课程教案

一、vue3.0环境准备与项目的创建

1、脚手架的安装
vue -V
npm uninstall vue-cli -g
npm install @vue-cli -g

2、创建项目
vue create vue3

3、如何在原有项目上进行vue3.0升级?
vue add vue-next

二、初识composition API

1、通过演示案例向大家展示composition api是如何与之前的代码共存

2、初认composition api的代码基本结构

3、知识点:
1）vue2.x的语法与vue3.0的语法可以共存

2）setup方法：当一个组件被初始化的时候执行一次

3）通过定义数据，并使用reactive方法使得数据支持动态响应

4）通过引入watch方法进行数据监听

5）通过定义函数使得props属性可以动态响应

需要将数据return出去

二、setup函数的使用

1、是一个新的组件选项，组件内使用 Composition API 的入口点

2、调用时机：在beforeCreate之前会被调用

3、this在setup函数中不可用

4、函数接受两个参数，第一个参数是props，第二个参数是context

5、可以对context参数进行解构

6、setup最终导出一个对象，在对象里也可以将函数导出

三、reactive 与 ref的使用与区别

1、ref的使用：
接受一个参数值并返回一个响应式且可改变的 ref 对象。ref 对象拥有一个指向内部值的单一属性 .value

1)在模版中访问
当 ref 作为渲染上下文的属性返回（即在setup返回的对象中）并在模板中使用时，它会自动解套，无需在模板内额外书写.value

2）作为响应式对象的属性访问
当 ref 作为 reactive 对象的 property 被访问或修改时，也将自动解套 value 值，其行为类似普通属性

注意:当嵌套在 reactive Object 中时，ref 才会解套。
从 Array 或者 Map 等原生集合类中访问 ref 时，不会自动解套

如果传入 ref 的是一个对象，将调用 reactive 方法进行深层响应转换

1、使用reactive方法创建的对象需要注意：

使用组合函数时必须始终保持对这个所返回对象的引用以保持响应性。这个对象不能被解构或展开

2、可以使用toRefs进行对象解构

toRefs：把一个响应式对象转换成普通对象，该普通对象的每个 property 都是一个 ref ，和响应式对象 property 一一对应

3、区别：

1）就像你在普通 JavaScript 中区别声明基础类型变量与对象变量时一样区别使用 ref 和 reactive

2）所有的地方都用 reactive，然后记得在组合函数返回响应式对象时使用 toRefs，虽然这样就不需要纠结使用reactive还是ref，但是我们还是需要他们的上述使用区别与注意点

三、computed的使用

1、传入一个 getter 函数，返回一个默认不可手动修改的 ref 对象

2、或者传入一个拥有 get 和 set 函数的对象，创建一个可手动修改的计算状态。

四、readonly的使用

1、传入一个对象（响应式或普通）或 ref，返回一个原始对象的只读代理。一个只读的代理是“深层的”，对象内部任何嵌套的属性也都是只读的

2、对于原始值的修改会触发只读代理的依赖追踪

五、watchEffect的使用

1、立即执行传入的一个函数，并响应式追踪其依赖，并在其依赖变更时重新运行该函数。

请注意，初始化运行是在组件 mounted 之前执行的。因此，如果你希望在编写副作用函数时访问 DOM（或模板 ref），请在 onMounted 钩子中进行

当一个 reactive 对象属性或一个 ref 作为依赖被追踪时，将调用 onTrack

依赖项变更导致副作用被触发时，将调用 onTrigger

2、停止侦听

1）在组件unmounted生命周期时自动停止

2）watchEffect会返回一个stop handler，我们可以直接调用它停止监听

3、清除副作用

有时副作用函数会执行一些异步的副作用，这些响应需要在其失效时清除（即完成之前状态已改变了）。所以侦听副作用传入的函数可以接收一个 onInvalidate 函数作入参，用来注册清理失效时的回调

五、watch的使用

1、与watchEffect对比

1）懒执行副作用；

2）更明确哪些状态的改变会触发侦听器重新运行副作用

3）访问侦听状态变化前后的值

2、与 watchEffect 共享的行为

watch 和 watchEffect 在停止侦听，清除副作用 (相应地 onInvalidate 会作为回调的第三个参数传入)，副作用执行时机

和 调试侦听 等方面行为一致

3、侦听器的数据源可以是一个拥有返回值的 getter 函数，也可以是 ref

4、侦听多个数据源

六、生命周期钩子函数

1、与 2.x 版本生命周期相对应的组合式 API

beforeCreate -> 使用 setup()
created -> 使用 setup()
beforeMount -> onBeforeMount
mounted -> onMounted
beforeUpdate -> onBeforeUpdate
updated -> onUpdated
beforeDestroy -> onBeforeUnmount
destroyed -> onUnmounted
errorCaptured -> onErrorCaptured

2、新增的钩子函数

除了和 2.x 生命周期等效项之外，组合式 API 还提供了以下调试钩子函数：
onRenderTracked
onRenderTriggered
两个钩子函数都接收一个 DebuggerEvent，
与 watchEffect 参数选项中的 onTrack 和 onTrigger 类似

七、依赖注入

1、使用介绍

你可以把依赖注入看作一部分“大范围有效的 prop”，除了：
1、祖先组件不需要知道哪些后代组件使用它提供的 property

2、后代组件不需要知道被注入的 property 来自哪里

2、新版本的使用方法

1）inject 接受一个可选的的默认值作为第二个参数。如果未提供默认值，并且在 provide 上下文中未找到该属性，则 inject 返回 undefined

2）注入的响应性

可以使用 ref 来保证 provided 和 injected 之间值的响应

如果注入一个响应式对象，则它的状态变化也可以被侦听

八、模版Refs

1）在setup中申明ref并返回，在模版中使用ref=''进行绑定

ref 被用在模板中时和其他 ref 一样：都是响应式的，并可以传递进组合函数（或从其中返回）

2）在v-for中使用

通过申明一个数组用于保存

九、响应式系统工具集

1、unref

如果参数是一个 ref 则返回它的 value，否则返回参数本身。它是 val = isRef(val) ? val.value : val 的语法糖。

2、toRef

toRef 可以用来为一个 reactive 对象的属性创建一个 ref。这个 ref 可以被传递并且能够保持响应性

3、toRefs

把一个响应式对象转换成普通对象，该普通对象的每个 property 都是一个 ref ，和响应式对象 property 一一对应。

4、isRef

检查一个值是否为一个 ref 对象

5、isProxy

检查一个对象是否是由 reactive 或者 readonly 方法创建的代理。

6、isReactive

检查一个对象是否是由 reactive 创建的响应式代理。

如果这个代理是由 readonly 创建的，但是包裹了由reactive 创建的对象，那么同样也会返回 true。

7、isReadonly

检查一个对象是否是由 readonly 创建的只读代理。