TESZTAUTOMATIZÁLÁS

4. forduló

<epam>

A kategória támogatója: EPAM

Ismertető a feladathoz

A 4. forduló után elérhetőek lesznek a helyezések %-os formában: azaz kiderül, hogy a kategóriában a versenyzők TOP 20% - 40% -60% -ához tartozol-e!

Szeretnénk rá felhívni figyelmedet, hogy a játék nem Forma-1-es verseny! Ha a gyorsaságod miatt kilököd a rendesen haladó versenyzőket, kizárást vonhat maga után!

Felhasznált idő: 00:00/30:00 **Elért pontszám:** 0/65

1. feladat 0/5 pont

Melyik válasz **nem** HTTP metódus?

Válasz

- ○ HEAD
- ○ GET
- ○ DELETE
- ◉ REMOVE
  Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A REMOVE nem HTTP metódus, így ez a helyes válasz.

A többi felsorolt válaszlehetőség mind HTTP metódus a következő jelentéssel:

HEAD: ugyanazt adja vissza, mint a GET, de az üzenettestet kihagyja a válaszból.

GET: a megadott erőforrás lekérését kezdeményezi.

DELETE: törli a megadott erőforrást.

2. feladat 0/10 pont

Egy funkció fejlesztése során a következő fejlesztési fázisok közül melyiknél **legkevésbé célszerű** automata teszteket írni?

Válasz

- ○ Miután a funkció átadásra került
- ● A funkció követelményeinek tisztázása előtt
  Ez a válasz helyes, de nem jelölted meg.
- ○ Miközben a funkció fejlesztés alatt van
- ○ Mielőtt a funkció fejlesztése elkezdődik, de a követeményeket ismerjük

Magyarázat

Helyes válasz a funkció követelményeinek tisztázása előtt, mivel az automata tesztek megírásához pontosan ismernünk kell az adott funkcióra vonatkozó követelményeket, függetlenül a funkció fejlesztésének állapotától.

3. feladat 0/10 pont

Egy weblap fejlesztő cég automata tesztelői szeretnék kinyerni a weboldalhoz tartozó dokumentum objektum modellből a hivatkozásokat és a hozzájuk tartozó weboldalon látható szövegeket.

Az alábbi formátumokban fordulnak elő hivatkozások a weboldalon:

1. Weboldal elso lapja

2. <li id="alszekcio">Tortenet

A következő reguláris kifejezések közül melyikre fogja visszadni nekünk a **match()** funkció az első komplett találatot, ami tartalmazza a teljes "a" elemet, a hivatkozást és hivatkozás szövegét külön kapcsolódó elkapó csoportokként?

Válasz

- ○
```
const regex = new RegExp("<a href=\".*\"></a>")
```
- ●
```
const regex = new RegExp("<a href=\"(.+?)\".*?>([\\w\\s\\/,\\.]*?)<\\/a>");
```
Ez a válasz helyes, de nem jelölted meg.
- ○
```
const regex = new RegExp("<a href=\"(.+?)\".*?>([\\w\\s\\/,\\.]*?)<\\/a>", "g");
```
- ○
```
const regex = new RegExp("<a href=\".+<\\/a>");
```

Magyarázat

const regex = new RegExp("<a href=\".*"<\\/a>") reguláris kifejezés csak azokra a sorokra illeszkedik, amelyek nem tartalmaznak szöveget a linkhez.

const regex = new RegExp("<a href=\"(.+?)\".*?>([\\w\\s\\/,\\.]*?)<\\/a>") reguláris kifejezés illeszkedik megfelelően a linkeket tartalmazó sorokra, visszaadja az első teljes találatot és az elfogó csoportokat is, melyek a hivatkozás és a hozzá tartozó szöveg.

const regex = new RegExp("<a href=\"(.+?)\".*?>([\\w\\s\\/,\\.]*?)<\\/a>", "g") reguláris kifejezés jó lenne, de tartalmazza a "g" zászlót, ami miatt a match funkció visszaadja az összes teljes találatot elfogó csoportok nélkül, ezért ez a válasz helytelen.

const regex = new RegExp("<a href=\".+<\\/a>") reguláris kifejezés visszaadja a teljes "a" osztályt, de nem definiál benne elfogó csoportokat, ezért ez a válasz helytelen.

