2022.10.05

Challenge tehnic - VueJS

1. Implementeaza o lista cu abilitati de CRUD intr-un framework de front-end (un field cu creare de produs, update stergere de produse)

2. Fetch pe un api care returneaza un obiect(puncte bonus daca legi cu exercitiul 1 si afisezi in front-end produsele returnate din api)

3. Explicati care este diferenta dintre Hard Copy si Soft Copy(ex: al unui array).

4. Scrieti o functie care primeste un obiect si intoarce un hard copy al acestuia, fara a folosi Object.assign sau spread operator (...).

5. Scrieti o functie care primeste 3 parametrii (arr, left, right) - un array, un numar de rotatii la stanga si un numar de rotatii la dreapta. Functia intoarce array-ul dupa efectuarea rotatiilor.

6. Functie care traverseaza un obiect cu mai multe obiecte nestate (un tree) si returneza obiectul cu un id specificat(ex: functia se va apela searchTreeById(tree ,4) cauta obiectul cu id-ul 4 din obiectul tree (de mai jos), de menitonat ca obiectele din children pot avea si ele la randul lor alti children si asa mai departe.

tree = {id: 1,name: 'first branch',children: [{id:2, children:[...]}, {id:3, children:[...]}]}

7. Implementeaza o functie care compara doua obiecte si returneaza true daca field urile acestora sunt diferite (sa se considere si posibilitatea sa aiba array uri sau alte obiecte nestate)

8. Diferenta dintre process.nextTick() si setImmediate()? (cu exemple)

9. Implementare store (ex: Vuex/Redux) comun de aplicatii dintr-un SPA.

10. Tocmai ce ai fost pus la conducerea unui proiect legacy code a carui mentenanta este costisitoare, ce ai imbunatati pentru a face proiectul mai usor de intretinut in the long run?

11. La ce te uiti atunci cand analizezi sa verifici si sa imbunatatesti o aplicatie din punct de vedere al securitatii si performantei?

Informatii suplimentare:

Timp estimat de lucru: 2 ore.
Se accepta si rezolvari partiale.
Rezolvarile vor fi trimise la adresa de email: people@tarainteractive.com pana vineri, ora 17.00.

Mult succes!

Challenge tehnic VueJS – Bardita Andrei

1. https://github.com/BMAWebDev/vue-CRUD-products
2. La fel ca la 1 (a se vedea proiectul de pe git)
3. Soft copy reprezinta un format digital al unui obiect (ex: foaie A4 sub forma unui pdf), iar hard copy reprezinta formatul fizic (foaia printata). Nu exista aceste tipuri de format pentru un array, maxim sa fie un soft copy. Ma gandesc ca intrebarea face referinta la deep vs shallow copy, unde diferenta dintre cele doua o face copierea dupa referinta (pointer) in cazul shallow copy, pe cand la deep copy se copiaza tot array-ul ca fiind unul nou de sine statator. Modificarea arrayului nou in cazul shallow copy aduce schimbari in ambele array-uri, nu doar in cel nou. De exemplu:

```
const x = [1, [2, 3]];
const y = x;

y[1] = [3, 4];

console.log(y); // rezultat: [1, [3, 4]]
console.log(x); // rezultat: [1, [3, 4]] <-- aici e problema,
deoarece modificarea lui y se face prin referinta
```

4. Functie care intoarce un deep copy (nu hard copy!)

```
const deepCopy = (object) => {
  return JSON.parse(JSON.stringify(object));
};

const oldObject = {
  x: 1,
  y: 1,
  location: {
    lat: 44.22,
    long: 22.1,
  },
};

const newObject = deepCopy(oldObject);

newObject.location.lat = 22.41;

console.log(oldObject, newObject);
```

5. Rotate array cu un numar de pasi la stanga si unul la dreapta

```
const rotateArr = (arr, left, right) => {
  for (let i = 0; i < left; i++) {
    const firstEl = arr.shift();

    arr.push(firstEl);
  }

  for (let i = 0; i < right; i++) {
    const lastEl = arr.pop();

    arr.splice(0, 0, lastEl);
  }

  return arr;
};

const arr = [1, 2, 3, 0, -1, -3];
const newArr = rotateArr(arr, 3, 6);

console.log(newArr);
```

6. Functie care traverseaza un arbore cu mai multe obiecte nestate si returneaza nodul cu id-ul specificat

```
const searchTreeById = (tree, id) => {
  if (tree.id == id) return tree;

  if (tree.children != null) {
    let result = null;
    for (let i = 0; result == null && i < tree.children.length; i++)
{
      result = searchTreeById(tree.children[i], id);
    }

    return result;
  }

  return null;
};

const tree = {
  id: 1,
  name: 'first branch',
  children: [
    { id: 2, children: [] },
    { id: 3, children: [] },
  ],
};

const searchedNode = searchTreeById(tree, 3);

console.log(searchedNode);
```

7. Return true daca fieldurile a doua obiecte sunt diferite

```
const checkObjectsDifferent = (object1, object2) => {
  return JSON.stringify(object1) != JSON.stringify(object2);
};

const x = { x: 1, arr: [1, 3, [1, 3]] };
const y = { x: 1, arr: [1, 3, [2, 3]] };

const objectsAreDifferent = checkObjectsDifferent(x, y);

console.log(objectsAreDifferent);
```

8. Event loop-ul are mai multe faze/etape. setImmediate se apeleaza doar in checkHandlers, pe cand nextTick se apeleaza la inceputul loopului si imediat dupa fiecare faza/etapa. Daca ambele sunt apelate in acelasi timp, nextTick are prioritate fata de setImmediate si va fi apelata prima.

9. La fel ca la 1 (a se vedea proiectul de pe git)

10. In primul rand structura fisierelor (experienta personala). De cele mai multe ori un proiect vechi are o structura fie invechita fie proasta pur si simplu. In al doilea rand componentizarea “la sange” pentru readability, apoi un eventual upgrade la toate modulele (unde este posibil) si intr-un final un refactor la cod pentru o mentenanta mai buna.

11. Pentru performanta ma uit la scorul de performanta de la Google Lighthouse, iar pentru securitate vorbim de best practices as much as possible, de exemplu tokenuri pe autentificare, fisiere cu continut sensibil care nu au ce cauta pe git (de exemplu un .env care poate contine API KEYs), date salvate eronat (posibil in localStorage (apropos de pinia cu persist-state)) si nu in ultimul rand la codul scris. Pot fi probleme pe query-urile catre baza de date (de asta prefer un query builder precum knexjs).