EXAMEN: Sistemes i Tecnologies Web Juny 2024

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.

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

Suposem que estem treballant en un portal de reserva de vols d'avió tal el que es mostra a la Figura 1.

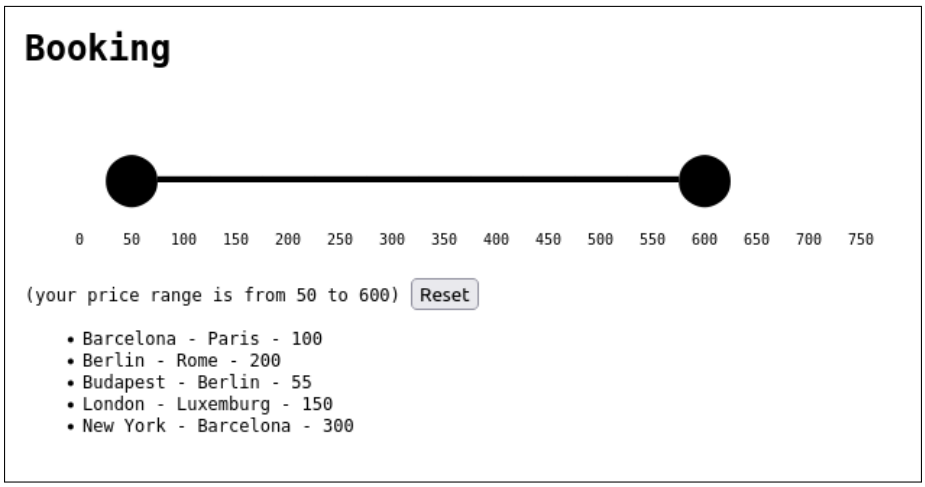


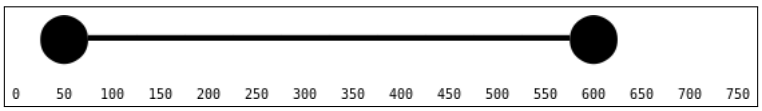
Figure 1: Portal de reserva de vols.

En aquest portal s'hi mostra un selector de preus (a la part superior) i una llista de vols que compleixen aquesta selecció (a la part inferior). Entre les dues parts anteriors, hi ha un missatge que mostra el rang seleccionat, i un botó que permet reiniciar el rang de preus.

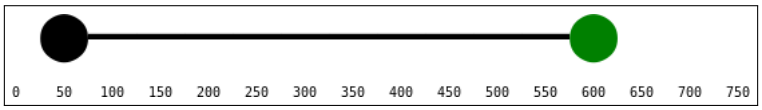
Podeu arrancar aquest portal executant l'arxiu **`exam.js`**.

Què cal fer?

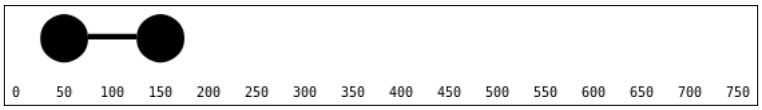
Aquest portal es dona ja fet, amb l'excepció d'un únic component de Vue anomenat **`RangeFilter`**. Aquest component farà la selecció del rang de preus pels quals filtrar i és el que s'ha d'implementar. A la Figura 2 es mostra com ha de quedar el resultat.



(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: Resultat desitjat.

No s'ha de modificar cap de les altres parts del frontend. Només el component **`RangeFilter`**. En particular **no** s'ha de modificar el component **`RootComponent`**.

Especificacions

– Cal implementar el template del **`RangeFilter`** d'acord amb els exemples que es donen a l'arxiu **`index.html`**, i que es mostren a la Figura 3.

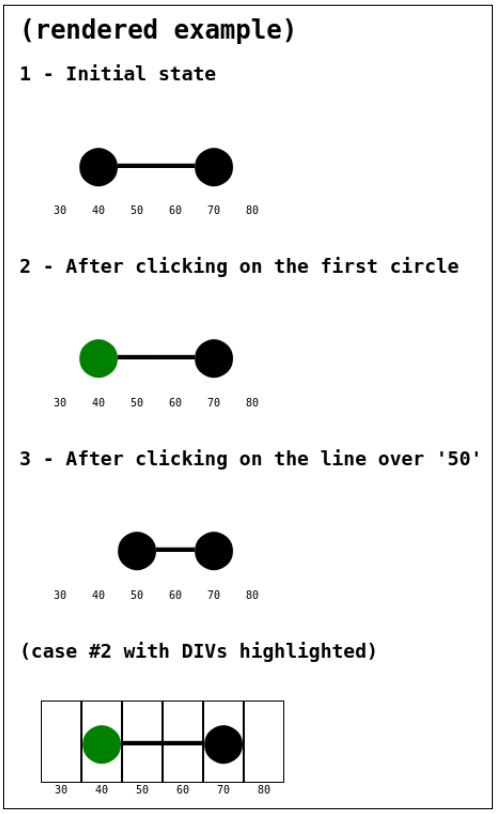


Figure 3: Resultat desitjat.