4. feladat 0/10 pont

Vegyük az alábbi JS kódrészletet:

```
const engineers = [
    {
        name: "John Doe",
        title: "Software Testing Engineer",
        age: 24
    },
    {
        name: "Jane Doe",
        title: "Software Engineer",
        age: 30
    },
    //... more items
]
```

Az engineers tömbből szerezzük meg azoknak a mérnököknek a nevét, akik 35 évesek vagy fiatalabbak, és Szoftverfejlesztőként (Software Engineer) dolgoznak.

A megadott kódrészletek közül melyikkel tudjuk ezt megcsinálni?

Válasz

- [x] `engineers.filter(e => e.title === "Software Engineer").filter(e => e.age <= 35).map(e => e.name)`

Ez a válasz helyes, de nem jelölted meg.

- [ ] `engineers.filter(e => e.title === "Software Engineer").filter(e => e.age < 35).map(e => e.name)`
- [ ] `engineers.map(e => e.title === "Software Engineer").map(e => e.age <= 35).filter(e => e.name)`
- [ ] `engineers.filter(e => e.title === "Software Engineer").filter(e => e.age <= 35).filter(e => e.name)`

Magyarázat

A helyes válasz azért helyes, mert:

1. Az első filter leszűri a mérnököket a titulusuk alapján.
2. A második filter már inputként az első filter eredményét kapja, így kiválasztja azokat a Szoftverfejlesztőket, akik 35 évesek vagy fiatalabbak.

3. A map inputja pedig az előző filter lesz, itt pedig az objektumokból kinyerjük csak a neveket, ahogy a feladat kérte.

5. feladat 0/15 pont

Adottak az alábbi Postman tesztek.

Melyik lehet a helyes script, ha azt szeretném tesztelni, hogy az adott request válaszideje kevesebb mint 200ms?

Válasz

- [x]
```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).to.be.below(200);
});
```
Ez a válasz helyes, de nem jelölted meg.

- [ ]
```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).not.to.be.below(200);
});
```

- [ ]
```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime <= 200).to.be.true;
});
```

- [ ]
```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).to.be.within(200);
});
```

Magyarázat

A helyes válasz:

```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).to.be.below(200);
});
```

Helytelen válaszok:

```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).not.to.be.below(200);
});
```

Ebben az esetben, azt várjuk el, hogy a válaszidő ne legyen kevesebb, mint 200.

```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime <= 200).to.be.true;
```

```
});
```

Ebben az esetben azt várjuk el, hogy a válaszidő legyen 200 vagy kevesebb. Ebben az esetben a 200 is elfogadott így nem teljesíti a feladatban megadott követelményeket.

```
pm.test("Response time is less than 200ms", function () {
        pm.expect(pm.response.responseTime).to.be.within(200);
});
```

A within assertion legalább 2 paramétert vár. Itt található dokumentáció a működéséről: https://www.chaijs.com/api/bdd/#method_within

6. feladat 0/15 pont

Jelöld meg az **igaz** állításokat!

Válaszok

- [x] A Github Actions workflow file YAMLben van írva, a Jenkinsfile pedig Groovyban.
  Ez a válasz helyes, de nem jelölted meg.
- [ ] A Github Actions infrastruktúráját teljes egészében annak a cégnek kell menedzselnie, aki használja.
- [x] A CI/CD alkalmazása nem csökkenti a használt alkalmazás környezetek számát.
  Ez a válasz helyes, de nem jelölted meg.
- [ ] Jenkinsben 3-féle pipeline létezik: declarative, imperative, scripted.

Magyarázat

A Github Actions infrastruktúrájáért a Github felel, lehet hozzákapcsolni saját runnert, abban az esetben a runnereket nekünk kell menedzselni, de a központi szervert a Github saját maga menedzseli.

A Jenkinsben 2 féle pipeline létezik: declarative és scripted. Imperative pipeline nem létezik.

A CI/CD nem ad garanciát arra, hogy a használt környezetek száma csökkenni fog.