EXAMEN: Sistemes i Tecnologies Web Juny 2022

Niu: Nom:

Permutació A

☞ Booking

Si entregueu via 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 copia de l'arxiu que entregueu.

This English version of the exam is given as a courtesy. Any text in the original version of the exam takes precedence to the contents of this document.

Guia de correció:

La nota serà...	Quan...	Guia de puntuació
De 0 a 5 punts	Quan no funciona tot i hi ha errors de concepte *greus* 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 greus.	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 https://moixero.uab.cat/ExamenSTW.zip
  ```

- Descomprimiu el fitxer zip.

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

- 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: `https://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.

Exercici frontend

Context

Suppose that we are working on a booking portal for flights such as the one shown in Figure 1.

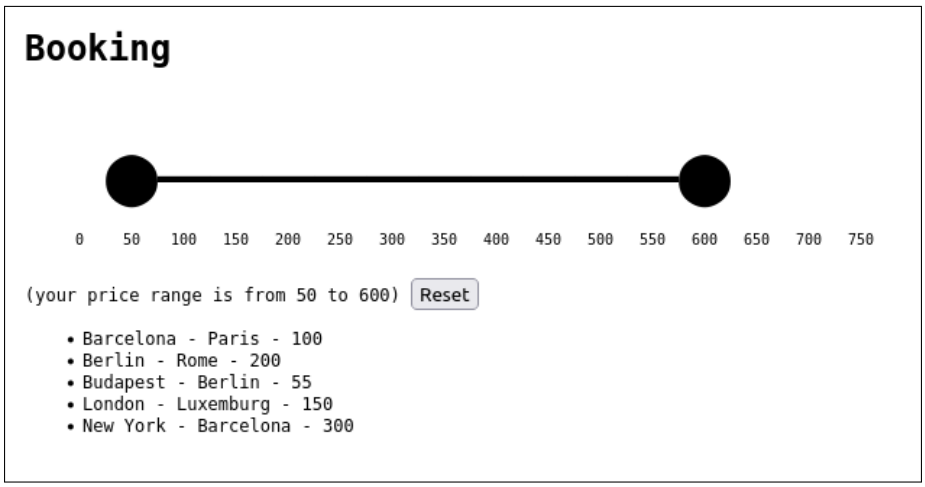


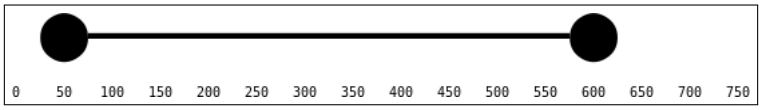
Figure 1: Flight booking portal.

In this portal there is a price range selector (above) and a list of flights that match that price range (below). In the middle, there is a message regarding the selected range and a button that resets the price range.

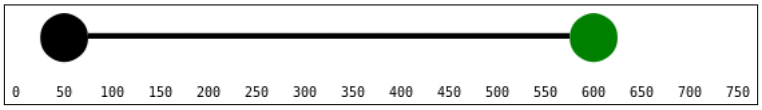
You can start this portal by running the **`exam.js`** file.

What to do

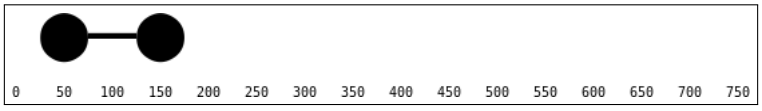
The portal is given to you mostly complete, with the exception of a single Vue.js component named **`RangeFilterComponent`**. This component will select a price range and it is what you need to implement. Figure 2 show the expected result.



(a) Estat inicial.



(b) Resultat de clicar al cercle que hi ha a sobre del text '600'.



(c) Resultat de clicar al cercle que hi ha a sobre del text '600' tal que a (b), i seguidament tornar a clicar a l'espai que hi ha a sobre del text '150'.

Figure 2: Expected result.

You must not modify any other frontend part.

Specifications

– You need to implement the template of **`RangeFilterComponent`** according with the examples given in **`index.html`**, and that are shown in Figure 3.

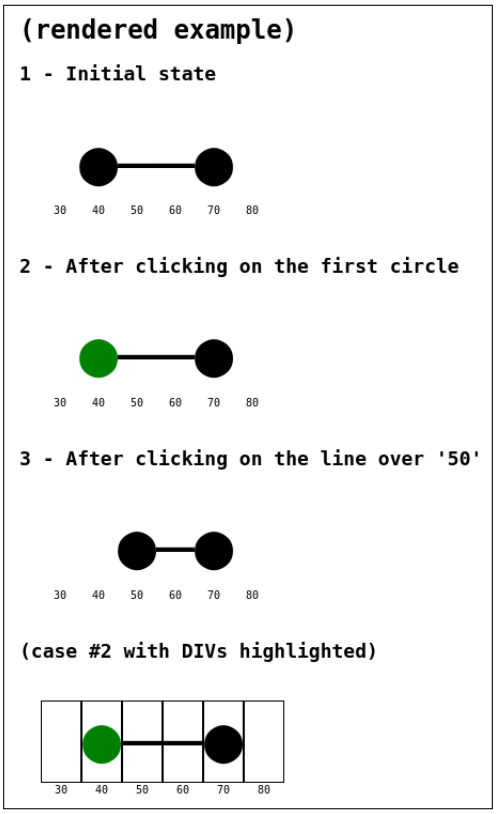


Figure 3: Expected result.

– The component is composed by a sequence of **`div`** tags, which contain either:

  – the UNICODE character '●',
  – the UNICODE character '–', or
  – nothing.

  Hence, the last example in Figure 3 corresponds to: '' (nothing), '●', '–', '–', '●', '' (nothing). The rendered HTML result must resemble:

