐用reactive()包裹

reactive使普通对象通过Proxy变为代理对象，代理劫持实现响应

reactive包装的对象不要直接去解构赋值，会让数据失去响应能力。因为其是代理有对象

用toRefs()把 reactive()对象解构获得零散的值。

toRefs之后得到的ref引用在获取数据值的时候需要加.value。改变结果数据也会改变未拆分的reactive()数据

<template>

<div class="home">

名字:{{ name }} 价格:{{ price }}

购买数量:{{count}}

<button @click="add">点我加1</button>

</div>

</template>

<script>

import { ref, reactive,toRefs } from "vue";

export default {

name: "Home",

components: {

},

setup() {

//响应式对象

const state = reactive({

count: 0,

name:"商品1",

price:100

});

//修改响应式对象

function add() {

//下面两种写法等效，toRefs让其在script中拥有两种写法

state.count += 1;

// stateRefs.count.value+=1; //toRefs(state).count.value+=1;

}

const stateRefs = toRefs(state)

return { ...stateRefs, add };

},

}

</script>

toRef 是需要一个个手动赋值，而 toRefs 是自动赋值。其余的定义和toRefs几乎一致

<template>

<div class="home">

名字:{{ name }} 价格:{{ price }}

购买数量:{{count}}

<button @click="add">点我加1</button>

</div>

</template>

<script>

import { ref, reactive,toRef } from "vue";

export default {

name: "Home",

components: {

},

setup() {

//响应式对象

const state = reactive({

count: 0,

name:"商品1",

price:100

});

//修改响应式对象

function add() {

//下面两种写法等效,toRef同样让其在script中拥有两种写法

//state.count += 1;

count.value += 1;

}

const count = toRef(state, "count");

return { count, add };

},

}

</script>