

# EXAMEN: Sistemes i Tecnologies Web Juny 2021

Niu:

Nom:

Permutació A

## Promise Logger

Si entregueu via [moixero.uab.cat](https://moixero.uab.cat), ompliu el següent requadre. Altrament, ompliu la resta de l'examen.

### Entrega Electrònica

The SHA1 checksum of the received file is:

.....

Time stamp is:

.....

**Guardeu-vos una còpia de l'arxiu que entregueu.**

Guia de correcció:

La nota sera...	Quan...	Guia de puntuació
De 0 a 5 punts	Quan no funciona tot i hi ha errors de concepte <i>greus</i> en algun dels següents elements clau: callbacks, clausures, classes, herència, promises, i els diversos elements de vue (reactivitat, directives, events, props, components, ...).	Es parteix d'un 5 i es resta 1 punt per cada error de concepte.
De 5 a 7 punts	Quan no s'aconsegueix fer funcionar tot l'examen i no hi ha errors de concepte <i>greus</i> .	Es parteix d'un 7 i es resten 0.25 punts per cada error.
De 7 a 10 punts	L'examen passa tots els tests i feu entrega electrònica.	Teniu un 10. Us l'heu guanyat.

## Instruccions

Seguiu les següents instruccions per a arrencar la màquina del laboratori i importar l'esquelet del projecte.

- Arrenqueu la màquina si no l'heu arrencada abans i seleccioneu la partició de Linux.
- Obriu una consola: Applications → Terminal.
- Descarregueu-vos l'esquelet del projecte executant la comanda:

```
wget http://moixero.uab.cat/ExamenSTW.zip
```

- Descomprimiu el fitxer zip.

```
7z x ExamenSTW.zip
```

Si la màquina a on esteu treballant es reinicia no perdeu la feina que heu fet.

- Feu l'examen (podeu fer servir el mateix terminal que ja teniu obert, i obrir l'arxiu els arxius `.js` amb el gedit).
- Per executar l'aplicació utilitzeu la comanda:

```
nodejs exam.js
```

- Un cop hagueu acabat de desenvolupar el projecte, si us funciona tota l'aplicació, haureu d'entregar el vostre codi electrònicament. **Aviseu-nos abans d'entregar electrònicament.**
- En el cas que no us funcioni, ompliu els forats de l'esquelet en aquest document.
- Si entregueu electrònicament, creeu un zip de la següent manera:

```
7z a sol.zip exam.js vue/app.js
```

- Comproveu el contingut de l'arxiu que entregareu (obriu l'arxiu i mireu el contingut dels arxius que hi ha a dintre).
- Un cop sapigueu segur que voleu entregar aquest arxiu, pugeu-lo a: <http://moixero.uab.cat/>.
- Anoteu els dos valors (el checksum i el timestamp) a l'examen en paper i entregueu l'examen en paper sense omplir els forats.
- Conserveu una còpia de l'arxiu entregat.

## Exercici backend

En aquest exercici es demana la implementació de tres funcionalitats, que ajudaran, conjuntament amb l'exercici de frontend, a visualitzar missatges de log. A continuació es presenten les especificacions que cal complir.

### Caching decorator

Volem crear un decorador de funcions asíncrones que “catchegi” les promeses resultants<sup>1</sup>. Cal desenvolupar la funció **cachingDecorator** que es descriu a continuació.

La funció es crida de la següent manera:

```
let g = cachingDecorator(f)
```

- La funció rep una funció **f** i retorna una altra funció, que podem anomenar **g**.
- Les funcions **f** i **g** són funcions d'un sol argument i que retornen una promesa.

Quan es crida **g(x)**, si no s'havia cridat ja amb l'argument **x**, aquesta ha d'executar **f(x)** i retornar el resultat d'aquesta execució. Per contra, si ja s'havia cridat amb anterioritat amb el mateix valor de **x**, aquesta ha de retornar la promesa resultant de l'execució anterior.

Exemple d'ús:

```
let f = x => Promise.resolve(x);
let g = cachingDecorator(f);

g(1).then(console.log); // Imprimeix '1' per consola (executa f).
g(2).then(console.log); // Imprimeix '2' per consola (executa f).
g(1).then(console.log); // Imprimeix '1' per consola (no executa f).
```

Indicacions:

- Es recomana guardar el resultat de l'execució de **f** en un diccionari, on **x** és la clau, i **f(x)** és el valor.
- Es pot fer servir **if (x in cache)** per consultar l'existència d'una clau en un diccionari.

Resultat del test (l'ordre pot variar):

```
1
2
1
resolving to 1
fulfilled 1
fulfilled 1
fulfilled 1
rejecting to 2
rejected 2
rejected 2
```

### Logging decorator

Volem crear un decorador de funcions asíncrones que “loggegi” el cicle de vida de les promeses retornades. Per a això, es demana desenvolupar la funció **loggingDecorator**.

