

TESZTAUTOMATIZÁLÁS

1. forduló



A kategória támogatója: EPAM

Ismertető a feladathoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

- MINDEN kérdésre **van helyes válasz**.
- Olyan kérdés **NINCS**, amire az összes válasz helyes, ha mégis az összes választ bejelölöd, arra a feladatra automatikusan 0 pont jár.
- A **radio button-os** kérdésekre **egy helyes válasz van**.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Azokat a feladatlapokat, amelyekhez **csatolmány** tartozik, javasoljuk **NEM mobilon** elindítani, erre az érintett feladatlapok előtt külön felhívjuk a figyelmet.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Helyezéseket a 4. forduló után mutatunk**, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.
- **Badge-ket** szintén a 4.forduló után kapsz majd először.
- Ha egyszerre több böngészőből, több ablakban vagy több eszközről megnyitod ugyanazt a feladatlapot, **nem tudjuk vállalni** az adatmentéssel kapcsolatban esetlegesen felmerülő anomáliákért a felelősséget!
- A hét forduló során az egyes kategóriákban (de nem feltétlenül mindegyikben) **könnyű-közepes-nehéz kérdésekkel** egyaránt találkozhatasz majd.

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/30:00

Elért pontszám: 0/65

1. feladat 0/5 pont

Milyen esetben beszélhetünk fals pozitív és fals negatív tesztesetekről?

Válasz

- ☒ A "fals pozitív" teszt hibát jelez olyan helyen, ahol valójában nincs.
A "fals negatív" teszt nem jelez hibát olyan helyen, ahol valójában van.
Ez a válasz helyes, de nem jelölted meg.
- ☐ A "fals negatív" teszt hibát jelez olyan helyen, ahol valójában nincs.
A "fals pozitív" teszt nem jelez hibát olyan helyen, ahol valójában van.
- ☐ A "fals pozitív" teszt hibát jelez olyan helyen, ahol valójában nincs.

A "fals negatív" teszt megtalált egy létező hibát.

☐ Nem létezik olyan hogy fals negatív és fals pozitív teszt

Magyarázat

Azért "fals pozitív", mert a teszteset elbukik, amikor valójában át kellett volna mennie hiba nélkül, és "negatív" eredményt kellett volna kapnod.

Azért "fals negatív", mert a teszt eredmény hibásan lett "negatív", valójában pozitív eredményt kellett volna kapnod.

2. feladat 0/10 pont

Adott a következő DOM struktúra. Mely selectorokkal tudjuk kiválasztani kizárólag a \$260 szövegű price elemet?

```
<products>
  <vendor>
    <vendor_name>The Chair Company</vendor_name>
    <product>
      <product_id>ID134</product_id>
      <short_desc>Queen Anne Chair</short_desc>
      <price pricetype="discounted cost">$85</price>
      <price pricetype="cost">$185</price>
    </product>
    <product>
      <product_id>ID818</product_id>
      <short_desc>Rockstar Chair</short_desc>
      <price pricetype="cost">$160</price>
    </product>
    <product>
      <product_id>ID718</product_id>
      <short_desc>King Chair</short_desc>
      <price pricetype="discounted cost">$160</price>
      <price pricetype="cost">$260</price>
    </product>
  </vendor>
</products>
```

Válaszok

☒ //product[last()]/price[@pricetype='cost']
Ez a válasz helyes, de nem jelölted meg.

☒ //price[text()=' \$260']
Ez a válasz helyes, de nem jelölted meg.

☐ price:last-child

☒ product:last-child price:last-child
Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A //product[last()]/price[@pricetype='cost'] selector helyes, először kiválasztásra kerül az utolsó product majd a hozzá tartozó cost típusú pricetypehoz tartozó price, azaz a \$260.

A //price[text()=' \$260'] selector helyes, ez a \$260 szöveggel rendelkező price elemet határozza meg.

A price:last-child selector helytelen, mert ez productionként az összes utolsó price értéket adja vissza, amibe bele tartozik ugyan a \$260 értékű elem is, de nem csak az.

A product:last-child price:last-child selector helyes, először kiválasztásra kerül az utolsó product, majd a hozzá tartozó utolsó price.

3. feladat 0/10 pont

Melyik eszköz alkalmas arra, hogy önmagában, a böngészőn kívül futtasson JavaScriptet?

Válaszok

☒ Deno
Ez a válasz helyes, de nem jelölted meg.

☒ Node.js
Ez a válasz helyes, de nem jelölted meg.

☐ React.js

Magyarázat

Kedves Versenyzők!

A JavaScriptCode válaszlehetőséget a visszajelzések alapján töröltük, mivel a kérdés értelmezésének függvényében igaz is lehet.

A JavaScriptCore valóban használható önmagában, viszont teljeskörű funkcionalitást nem fog szolgáltatni. A NodeJS engineje a V8. Visszont a JSCore is alkalmazva van hasonlóan, itt már a Bun a teljes runtime: <https://bun.sh>

Így értelemszerűen használható a JSCore vagy V8 is böngészőn kívüli JS futtatásra, viszont egy teljes keretrendszert kell köré építeni, hogy tényleges alkalmazást lehessen fejleszteni böngésző nélkül.

