NODE.JS FULLSTACK FEJLESZTÉS

7. forduló

A kategória támogatója: Emarsys

Ismertető a feladathoz

Tesztelés

Felhasznált idő: 00:00/15:00 **Elért pontszám:** 0/12

1. feladat 0/2 pont

TDD során fontos, hogy az alábbi sorrendet betartsuk:

Válasz

- ○ 1. Implementáljuk az elvárt működés egy kis részét.
  2. Írjunk tesztet az új kódra.
  3. Rontsuk el az implementációt, hogy a teszt hibát mutasson.
  4. Refaktoráljuk a teszthez tartozó kód részeket.
- ● 1. Írjunk egy tesztet, ami hibát mutat.
  2. Írjuk meg rá az implementációt, hogy átmenjen a teszt.
  3. Refaktoráljuk a teszthez tartozó kód részeket.

  Ez a válasz helyes, de nem jelölted meg.
- ○ 1. Írjuk meg előre az összes tesztet a specifikáció alapján.
  2. Írjuk meg a tesztek implementációit tetszőleges sorrendben.
  3. Írjuk meg a kódhoz tartozó dokumentációt.
- ○ 1. Írjunk teszteket, amik hibát muatatnak.
  2. Írjuk meg az implementációt, hogy átmenjenek a tesztek.
  3. Refaktoráljuk a kódot.

Magyarázat

https://en.wikipedia.org/wiki/Test-driven_development

2. feladat 0/4 pont

Hogyan változtassuk meg az alábbi kód részletet, hogy a hozzá tartozó tesztek átmenjenek?

Implementáció:

```
function getData(groups, items) {
  return [
    ...(Object.entries(groups)
      .reduce((result, [groupId, groupItems]) => {
        if (groupItems.filter(item => items.includes(item)).length > 0)
          result.push(groupId);



        return result;
      }, [])),
    ...(items.reduce((result, item) => {
      if (Object.values(groups).flat().includes(item)) {
        result.push(item);
      }
      return result;
    }, []))
  ];
}
```

Tesztek:

```
it('result contains each group with at least one of the provided items', () => {
  const groups = {
    t1_g1: ['t1_gi1', 't1_gi2'],
    t1_g2: ['t1_gi3'],
    t1_g3: ['t1_gi9']
  };
  const items = ['t1_ui1', 't1_ui2', 't1_gi1', 't1_gi2', 't1_gi3'];

  expect(getData(groups, items)).to.containSubset(['t1_g1', 't1_g2']);
});

it('result contains each ungrouped item', () => {
  const groups = {
    t2_g1: ['t2_gi1', 't2_gi2'],
    t2_g2: ['t2_gi3'],
    t2_g3: ['t2_gi9']
  };
  const items = ['t2_ui1', 't2_ui2', 't2_gi1', 't2_gi2', 't2_gi3'];

  expect(getData(groups, items)).to.containSubset(['t2_ui1', 't2_ui2']);
});

it('result does not contain any grouped item', () => {
  const groups = {
    t3_g1: ['t3_gi1', 't3_gi2'],
    t3_g2: ['t3_gi3'],
    t3_g3: ['t3_gi9']
  };

  const items = ['t3_ui1', 't3_ui2', 't3_gi1', 't3_gi2', 't3_gi3'];

  expect(getData(groups, items)).not.to.containSubset(['t3_gi1', 't3_gi2', 't3_gi3']);
});
```

Válasz

Az alábbi feltételt negálni kell:

```
if (groupItems.filter(item => items.includes(item)).length > 0)
```

Az alábbi feltételt negálni kell:

```
if (Object.values(groups).flat().includes(item))
```

Ez a válasz helyes, de nem jelölted meg.

A jelenlegi implementáció jól működik

Az alábbi sort kell cserélni:

```
result.push(item);
```

erre:

```
return item;
```

Magyarázat

---

3. feladat 0/6 pont

Mely állítások **igazak** a következő kódrészletre?

```
import { test}  from 'node:test'import * as assert from "assert";function success() { assert.strictEqual(1, 1

test('test 1', async t => {
  success();  await t.test('test 1.1', success)

  await t.test('test 1.2', async t => {
    await t.test('test 1.2.1', success)
    success()
  })

  await t.test('test 1.3', success)
})

test('test 2', t => {
  t.skip()
  success()
})

test('test 3', t => {
  t.test('test 3.1', success)
  t.test('test 3.2', success)
  success()
})
```

Válasz

- **test 1** és altesztjei a következő sorrendben írják a TAP-re az OK üzeneteket, mert először mindig a legmélyebb altesztek végeznek:

    1. test 1.2.1
    2. test 1.1
    3. test 1.2
    4. test 1

- **test 3** nem megy át, mert test 3.1 és test 3.2 nincs bevárva.
  Ez a válasz helyes, de nem jelölted meg.

- **test 2** hibát dob, mert a testContext.skip() nem állítja meg a futtatást.

Magyarázat

Kedves Versenyzők!

A"test 2 hibát dob, mert a testContext.skip() nem állítja meg a futtatást." a korábban jelzettel ellentétben nem igaz, ugyanis ha lefuttatjuk a kódrészt, akkor nem dob hibát, hiszen a success-ben lévő assert.strictEqual(1, 1) nem dob hibát. Az igaz, hogy a skip nem állítja meg a futást, de így egyben nem, mert a tagmondat első fele hamis.

test 1 *és altesztjei a következő sorrendben írják a TAP-re az OK üzeneteket, mert először mindig a legmélyebb altesztek végeznek:* HAMIS, mert a subtest-eken sorba megy a teszt futtató, viszont egy felsőbb context nem sikeres, amíg az altesztjei nem végeztek

test 2 *hibát dob, mert a testContext.skip() nem állítja meg a futtatást:* IGAZ. Ahhoz, hogy egy teszt teljesen kimaradjon, meg kell adni a {skip: true} opciót

test 3 *nem megy át, mert test 3.1 és test 3.2 nincs bevárva:* IGAZ