文件 package.json

"vue-class-component": "^7.0.2",

"vue-property-decorator": "^8.1.0",

**区别：**

vue class component 是vue 官方出的  
vue property decorator 是社区出的  
其中vue class component 提供了 vue component 等等  
vue property decorator 深度依赖了 vue class component 拓展出了很多操作符 @Prop @Emit @Inject 等等 可以说是 vue class component 的一个超集  
正常开发的时候 你只需要使用 vue property decorator 中提供的操作符即可 不用再从vue class componen 引入vue component



**typescript中private、public、protected修饰符**

public : 公有 在当前类里面、 子类 、类外面都可以访问

protected：保护类型 在当前类里面、子类里面可以访问 ，在类外部没法访问

private ：私有 在当前类里面可以访问，子类、类外部都没法访问

属性如果不加修饰符 默认就是 公有 （public）

1、默认为public

2、当成员被标记为private时，它就不能在声明它的类的外部访问

3、protected和private类似，但是，protected成员在派生类中可以访问

4、构造函数也可以被标记为protected。这意味着这个类不能再包含它的类外被实例化，但是能被继承，也就是可以在派生类中被super执行

子组件传值给父组件

@Emit('text')

btnClick() {

return this.text

}

// 上下都行这样也行

private btnClick() {

this.$emit('text', this.text)

}



// !是和?相对的，是typescript的语法，表示强制解析（也就是告诉typescript编译器，我这里一定有值）。

// 你写?的时候再调用，typescript会提示可能为undefined

@Prop() private msg!: string

@Prop() private person!: string

什么是接口？  
在面向对象语言中，接口（Interfaces）是一个很重要的概念，它是对行为的抽象，而具体如何行动需要由类（classes）去实现（implements）。

TypeScript 中的接口是一个非常灵活的概念，除了可用于[对类的一部分行为进行抽象](https://ts.xcatliu.com/advanced/class-and-interfaces.html#%E7%B1%BB%E5%AE%9E%E7%8E%B0%E6%8E%A5%E5%8F%A3)以外，也常用于对「对象的形状（Shape）」进行描述。  
一个简单的例子：

//定义接口

interface Person{

name: string,

age: number, // 必选属性

job?: string, //可选属性，表示不是必须的参数，

readonly salary: id, //表示是只读的属性,但是在初始化之后不能重新赋值，否则会报错

[ propName : string ] : any, // 任意类型

}

//定义一个变量，它的类型时接口Person，这样就约束了接口的内容

let person: Person = {

name: 'jack',

age: 28,

job: 'IT dog',

id: 9872,

salary: 9999,

}

function printMan(person) {

console.log( `我是 ${person.name}, 我今年 ${person.age},我的工作是${person.job}`);

}

上面例子中的就是一个接口的实例，利用接口约束了传入变量的内容，注意，在赋值时：变量的形状必须和接口的形状保持一致。

接口类型有以下几种属性可选

**1. 必选属性 => "：" 带冒号的属性是必须存在的，不可以多也不能少**

**2. 可选属性 => " ? " 表示有选择的选项，可有可无**

**3. 只读属性 => " readonly " 对象的字段只在创建的时候赋值，注意哦，注意，只读的约束存在于第一次给对象赋值的时候，而不是第一次给只读属性赋值的时候：**

**4. 任意属性 [ propName : 类型 ] : any 表示定义了任意属性取string 类型的值**

**需要注意的是，一旦定义了任意类型，那么确定属性和可选属性都必须是它的子属性**

当使用element-ui的方法时 出现编译错误

原因：写法出现了错误 例子：

that.$refs.upload2.clearFiles()//错误的 <br> that.refs['upload2']'clearFiles'//正确的

// declare 表示声明作用。

declare global {

// interface 接口 「对象的形状（Shape）」进行描述。

interface Window {

attachEvent(type: string, callback: any): void;

detachEvent(type: string, callback: any): void;

}

}

interface EchartsInitOption {

devicePixelRatio?: number;

renderer?: string;

width?: number | string;

height?: number | string;

}

有jsx语言时

lang=‘tsx’

<script lang="tsx">

**typescript不识别jQuery的$**

<script src='./lib/vendor/jquery/jquery.min.js'></script>

index.html已引入，在shims-tsx.d.ts 文件内声明全局

declare global {

var $: any

}

VS code里这样写会报错

me.$refs.form.getContainerHeight()

改为

let ref = me.$refs.form as HTMLFormElement

// 或者 let ref: any = me.$refs.form

ref. getContainerHeight()