– El component ha d'estar format per una seqüencia de tags **`div`** en els que s'hi representa:

  – o bé el caràcter UNICODE '●',
  – o bé el caràcter UNICODE '–',
  – o bé res.

  Així doncs, l'últim exemple de la Figura 3 correspon a: '' (res), '●', '–', '–', '●', '' (res). El HTML renderitzat ha de ser tal que el següent:

```
<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>
```

Nota: podeu trobar (i copiar) aquests caràcters i la resta de tags a l'esquelet que acompanya aquest enunciat.

- Un dels cercles correspondrà al valor mínim seleccionat i un altre al valor màxim seleccionat. Entre el valor mínim i màxim s'hi representaran guions, sempre que el valor mínim estigui a l'esquerra del valor màxim.
- El component que heu d'implementar s'ha de poder fer servir de la següent manera:

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

on **`thresholds`** és una llista de nombres enters, i on **`range`** és un diccionari amb els camps **`min`** i **`max`**.

  - L'atribut **`v-bind:thresholds`** del component ha de marcar cadascun dels preus que s'han de poder seleccionar. A l'exemple de la Figura 3 això seria **`thresholds = [30, 40, 50, 60, 70, 80]`**.
  - L'atribut **`v-model`** ha de proporcionar un *two-way data binding* del diccionari **`range`** corresponent a la interacció de l'usuari amb el component. És a dir, **`range.min`** i **`range.max`** han de correspondre amb els valors associats a les posicions dels cercles. A l'exemple de la Figura 3 això seria **`range = {min: 40, max: 70}`**, excepte pel pas 3 que seria **`range = {min: 50, max: 70}`**.

- Els cercles han de canviar de color i posició d'acord amb les següents instruccions:
  - Al clicar en un dels dos cercles, aquest ha de canviar de color negre a color verd.
  - Al tornar-hi a clicar, si era verd, ha de tornar al color inicial.
  - Si hi ha dos cercles verds, es canviarà a negre el cercle al que no s'hi acaba de fer clic.
  - Si hi ha un cercle verd i es fa clic a un **`div`** que no contingui un cercle, el cercle verd es mourà a on s'ha fet clic i es tornara de color negre.
- Els cercles també es mouran (automàticament) quan canviï el valor de **`range`** des de fora del component mitjançant *two-way data binding*. Quan així es moguin, es tornaran negres.

Notes

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

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

- Cal fer servir vue versió 3.
- Podeu fer un *deep watch* d'un camp d'un objecte amb la següent sintaxi:

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

- Recordeu que els dos tags següents són equivalents:

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

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

Exercici backend

Context

Suposem que estem implementant una part del backend del portal que es mostra a la Figura 1. En aquest backend tenim una serie de proveïdors d'informació de vols, i els hem d'agregar per proporcionar la informació que es mostrarà al portal.

Hi ha quatre proveïdors d'informació: **`AirEuropa`**, **`Delta`**, **`Ryanair`**, i **`Vueling`**. Aquests proveïdors ja s'us proporcionen, i en realitat es tracta d'objectes que implementen el mètode **`getFlights`**, tal que quan es crida retorna una promesa que es resoldrà a una llista de vols. Per exemple,

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

retornarà una promesa que es resoldrà a:

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

Hi haurà casos en que aquests proveïdors fallaran i no resoldran la promesa a temps.

Agregador `flightsServer`

Cal programar l'objecte **`flightsServer`**. Aquest objecte ha d'implementar el mètode **`getFilteredFlights`**, que mirarà d'obtenir tots els vols dels proveïdors anteriorment mencionats i els filtrarà d'acord amb un preu mínim i màxim. I.e.,

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

on **`p`** serà una promesa que es resoldrà a una llista tal que les retornades pels proveïdors d'informació.

- El resultat de **`getFilteredFlights`** s'ha d'obtindre considerant tots els resultats retornats (a temps) al cridar **`getFlights`** per tots els proveïdors d'informació.

  Si un proveïdor triga més de 500 ms a resoldre la promesa que retorni, aquest proveïdor s'ha d'ignorar i el resultat ha de contenir la informació proporcionada per els altres proveïdors.

  Si cap proveïdor triga més de 500 ms a resoldre la promesa que retorni, el mètode **`getFilteredFlights`** ha de retornar els resultats tant aviat com pugui.

- El mètode **`getFilteredFlights`** ha d'excloure del seu resultat els vols amb **`price`** estrictament menor que **`min`** i estrictament major que **`max`**.

Exemple d'ús:

- Cridar **`flightsServer.getFilteredFlights(90, 110)`** ha de retornar una promesa que s'ha de resoldre a:

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

- Cridar **`flightsServer.getFilteredFlights(100, 600)`** ha de retornar una promesa que s'ha de resoldre a:

  ```
  [{"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}]
  ```

- Cridar **`flightsServer.getFilteredFlights(50, 60)`** ha de retornar una promesa que s'ha de resoldre de forma alternada a **`[{"from":"Budapest","to":"Berlin","price":55}]`** i a **`[]`**.

Notes

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