Így erre a legteljesebb válasz a NodeJS és Deno, őket Runtimenak hívjuk, mivel adnak APIkat arra hogy JS Based szoftvert tudjunk fejleszteni.

A JSCore egy Engine, amely nem biztosít environmentet és runtimeot applikációk futtatására és írására. JSCore esetén a Browsert a Safari jelenti, a Server environmentet pedig a Bun, Mobile esetén pedig ReactNative.

Köszönjük megértéseteket!

A Node.js egy már régóta használt JavaScript futtatási környezet, amit böngészőn kívüli JavaScript futtatáshoz használunk. A Deno egy modern JS/TS futtatási környezet, amit a Node eredeti készítője fejleszt, ugyanazt a célt szolgálja, mint a Node. A React.js egy frontend könyvtár, nem alkalmas böngészőn kívüli JS futtatásra. JavaScriptCore ugyan egy JS Motor, viszont önmagában nem lehet vele JS-t futtatni, ehhez runtime-ot, pl. a Bunt kell használnunk.

4. feladat 0/10 pont

A válaszokban HTTP osztályok és státusz kódok csoportosítása látható.

Válaszd ki azt, amelyiknél az osztályok neve és a státuszkódok megfelelő párosításban vannak!

Válasz

☒ Tájékoztató információk (100–199)

Sikeres kérés (200–299)

Átírányítás (300–399)

Klienshiba (400–499)

Szerverhiba (500–599)

Ez a válasz helyes, de nem jelölted meg.

- ☐ Tájékoztató információk (200–299)
 - Sikeres kérés (100–199)
 - Átirányítás (300–399)
 - Klienshiba (400–499)
 - Szerverhiba (500–599)
- ☐ Tájékoztató információk (100–199)
 - Sikeres kérés (200–299)
 - Átirányítás (300–399)
 - Klienshiba (400–499)
 - Szerverhiba (500–599)
 - Nem szabványos (600-699)
- ☐ Tájékoztató információk (100–199)
 - Sikeres kérés (200–299)
 - Átirányítás (300–399)
 - Szerverhiba (400–499)
 - Klienshiba (500–599)

Magyarázat

Az állapotkódok 3 számjegyből állnak, az első számjegy utal a tartalmukra, ez a számjegy 1-től 5-ig terjedhet. Ez alapján a következő csoportjai vannak az állapotkódoknak:

100-199: tájékoztató információk: A kérést megkapta a szerver, feldolgozás következik

200-299: sikeres kérés: A kérést sikeresen megkapta, elfogadta, megértette a szerver.

300-399: átirányítás: További tevékenységekre van szükség a kérés befejezéséhez.

400-499: klienshiba: A kérés rossz szintaxisú vagy nem teljesíthető.

500-599: szerverhiba: A szervernek nem sikerült egy helyes kérést végrehajtania.

5. feladat 0/15 pont

Melyik állítás **NEM** a Continuous Integration (Folyamatos integráció) alapelve?

Válasz

- ☐ A build legyen annyira gyors, amennyire csak lehetséges.
- ☐ Ha lehetséges, teszteljünk a production környezet egy klónjában.
- ☒ Konténerizáljuk minden komponensét a szoftvernek.
Ez a válasz helyes, de nem jelölted meg.
- ☐ A build önmagát kell hogy tesztelje.

Magyarázat

A CI alapjaihoz tartozik az, hogy a buildnek gyorsnak kell lennie, ugyanis többször is buildelünk naponta. A buildnek önmagát kell tesztelnie, azaz automata tesztek szükségesek, és ha lehetséges a production környezet egy klónjában teszteljünk. A konténerizáció ugyan egy devops praktika, de nem tartozik a CI alapjaihoz, csak kiegészíti azt.

6. feladat 0/15 pont

Az alábbiak közül melyik wait metódus támogatott Selenium keretrendszerben?

Válasz



```
new WebDriverWait(driver, Duration.ofSeconds(3)).until(driver -> driver.findElement(By.name("q")));
```

Ez a válasz helyes, de nem jelölted meg.



```
new WebDriverWait(driver, 10L).until(Conditions.elementToBeClickable(By.xpath("//a/h3")));
```



```
new WebDriverWait(driver, Duration.ofSeconds(3)).implicitlyWait(Duration.ofSeconds(10));
```



```
Thread.sleep(Duration.ofSeconds(3));
```

Magyarázat

A `Thread.sleep` a Java api része és egy `long` típusú értéket vár paraméterül nem pedig `Duration` objektumot. Lehetőség szerint ehelyett használjuk a feltételes `wait`-et (explicit wait), mert ebben az esetben mindig vár a `Thread` az idő lejártáig, viszont explicit `wait` esetén hamarabb is folytathatja futását.

A `WebDriverWait` osztályban nincs `implicitlyWait` metódus definiálva így az helytelen, az `implicitlyWait` helyes használata a következő: `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));`

A `Conditions` osztály nem létezik Seleniumban. Helyesen az osztály neve `ExpectedConditions`.



[Legfontosabb tudnivalók](#)

[Kapcsolat](#)

[Versenyszabályzat](#)

[Adatvédelem](#)

© 2023 Human Priority Kft.

KÉSZÍTETTE cone

Megjelenés

Világos