







# WEBFEJLESZTÉS HAGYOMÁNYOS ESZKÖZÖKKEL





A kategória támogatója: Training360 Kft.

#### Ismertető a feladathoz

Iszogatod a cégnél a kávéd, van még majd 30 perced kezdésig, mikor kedvesen mosolyogva eléd raknak egy tesztet. Jelentkeztél, hogy feljebb léphess a tapasztalati (fizetési) ranglétrán, és bizonyítanod kell. Több fordulóban. 25 kérdés. 25 perc elég lesz? Aztán stand up. Így megy ez. No comment.

Felhasznált idő: 00:00/25:00

Elért pontszám: 0/25

# 1. feladat 0/1 pont

Válaszd ki a safe HTTP metódust! (RFC szerint)

#### Válasz



GET

Ez a válasz helyes, de nem jelölted meg.

- POST
- PUT
- PATCH
- DELETE

#### Magyarázat

Akkor *safe* a metódus, ha nem változtatja meg a szerver állapotát. Ez pedig egyedül a *GET*-re igaz, hiszen a többi az erőforrásokat módosítja.

<u>Hypertext Transfer Protocol -- HTTP/1.1</u>

# 2. feladat 0/1 pont

Válaszd ki az idempotens HTTP metódusokat! (RFC szerint)

Válaszok

~	GET Ez a válasz helyes, de nem jelölted meg.
	POST
<u>~</u>	PUT Ez a válasz helyes, de nem jelölted meg.
	PATCH
<b>✓</b>	DELETE  Ez a válasz helyes, de nem jelölted meg.
M	agyarázat
٧	depotens egy művelet, ha többször is végrehajtva ugyanazt az eredményt kapjuk. Azaz jelen esetben mindegy, hogy egyszer vagy N-szer (N > 0) kerül elküldésre a kérés. Ennek a megvalósítására nekünk is figyelnünk kell, mert ha szerveroldalról egy DELETE kérésre egyszer 200-at, aztán 404-et kapunk, akkor nem idempotent.

# 3. feladat 0/1 pont

<u>Hypertext Transfer Protocol -- HTTP/1.1</u>

Válaszd ki, melyik állítások igazak a HTTP GET metódusra!

#### Válaszok

<b>∠</b>	Cache-elhető a válasz
	Es a válacs balvac, da nam ialäl

Ez a válasz helyes, de nem jelölted meg.

ŀ	4	Könyvje	zőbe	elment	hető
---	---	---------	------	--------	------

Ez a válasz helyes, de nem jelölted meg.

Böngészőben küldés után, az oldal frissítésekor újra el lesznek küldve az adatok

Kevésbé biztonságos, mint a POST

Ez a válasz helyes, de nem jelölted meg.

# Magyarázat

- A GET metódus erőforrást kér el, ezért nagyon egyszerűen cache-elhető
- Mivel az URL-hez fűzi hozzá az adatokat, ezért az URL-t könyvjelzőbe elmenthetjük
- A POST-al ellentétben nem fogja újra elküldeni a kérést egy refresh után
- A nem ASCII karakterek ki lesznek cserélve egy %HEXA formátumú kódra
- Mivel az URL-ben van az összes adat, a jelszót is beleértve, ezért kevésbé biztonságos, mint a POST

#### Kedves Versenyzők!

A "Csak ASCII karaktereket támogatottak" válaszlehetőséget töröltük, ugyanis nem egyértelmű annak helyessége, vagy helytelensége (a mostani RFC szabvány szerint lehetőség van GET metódus esetében is body-t küldeni).

# 4. feladat 0/1 pont

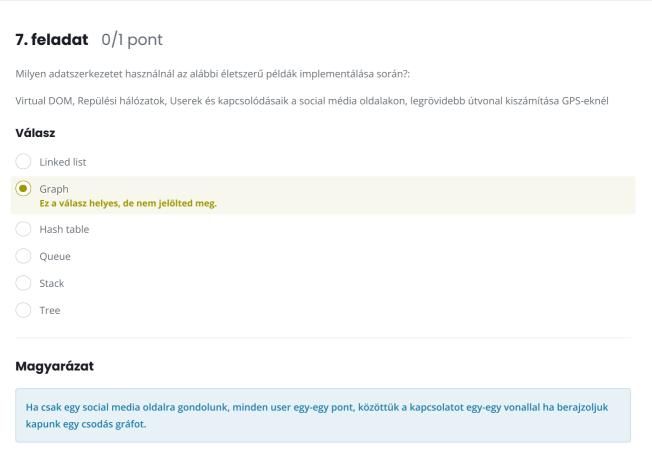
Milyen adatszerkezetet használnál az alábbi életszerű példák implementálása során?:

