EXAMEN: Sistemes i Tecnologies Web Juny 2022

Niu: Nom:

Permutació A

☞ Catcha

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.

© Universitat Autònoma de Barcelona, 2022. All rights reserved. No publication or reproduction is allowed without prior written permission from Universitat Autònoma de Barcelona.

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
  ```

- Si heu aconseguit fer funcionar tota l'aplicació al 100%, **aviseu-nos abans d'entregar**.
- Per entregar el resultat de l'examen, cal que ho feu electrònicament, creant un zip de la següent manera:

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

- Calculeu el hash de l'arxiu:

  ```
  sha1sum sol.zip
  ```

- 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/`.
- **Verifiqueu que el hash de l'arxiu coincideix.**
- Anoteu els dos valors (el checksum i el timestamp) a l'examen en paper i entregueu l'examen en paper.

© Universitat Autònoma de Barcelona, 2022.

Frontend

Context

Hem de desenvolupar un "Completely Automated Public Turing test to tell Computers and Humans Apart". És a dir, hem de fer un CAPTCHA tal el que es mostra a la Figura 1. En aquest cas, caldrà seleccionar els gats i per tant serà un "CATcha".

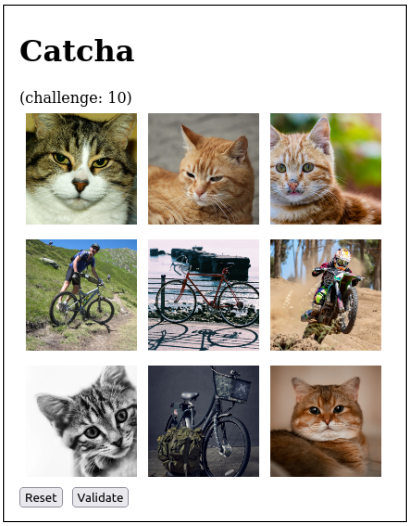


Figure 1: Un "catcha".

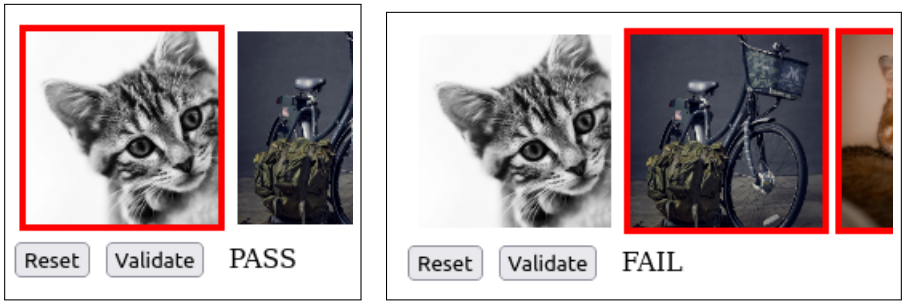
Què cal fer?

La majoria del frontend del "catcha" ja es dona fet, i simplement cal implementar un petit component de Vue.js anomenat **`SelectableImageComponent`**. Aquest component mostrarà una imatge que és podrà seleccionar i deseleccionar. A la Figura 2 es mostra com ha de quedar el resultat de 9 d'aquests components.

Figure 2: Un "catcha" on s'ha fet click a les cinc imatges on hi ha un gat. En aquesta imatge s'hi mostren 9 components **`SelectableImageComponent`** disposats en files i columnes.

Un cop seleccionades les imatges desitjades, el botó "Validate" enviarà aquesta seleció al backend i renderitzará el resultat de la validació tal i com es mostra a la Figura 3. Aquesta funcionalitat ja està implementada.

© Universitat Autònoma de Barcelona, 2022.



(a) Resultat d'una validació amb èxit.

(b) Resultat d'una validació fallida.

Figure 3: Resultat desitjat.

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

Especificacions

- El **`SelectableImageComponent`** s'ha de renderitzar com un simple tag **`<img>`**, que podrà tindre o no tindre vora. Si té vora, direm que està seleccionat. Direm que no està seleccionat en el cas contrari.

  El resultat ha de ser tal que el següent códi html en cas que no tingui vora:

  ```
  <img src="cat_or_bike_image.png" style="">
  ```

  O tal que el següent en cas que sí que en tingui:

  ```
  <img src="cat_or_bike_image.png" style="border: 4px red solid;">
  ```

- Inicialment, un **`SelectableImageComponent`** no tindrà vora. Al fer clic a un **`SelectableImageComponent`**, si aquest no tenia vora, en tindrà, i si sí que en tenia de vora, aleshores en deixarà de tindre.

- El component que heu d'implementar s'ha de poder fer servir de la següent manera:

