Examen STW: Recuperació 2020

1 Exercici Vue.js

Heu de programar un component **<countdownButton>** que mostri els segons d'un compte enrere en un botó.

```
<countdownButton timeout="4" v-on:buzz="finalCountdown = true"></countdown>
```

- El paràmetre **timeout** d'aquest component indica el nombre de segons amb els que començar a comptar enrere.
- El template del component ha ser un **<button>** amb el nombre de segons restants.
- El botó estarà **disabled** si el comptador no ha arribat a zero. En canvi, estarà activat quan arribi a zero.
- El component ha d'emetre l'event **buzz** al fer click al botó.

1.1 El Resultat

Es proporciona la següent instancia de prova.

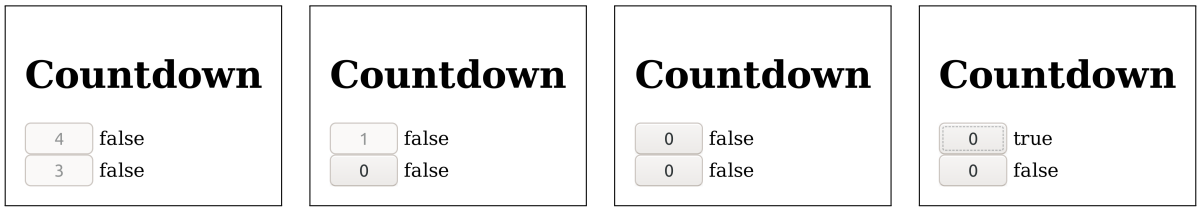
```
new Vue({
  el: "#app",
  data: {
    countdownOneClick: false,
    countdownTwoClick: false,
  },
  template: `<div>
    <countdownButton timeout="4"
            v-on:buzz="countdownOneClick = true"></countdownButton>

            {{countdownOneClick}} <br>

    <countdownButton timeout="3"
            v-on:buzz="countdownTwoClick = true"></countdownButton>

            {{countdownTwoClick}}
    </div>`,
})
```

El resultat ha de ser com el que es mostra a la Figura 1.



(a) Estat inicial. (b) Als 3 segons. (c) Un cop tots els comptadors han arribat al final. (d) Un cop tots els comptadors han arribat al final i s'ha fet click al primer botó.

Figure 1: Resultat a diversos instants de temps.

2 Exercici Javascript / Nodejs

En aquest exercici cal implementar el mètode **`interruptablePromise`**.

```
o = interruptablePromise(p)
```

Aquest mètode rep com a paràmetre una promise **`p`**. Aquest mètode retorna un objecte **`o`**, on **`o.q`** és una promesa i **`o.disable`** és una funcio.

Ha de fer el següent:

a) Si **`p`** es resol (**`resolve`**) i no s'ha cridat **`o.disable`** abans, aleshores la promesa **`q`** s'ha de resoldre igual que **`p`**. Es a dir, si **`p`** fa **`resolve(v)`** i no s'ha executat **`o.disable`**, aleshores **`q`** també ha de fer **`resolve(v)`**.

b) Si **`p`** es resol (**`resolve`**) i sí que s'ha cridat **`o.disable`** abans, aleshores no s'ha de fer res.

c) Si **`p`** es refusa (**`reject`**), aleshores la promesa **`q`** també s'ha de refusar (independentment d'haver cridat **`o.disable`**).

Nota: no feu servir variables globals.

Exemples d'ús:

```
const resolveIn = (v, t) => new Promise((resolve, reject) =>
                                          setTimeout(() => resolve(v), t));
const rejectIn = (v, t) => new Promise((resolve, reject) =>
                                         setTimeout(() => reject(v), t));

var o = interruptablePromise(Promise.resolve(1));
o.q.then(console.log); // Result: 1

var o = interruptablePromise(Promise.reject(2));
o.q.catch(console.log); // Result: 2

var o = interruptablePromise(resolveIn(3, 10));
o.q.then(console.log); // Result: 3

var o = interruptablePromise(resolveIn(4, 10));
o.disable();
o.q.then(console.log); // (nothing is printed)

var o = interruptablePromise(rejectIn(5, 10));
o.disable();
o.q.catch(console.log); // Result: 5

var o = interruptablePromise(Promise.resolve(6));
o.disable();
o.q.then(console.log); // (nothing is printed)

// Your tests here
```