```
<div class="range">
  <div>
          <div class="range-item"> </div>
          <span class="range-threshold">30</span>
  </div>
  <div>
          <div class="range-item" style="color: green">●</div>
          <span class="range-threshold">40</span>
  </div>
  <div>
          <div class="range-item">–</div>
          <span class="range-threshold">50</span>
  </div>
  <div>
          <div class="range-item">–</div>
          <span class="range-threshold">60</span>
  </div>
```

```
<div>
        <div class="range-item">●</div>
        <span class="range-threshold">70</span>
</div>
<div>
        <div class="range-item"></div>
        <span class="range-threshold">80</span>
</div>
</div>
```

Note: you can find (and copy) these characters and the rest of the tags in the source code accompanying this document.

- One of the bullets must correspond to the selected minimum value and another to the selected maximum value. Dashes must fill the spaces between the minimum and maximum bullets, whenever the minimum bullet is to the left of the maximum bullet.
- The component that you need to implement is to be used as follows:

```
<Range-Filter v-bind:thresholds="thresholds" v-model="range">
</Range-Filter>
```

where **thresholds** is a list of integer numbers, and where **range** is a dictionary with fields **min** and **max**.

  - The attribute **v-bind:thresholds** of the component tag indicates each of the price points that can be selected. In the example in Figure 3 this would be **thresholds = [30, 40, 50, 60, 70, 80]**.
  - The attribute **v-model** must enable *two-way data binding* of the **range** dictionary, according to the user interaction with the component. I.e., **range.min** and **range.max** have to match the values associated with the bullet positions. In the example in Figure 3 this would be **range = {min: 40, max: 70}**, except for step 3, which would be **range = {min: 50, max: 70}**.

- Bullets have to change color and position according to the following indications:
  - On clicking in a bullet, this bullet has to turn green.
  - On clicking again on the same bullet, the bullet has to turn black.
  - If two bullets are green, the first to turn green has to turn black.
  - If there is a green bullet and a **div** tag that does not contain a bullet is clicked, the green bullet has to be moved to the clicked **div** tag and turn black.
- Bullets have to move (automatically) whenever the value of **range** changes from outside the component through *two-way data binding*. When thus moved, bullets need to turn black.

Notes

- The web interface is already configured with:

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

- Use vue 3.
- You can *deep watch* an object field with:

```
watch: {
    'object.field': function() { ... },
}
```

- Recall that these two tags are equivalent:

```
<Component v-model="data" />
<Component v-bind:modelValue="data"
        v-on:update:modelValue="x => data = x" />
```

- Do not use async, nor await.
- You cannot use class expressions (class syntactic sugar).

```
const RangeFilterComponent = {




// (object continues on next page)
```

```
// (continues from previous page)









}

const app = Vue.createApp(RootComponent);
app.component('Range-Filter', RangeFilterComponent);
const vm = app.mount("#app");
```

Exercici backend

Context

Suppose that we are implementing the backend part of the portal shown in Figure 1. In this backend we have a series of flight information providers, and we have to aggregate their results to display them in the portal.

There are four information providers:: **`AirEuropa`**, **`Delta`**, **`Ryanair`**, i **`Vueling`**. These providers are already available, and in practice these are objects which implement the **`getFlights`** method, which when called returns a promise that resolves to a flight list. For example,

```
p = AirEuropa.getFlights()
```

returns a promise that resolves to:

```
[
  { from: 'Barcelona', to: 'Paris', price: 100 },
  { from: 'Berlin', to: 'Rome', price: 200 },
]
```

Occasionally, these providers will fail and will not resolve their promise in time.

The `flightsServer` agregator

You need to code the **`flightsServer`** object. This object has to implement the **`getFilteredFlights`** method, which tries to obtain all flights from the aforementioned information providers and filters them according to a minimum and maximum price. I.e.,

```
p = flightsServer.getFilteredFlights(min, max)
```

where **`p`** will be a promise that will be resolved to a list such as the one provided by information providers..

- The result of **`getFilteredFlights`** needs to be created considering all results returned (in time) when calling **`getFlights`** for all information providers.

  If an information providers takes more than 500 ms to resolve their returned promise, this providers must be ignored and the result must only contain the information provided by the other providers.

  If no providers takes more than 500 ms to resolve a their returned promise, the **`getFilteredFlights`** method has to resolve its promise as soon as possible.

- The **`getFilteredFlights`** method has to exclude from its result the flights with **`price`** strictly less that **`min`** and strictly greater than **`max`**.

Usage examples:

- Calling **`flightsServer.getFilteredFlights(90, 110)`** has to return a promise that resolves to:

  ```
  [{"from":"Barcelona","to":"Paris","price":100}]
  ```

- Calling **`flightsServer.getFilteredFlights(100, 600)`** has to return a promise that resolves to:

  ```
  [{"from":"Barcelona","to":"Paris","price":100},
   {"from":"Berlin","to":"Rome","price":200},
   {"from":"London","to":"Luxemburg","price":150},
   {"from":"New York","to":"Barcelona","price":300}]
  ```

- Calling **`flightsServer.getFilteredFlights(50, 60)`** has to return a promise that alternately resolves to **`[{"from":"Budapest","to":"Berlin","price":55}]`** and to **`[]`**.

Notes

- Do not use async, nor await.
- You cannot use class expressions (class syntactic sugar).

Resposta

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

```
const flightsServer = {

// (object continues on next page)
```

```
// (continues from previous page)

}
```