  ```
  <selectable-image-component src="cat_or_bike_image.png"
          v-model="selected" />
  ```

  on **`src`** serà el que es farà servir a l'attribut **`src`** del tag **`<img>`**, i on **`selected`** serà una variable que valdrà **`true`** si el component està seleccionat i valdrà **`false`** si no ho està.

- El binding de la variable **`selected`** ha de ser bidireccional.

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.

- 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).

- No es pot manipular el DOM sense fer servir vue.

© Universitat Autònoma de Barcelona, 2022.

Backend

El backend del "catcha" el tenim gairebé acabat i només ens cal implementar la classe **Catcha**. Per implementar-la, ens caldrà fer servir un "proveidor criptogràfic".

El proveidor criptogràfic

El proveidor criptogràfic següent genera una seqüencia determinista de valors **true** i **false**.

```
const convolutedCryptoProvider = function(seed) {
  return new Promise(function(resolve, reject) {
    resolve({
        boolean: seed % 2 == 0,
        next: () => convolutedCryptoProvider((seed * 13 + 1) % 25)
      });
  });
}
```

Us el podeu imaginar com una especie de **random** que genera valors **true** i **false** a partir d'un **seed**.
Per exemple, si **r1** és el valor al que es resol la promise que es retorna al cridar a **convolutedCryptoProvider(7)**, veurem que:

```
r1 = { boolean: false, next: [Function: next] }
```

De **r1** en traiem un primer valor booleà, i si cridem **r1.next**, obtenim una nova promise que es resoldrà a, digue'm-li, **r2**:

```
r2 = { boolean: false, next: [Function: next] }
```

De **r2** podem treure un segón valor booleà, i així successivament:

```
r3 = { boolean: true, next: [Function: next] }
r4 = { boolean: true, next: [Function: next] }
r5 = { boolean: false, next: [Function: next] }
// ...
```

La llista de valors booleans (opcional)

En aquest punt es recomana que implementeu la funció **getBooleanList**.

```
p = getBooleanList(seed)
```

Aquesta funció fa servir un **seed** per generar una llista de 9 booleans amb **convolutedCryptoProvider**, i aquests es retornen en forma de promesa. És a dir que **getBooleanList(7)** retornarà:

```
Promise { [ false, false, true, true, false, false, true, true, false ] }
```

Indicació: podeu fer servir una estrategia recursiva com es va veure a la pràctica del progressbar... o podeu mirar d'explotar (at your own risk) el fet que només calen 9 booleans.

La classe Catcha

La classe **Catcha**, que heu d'implementar, s'haurà de poder fer servir de la següent manera:

```
const challenge = 7;

let catcha = new Catcha(challenge)

promise_images = catcha.getImages()
promise_result = catcha.checkAnswer(selected)
```

© Universitat Autònoma de Barcelona, 2022.

- Un cop construit, un **`catcha`** equivaldrà a una llista de 9 booleans que no canviarà mai. Aquesta llista serà generada deterministicament pel **`convolutedCryptoProvider`**, fent servir el paràmetre **`challenge`** del constructor com a **`seed`**.
- La funció **`getImages`** retornarà una promise que es resoldrà a la llista d'imatges del **`catcha`**. Retornarà una imatge d'un gat per cada **`true`** generat pel **`convolutedCryptoProvider`**, i una imatge d'una bicicleta per cada **`false`** generat pel **`convolutedCryptoProvider`**. En el mateix ordre.

  Per exemple, quan **`challenge`** sigui 7, **`getImages`** retornarà una promise que es resoldrà a:

  ```
  [ 'img/bike1.png', 'img/bike2.png', 'img/cat1.png', 'img/cat2.png',
    'img/bike3.png', 'img/bike4.png', 'img/cat3.png', 'img/cat4.png',
    'img/bike5.png' ]
  ```

  Nótis la similitud del resultat amb el resultat de **`getBooleanList`**.
- Per altra banda, la funció **`checkAnswer`** rebrà una llista de valors booleans i retornarà una promesa que es resoldrà a un booleà. El resultat de **`checkAnswer`** serà **`true`** si la llista de valors booleans és la correcta. És a dir, la que correspongui amb els valors generats pel **`convolutedCryptoProvider`**.

  Per exemple, quan **`challenge`** sigui 7, cridant **`checkAnswer`** amb argument

  ```
  [ false, false, true, true, false, false, true, true, false ]
  ```

  retornarà una promise que es resoldrà a **`true`**, mentre que cridant **`checkAnswer`** amb argument

  ```
  [ true, false, false, false, false, false, false, false, true ]
  ```

  retornarà una promise que es resoldrà a **`false`**.

Notes

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

© Universitat Autònoma de Barcelona, 2022.