基于TS的VUE3

- [6. 其它 | Vue3+TS 快速上手 (gitee.io)]

区别：

- vue2:必须有根标签
- vue3：可以没有根标签 template
- difineComponent 函数 目的定义一个组件，内部可以传入一个配置对象 返回一个option对象
- vue-cli@4.5.0 以上
- mount 挂载 unmount 卸载

app.js与2.0区别

- app 比vm 更轻
- mount 挂载 unmount 卸载 页面
-

```
improt {createApp} from 'vue' //vue3.0
improt vue  from './App.vue'// vue2.0
const app  = createApp(App)//创建实例
app.mount('app') // 挂载
//vue 2.0
const vm =new Vue({
    render:h=>h(app)
}).$mount('app')
```

组合API

setup

- 是所有组合API的入口函数 只执行一次 组件中用到的：数据、方法都要配置在setup中
- 返回一个对象 则对象的属性和方法，在模板中均可使用
- 在beforeCreate之前执行一次，this是undefined

setup 两个参数

- props：值为对象，包含：组件外部传递过来，且组件内部声明接收了的属性。
  - 需要配置项接受后显示：props:['xxx','xxx']
- context：上下文对象
  - attrs: 值为对象，包含：组件外部传递过来，但没有在props配置中声明的属性, 相当于this.$attrs
  - slots: 收到的插槽内容, 相当于 this.$slots 。使用v-slots:xxx 插槽名字
  - emit: 分发自定义事件的函数, == this.$emit 子：配置项使用 emits:['事件名']

- ```
  setup(){
      let name ='张三'
      let age =19
      function sayhello(){
          alreat(name)
      }
      //返回得数据  页面可以直接使用
      return{
           name,
          age,
          sayhello
      }
  }
  ```

ref函数

- 定义一个响应式数据，返回一个ref对象, 适用：基本类型 对象类型(一般不用ref修饰对象)
- 语法: const xxx = ref(内容)
- 操作数据: xxx.value=
- html 中不需要 .value

- ```
  <p>{{name}}</p>
  improt {ref} from 'vue'
  setup(){
      let name =ref('张三')
      let age =ref(15)
      let job =ref({
          type:"前端工程师",
          salary:'30k'
      })
      function sayhello(){
          name.value ='李四',
          age.value= 13
          job.value.type='UI设计师'
      }
      //返回得数据  页面可以直接使用
      return{
           name,
          age,
          sayhello
      }
  }
  ```

reactive函数

- 作用: 定义一个对象类型的响应式数据（基本类型不要用它，要用 ref 函数）
- 语法： const 代理对象= reactive(源对象) 接收一个对象（或数组），返回一个代理对象（Proxy的实例对象，简称proxy对象）
- reactive定义的响应式数据是“深层次的”。
- 内部基于 ES6 的 Proxy 实现，通过代理对象操作源对象内部数据进行操作
-
```
setup(){
    let job =reactive({
        type:"前端工程师",
        salary:'30k'
    })
    function sayhello(){
        job.type='UI设计师'
    }
    //返回得数据  页面可以直接使用
    return{
        sayhello
    }
}
```

reactive对比ref

- 从定义数据角度对比：
  - ref用来定义：基本类型数据=。
  - reactive用来定义：对象（或数组）类型数据
  - 备注：ref也可以用来定义对象（或数组）类型数据, 它内部会自动通过 reactive 转为代理对象
- 从原理角度对比：
  - ref通过 Object.defineProperty() 的 get 与 set 来实现响应式（数据劫持）。
  - reactive通过使用Proxy来实现响应式（数据劫持）, 并通过Reflect操作源对象内部的数据。
- 从使用角度对比：
  - ref定义的数据：操作数据需要 .value ， 读取数据时模板中直接读取不需要.value 。
  - reactive定义的数据：操作数据与读取数据：均不需要 .value

计算属性

- 使用 引用computed computed(()=>{return xxx})

监视属性

- 监听ref定义得属性
  - watch(xxx,(old,new)=>{},{immediate:true,deep:true}) deep:true 在vue3无效
- 监视多个 可以调用N次watch
  - watch([xxx,xxx],(old,new)=>{})

- 监听 对象 此处无法正确获取 old的值
  - 此处无法正确获取 old的值
  - deep:true 在vue3无效
- 监听 对象中的某个属性
  - xxx 变为 ()=>对象名.属性
  - 多个 [()=>对象名.属性, ()=>对象名.属性]
- 监视的特殊情况
  - reactive定义的数据时 deep无效
  - reactive定义的数据的某个对象属性时 deep有效
- 监听ref声明的对象时 需要 .value或 deep:true

watchEffect函数

- 监视回调中用到那个属性监视那个
-
```
watchEffect(()=>{
    const x1=sum.value
    const x2=person.age
})
```

与computed区别

- computed注重返回值
- watchEffect 注重过程

生命周期

- 两个更名： beforeUnmount挂载 unmounted卸载
- beforeCreate 和 create ==> setup()
- beforeMount/mounted==>onBeforeMount/onMounted
- beforeUpdate/update==>onBeforeUpdate/onUpdate
- beforeUnMount/unmounted==>onBeforeUnMount/onUnmounted

自定义hook函数

- 类似与vue2中混入mixin
- 将共用的方法封装起了 也可以封装Vue中的组合API

toRef

- 把别的数据 变成ref 响应时数据 toRef(对象,'name'键)

toRefs

- ...toRef(对象)

其他Composition API 不常用