

Examen STW: DEMO 1

1 Exercici Vue

Heu de programar una web amb Vue.js per mostrar el pronòstic meteorològic tal com es descriu a continuació.

1.1 Les dades

La previsió del temps de la setmana la teniu ja disponible a l'objecte JSON anomenat **forecast**.

```
const forecast = [  
  { weekday: "Wednesday", type: "Rainy", temperature: 20, scale: "celsius" },  
    /*...*/  
  { weekday: "Friday", type: "Rainy", temperature: 10, scale: "celsius" },  
];
```

1.2 El resultat

El pronostic s'ha de presentar tal i com es mostra a les Figs. 1 i 2.

Current Week Forecast										
View in <input type="button" value="°C"/> <input type="button" value="°F"/>										
Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny
20 °F	20 °F	20 °F	20 °F	20 °F	20 °F	20 °F	20 °F	20 °F	20 °F	20 °F

Figure 1: Exemple de com s'ha de veure la previsió en Fahrenheit.

Current Week Forecast										
View in <input type="button" value="°C"/> <input type="button" value="°F"/>										
Wednesday	Thursday	Friday	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny	Sunny
68 °C	68 °C	68 °C	68 °C	68 °C	68 °C	68 °C	68 °C	68 °C	68 °C	68 °C

Figure 2: Exemple de com s'ha de veure la previsió en Celcius.

Details

- S'han de poder canviar les unitats de les temperatures amb els botons corresponents.
- No es pot assumir que hi ha un número constant de columnes a la taula.

2 Exercici Javascript / Nodejs

En aquest exercici cal implementar el metode **raceN**.

```
result = raceN(list, N)
```

Aquest mètode rep com a paràmetres una llista de promises (**list**) i un enter (**N**). Aquest mètode retorna una promise (**result**).

Ha de fer el següent:

- La promise resultant s'ha de resoldre (amb **resolve**) just quan **N** promises de **list** s'hagin resolt (amb **resolve**). El resultat serà una llista amb els valors als que s'hagin resolt les promises de **list** (sense importar l'ordre).
- Si no s'arriba a tindre **N** promises resoltes correctament, la promesa resultant s'ha de rebutjar (amb **reject**) amb algun dels valors que haig retornat alguna de les promises rebutjades de **list**.
- Si **list.length < N** i no es rebutja cap promise de **list**, cal resoldre la promesa tal que a l'apartat a), però amb una llista més curta.
- Si **list** és una llista buida, la promesa resultant no s'ha de resoldre mai.

Nota: aquest exercici està inspirat en la funció **Promise.race** descrita a https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise/race.

Exemples d'ús:

```
// 1
p = raceN([Promise.resolve(1), Promise.resolve(2)], 1);
p.then(console.log);
// Resultat: [1]

// 2
p = raceN([Promise.resolve(1), Promise.resolve(2)], 2);
p.then(console.log);
// Resultat: [1, 2]

// 3
var plate = new Promise(function(resolve, reject) {
  setTimeout(() => resolve(1), 500);
});

p = raceN([plate, Promise.resolve(2)], 1);
p.then(console.log);
// Resultat: [2]

// 4
var plate = new Promise(function(resolve, reject) {
  setTimeout(() => reject(1), 500);
});

p = raceN([plate, Promise.resolve(2)], 2);
p.then(console.log).catch(e => console.log("err: " + e));
// Resultat: err: 1

// Your test here
```
