Reactivitat

La reactivitat es una de les peces clau en aquest nou model de programació dinàmic. Entendre a la perfecció el seu funcionament es clau per a poder desenvolupar amb eficiència i poder comprendre els errors d'asyncronia que ens podem trobar.

Observer pattern

La reactivitat es basa en el patró de disseny d'observables. Tradicionalment, quant es volia saber si hi havia un canvi en un element, s'havia de consultar periodicament demanant per nous canvis.

Aquest padró defineix una forma de comunicació distinta, on un observavor es subscriu a un subjecte observable i aquest subjecte notifica a l'observador els canvis.

Més informació

Reactivitat a Vue

La reactivitat a Vue és molt senzilla i transparent. Declarant un tipus de variables com a reactives automàticament activarà els canvis a la resta d'elements reactius com templates o computed.

Hem de tenir en compte que per defecte les variables normals, props... no son reactives, per tant canvis en aquestes no es veuràn reflectides de forma automàtica.

Hi ha bastantes formes d'usar la reactivitat, a continuació es detallen les més comuns.

ref

Forma més bàsica de reactivitat, basta declarar una variable com a ref

```
// const nomVariable = ref<typo>(valorInicial)
const nom = ref<string>('')
```

L'accés a les variables reactives dins el template es realitza indicant el nom directament. En canvi dins el codi s'ha de cridar la propietat `.value` de la variable.

```
<!-- Dins el template-->
{{ nom }}
```

```
// Al codi typescript
nom.value
```

computed

Les "variables" computed executen una funció per a calcular el seu valor i reaccionen als canvis de les variables reactives que s'hi defineixen.

Es a dir, es subscriuen als canvis de les variables reactives i executan la funció cada vegada que hi ha un canvi per a calcular el valor. El valor resultant també es reactiu.

```
// Calculam el nom complet a partir del nom i llinatges de la persona.
Notar que persona és tipus ref
const nom_complet = computed(() => {
  return persona.value.nom + ' ' + persona.value.llinatges
})
```

Per defecte les variables reactives no ens permeten canviar el valor, son readonly. Existeix una forma avançada de poder fer això que consisteix en definir un objecte amb les propietats get i set.

```
const variableNom = computed({
  get() {...}
  set(newVal) {...}
}
```

> Cas pràctic

El cas més útil per a usar aquesta forma avançada del computed és en combinació de les props, emits i la directiva v-model.

```
const value = computed({
  get () {
    return props.modelValue
  },
  set (newValue) {
    emit('update:modelValue', newValue)
  }
})
```

En aquest cas, la variable `value` agafa el valor de la propietat `modelValue` i cada vegada que es realitza un canvi a la variable value, s'actualitza de forma automàtica mitjançant l'emit. Això ens sinconitza la propietat del component pare i del fill de forma automàtica.

watch

Vue disposa d'un altre mecanisme per executar un codi si una variable reactiva canvia i es el watch. Al watch s'ha de definir un o multiples desencadenants, la funció a executar quant el desencadenant canvia i uns opcions de configuració

```
```

```
watch(desencadenant, funcio, opcions)

watch(
    persona, // al modificar la persona s'executa
    () => { // funció a executar
      ...
    },
    { // opcions (opcionals)
      immediate: true, // destacar immediate que executarà el codi al inici
encara que no hagi canviat
    }
  )
```

Notar que el desencadenant es una propietat reactiva (pot ser ref, computed...). En cas de voler observar una propietat no reactiva (per exemple les props d'un component) l'hem de transformar abans a reactiva o usar una funció en el seu lloc `() => props.persona`

toRef

La forma de convertir una propietat no reactiva a reactiva es mitjançant toRef.

```
const persona = toRef(() => props.persona)
```