La funció es crida de la següent manera:

```
let g = loggingDecorator(f, p, r, e)
```

- La funció rep quatre arguments: una funció **f**, i les cadenes de text **p**, **r** i **e**.

<sup>1</sup> Cal no confondre “les promeses resultants” amb “el resultat de les promeses”.

- La funció retorna una altra funció, que podem anomenar **g**.
- Les funcions **f** i **g** són funcions d'un sol argument i que retornen una promise.

La funció **g** ha de fer el mateix que **f**, excepte que addicionalment ha “loggejar” els següents missatges informatius:

- A l'obtindre la promesa resultant de **f** ha d'imprimir:

```
p + " (" + f.name + ") "
```

- Si la promesa resultant es satisfà, ha d'imprimir:

```
r + " (" + f.name + ", resolved, " + res + ") "
```

on **res** es el valor al que es s'ha 'settled' la promesa.

- Si la promesa resultant es refusa, ha d'imprimir:

```
e + " (" + f.name + ", rejected, " + err + ") "
```

on **err** es el valor al que es s'ha 'settled' la promesa.

Exemple d'ús:

```
let f = x => Promise.resolve(x);
let g = loggingDecorator(f, "a", "b", "c");
g(3)
```

L'exemple anterior ha imprimir el següent text a la consola:

```
a (f)
b (f, resolved, 3)
```

Resultat del test:

```
a (f)
a2 (f2)
a3 (f3)
b (f, resolved, 1)
c3 (f3, rejected, 3)
g3 rejected with 3
b2 (f2, resolved, 2)
```

## Console replacement

Volem substituir l'objecte **console** per un d'equivalent, amb un registre global de missatges tal com s'especifica a continuació.

- Cal crear la variable global **global\_log** i inicialitzar-la (es dona fet).
- Cal substituir l'objecte global **console** per un de millorat. En aquest nou **console**, la funció **log** ha de imprimir el missatge per consola i també ha d'afegir aquest missatge al final de la llista **global\_log**. És important fer-ho en aquest ordre: primer imprimir i després afegir.

Ha de funcionar de la següent manera:

```
console.log("hola"); // Això ha d'escriure "hola" per consola.
console.log("adeu"); // Això ha d'escriure "adeu" per consola.
// Aquí la variable 'global_log' ha de contindre [ ..., "hola", "adeu" ].
```

- Cal fer servir herència perquè l'objecte substituït hereti tota la funcionalitat de l'anterior (per exemple, el mètode **assert** ha de seguir funcionant).
- Cal fer tot això en un private scope (es dona fet).

Resultat del test:

```
hola
adeu
[
  'Lorem ipsum dolor sit amet',
  'consectetur adipiscing elit',
  'sed do eiusmod tempor incididunt...',
  'hola',
  'adeu'
]
```

## Test

Si tot us funciona correctament, l'execució de l'arxiu **exam.js** ha de donar:

```
a (f)
a2 (f2)
a3 (f3)
hola
adeu
[
  'Lorem ipsum dolor sit amet',
  'consectetur adipiscing elit',
  'sed do eiusmod tempor incididunt...',
  'hola',
  'adeu'
]
1
2
1
b (f, resolved, 1)
resolving to 1
fulfilled 1
fulfilled 1
fulfilled 1
rejecting to 2
rejected 2
rejected 2
c3 (f3, rejected, 3)
g3 rejected with 3
b2 (f2, resolved, 2)
Web Server Started at http://localhost:8080
```

## Resposta

Ompliu els següents espais. Recordeu que la mida no dona cap indicació del nombre de línies de codi que us calen.

```
// BACKEND EXERCISE  
// 1- CACHING DECORATOR
```

```
{ /* Tests [...] */ }  
  
// 2- LOGGING DECORATOR
```

```
{ /* Tests [...] */ }
```

```
// 3- CONSOLE REPLACEMENT
```

```
(function() {
```

```
})();
```

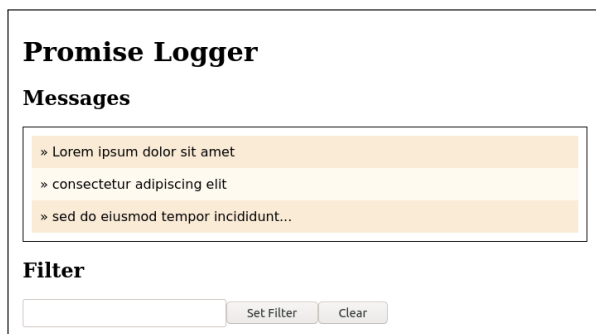
```
{ /* Tests [...] */ }
```

```
{ /* Web server stuff [...] */ }
```


