Return 数据

 return { message, ages, add, ...toRefs(user), changeUser, student, changeStudent ,user2}

ref（对象）

JS上

   let student = ref({

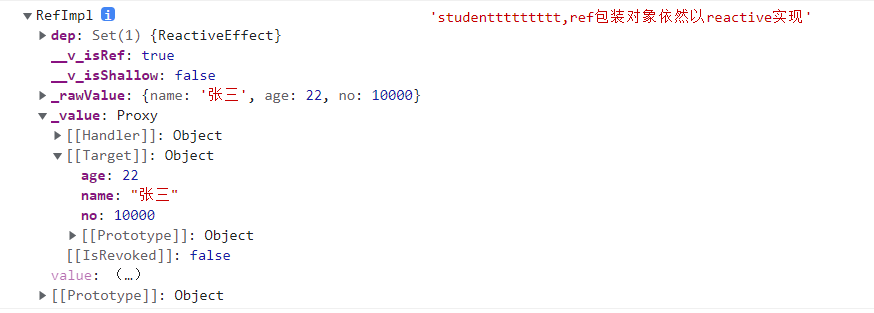
      name: "张三",

      age: 22,

      no: 10000

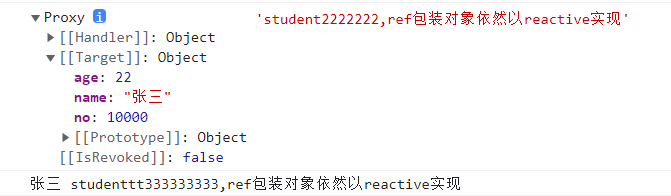
    })

    console.log(student,"studenttttttttt,ref包装对象依然以reactive实现") //当ref包装对象的时候，内容仍然通过reactive的方式实现响应式



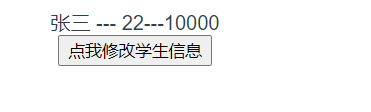
    console.log(student.value,"student2222222,ref包装对象依然以reactive实现")

    console.log(student.value.name,"studenttt333333333,ref包装对象依然以reactive实现")



页面上





reactive

JS上

  const user = reactive({

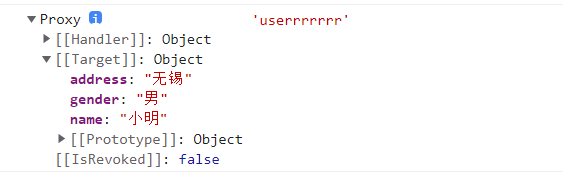
      name: "小明",

      gender: "男",

      address: "无锡"

    })

    console.log(user,"userrrrrrr")



     console.log(user.name,"userrrrrrr2222222")



页面上

使用toRef拆分reactive对象

在js上

   console.log(toRefs(user), "toRefs解构转化");

    console.log(toRefs(user).name, "toRefs解构转化name");

    console.log(toRefs(user).name.\_key, "toRefs解构转化name的key");

    console.log(toRefs(user).name.\_object, "toRefs解构转化name的object");

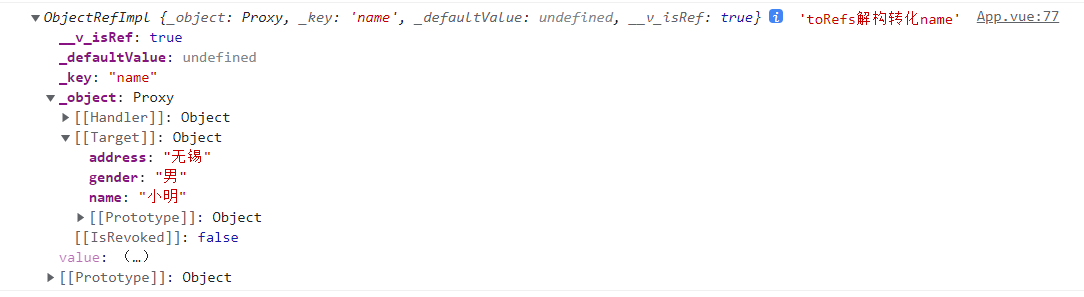
    console.log(

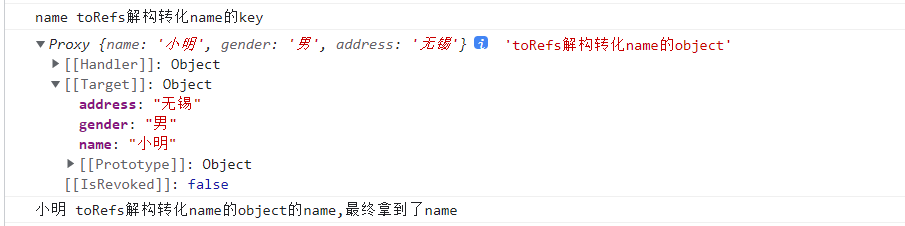
      toRefs(user).name.\_object.name,

      "toRefs解构转化name的object的name,最终拿到了name"

    );







页面表现