## Exercici frontend

### Què cal fer?

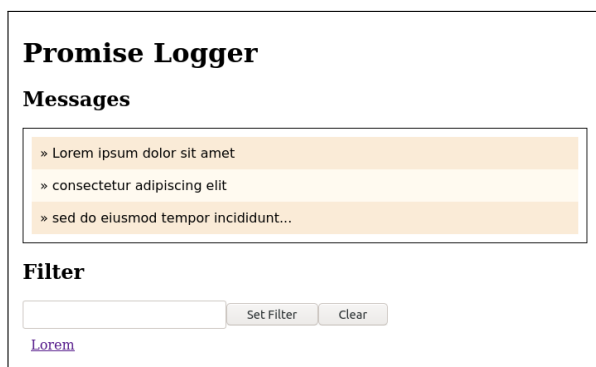
Cal implementar una interfície gràfica amb Vue.js que representi visualment una llista de missatges de log, i que permeti filtrar-los. A la Fig. 1 es mostra com ha de quedar el resultat.



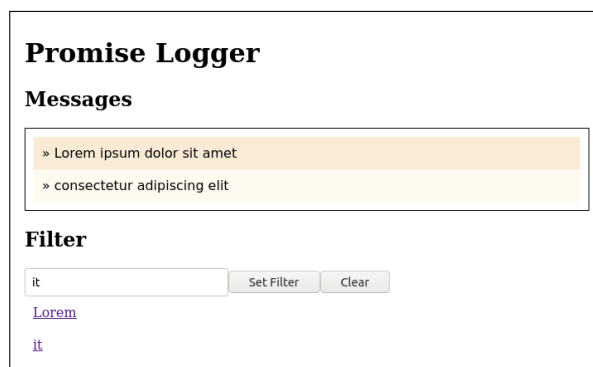
(a) Estat inicial.



(b) Resultat al filtrar pel terme 'Lorem'.



(c) Resultat al filtrar pel terme 'Lorem' i fer click a 'Clear'.



(d) Resultat al filtrar pel terme 'it' després de filtrar pel terme 'Lorem'.

Figure 1: Resultat desitjat. En aquest cas, l'exercici anterior no està implementat.

## Especificacions

- El visualitzador està compost per un component arrel **LogViewer** que inclou la llista de missatges i un component **FilterComponent** que inclou els elements destinats a la selecció del filtre.
- El component **LogViewer** ha de generar el codi html tal que el següent:

```
<h2>Messages</h2>
<div>
  <ul>
    <li>Lorem ipsum dolor sit amet</li>
    <li>consectetur adipiscing elit</li>
    <li>sed do eiusmod tempor incididunt...</li>
  </ul>
  <!-- FilterComponent -->
</div>
```

Cal repetir un tag `<li>` per cada missatge de log, i cal incloure el **FilterComponent** al lloc indicat pel comentari.

- El component **LogViewer** ha de rebre els missatges de log en format JSON mitjançant la url:

**http://localhost:8080/global\_log**



Podeu fer servir

```
fetch(url).then(res => res.json()).then();
```

per accedir a l'API REST.

- El component **FilterComponent** ha de generar el codi html tal que el següent:

```
<div>
  <h2>Filter</h2>
  <input>
  <button>Set Filter</button>
  <button>Clear</button>
  <div class="stored-filter"><a href="#">Lorem</a></div>
  <div class="stored-filter"><a href="#">it</a></div>
</div>
```

- Al clicar al botó 'Set Filter', els missatges de **LogViewer** s'han de filtrar automàticament d'acord amb el contingut del **<input>**. S'han de mostrar els elements que contenen el contingut de input.

Indicació: **haystack.indexOf(needle) >= 0**

- Cal incloure un històric dels elements pels que s'ha filtrat anteriorment, que permetin tornar a filtrar pel mateix text al fer-hi clic. Cal repetir el **<div>** pertinent per a que això sigui possible. Al filtrar per un element històric, no s'ha de tornar a afegir a la llista dels elements històrics.
- Al clicar al botó 'Clear', s'ha de fixar el text a buscar a la cadena buida i no actualitzar l'històric.

## Notes

- S'ús dona un esquelet d'aquesta interfície web configurat amb:

```
app.use(express.static(path.join(__dirname, 'vue')));
```

- La funció **[] .filter(test)** crea una nova llista amb tots els elements que passen el test.

```
[1,2,3].filter(x => x == 2); // retorna [ 2 ]
```

- Cal fer servir vue versió 3.
- No es pot fer servir async/await.
- No es poden fer servir class expressions (class syntactic sugar).

```
const LogViewer = {
```

```
}
```

```
const FilterComponent = {

}

let app = Vue.createApp(LogViewer);

//
```