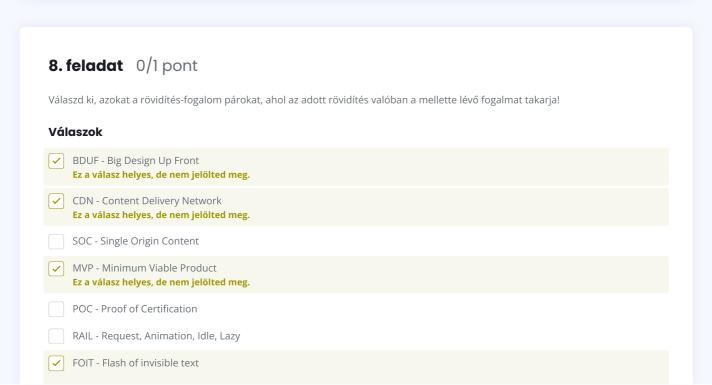
Mozgólépcső, Spotify (vagy egyéb lejátszó) next-back gombok, Windows-os Alt + Tab

	Linked list  Ez a válasz helyes, de nem jelölted meg.
	Graph
	Hash Tables
	Queue
	Stack
	Tree
Ma	gyarázat
az	gy Linked list (Láncolt lista) elemek sorozatából épül fel, és minden tartalmaz egy hivatkozást a következő elemre. Amennyiben z utolsó elem tartalmazza a hivatkozást az első elemre, úgy megvalósul a körkörösség, Tehát a lejátszó esetében, vagy az Alt + ab nyomkodásánál az utolsó elemet elérve újra az elsőre ugrunk.
5 (	feladat 0/1 pont
	en adatszerkezetet használnál az alábbi életszerű példák implementálása során?:
	wser history, log, undo/redo, emailek
Vál	asz
	Linked list
	Graph
	Hash Tables
	Queue
	Stack Ez a válasz helyes, de nem jelölted meg.
	Tree
Ma	gyarázat
	Stack (Verem) egy LIFO (Last In First Out) adatszerkezet. A legkésőbbi elemhez férünk hozzá először. Jönnek az újabb és újabb nailek, belekerülnek a Verembe, mindig a legújabb van elöl.
<b>6.</b> 1	feladat 0/1 pont
Mily	en adatszerkezetet használnál az alábbi életszerű példák implementálása során?:
HTM	IL DOM, Domain Name Server, Fájlrendszer, IP routing tábla
Vál	αsz
	Linked list
	Graph

Hash table

Queue	
Stack	
Tree Ez a válasz he	elyes, de nem jelölted meg.
Magyarázat	
	Tree-re gondolunk, az elemek fa struktúrában helyezkednek el, Van egy html gyökér elem, felállítható a szülő- olat, vannak szomszédos elemek, stb.
7. feladat	0/1 pont





Ez a válasz helyes, o	le nem jelölted meg.		
FOUT - Flash of unl	FOUT - Flash of unloaded text		
Magyarázat			
A rövidítések helyes n	negoldásai a következők:		
BDUF - Big Desig	n Up Front		
• CDN - Content D	elivery Network		
SOC - Separation	Of Concerns		
MVP - Minimum	Viable Product		
• POC - Proof Of Co	oncept		
• RAIL - Response,	Animation, Idle, Load		
• FOIT - Flash of in	visible text		

# 9. feladat 0/1 pont

• FOUT - Flash of unstyled text

Adott az alábbi HTML és JS kód:

Saját gépen egy .jpeg kiterjesztésű képfájl kiterjesztését átírom pdf-re.

Betallózom ezt a fájlt a fenti alkalmazás segítségével.

Mi lesz a konzolon a kód lefutása után?

#### Válasz

image/jpeg



application/pdf

Ez a válasz helyes, de nem jelölted meg.



image/jfif

# Magyarázat

A *type* nem néz bele a fájlba a MIME típus megállapításához, hanem a fájl kiterjesztése alapján kalkulálja ki azt. Emiatt hiába van egy kép fájlunk, mivel a kiterjesztést módosítottam, ezért *application/pdf* lesz a konzolon. Soha nem elég csak kliensoldalon ellenőrizni a fájlt típusát.

# 10. feladat 0/1 pont Az alábbiak közül melyik operator, illetve statement használható JavaScriptben? Válaszok ✓ with statement Ez a válasz helyes, de nem jelölted meg. ✓ label statement Ez a válasz helyes, de nem jelölted meg. goto statement in operator Ez a válasz helyes, de nem jelölted meg. of operator void operator Ez a válasz helyes, de nem jelölted meg. Magyarázat A goto statement nem elérhető JavaScriptben, de a label igen, ahogy a break, és a continue is. Az of operátor önállóan nem létezik, de for of ciklus igen. A többi operator és statementek létezik JavaScriptben, de ilyen-olyan okokból kifolyólag nem használjuk. with label in void

# 11. feladat 0/1 pont Válaszd ki azokat az állításokat, melyek igazak az Observer tervezési mintára! Válaszok ✓ Az Observer esetében a megfigyelők és a megfigyeltek tudnak egymásról Ez a válasz helyes, de nem jelölted meg. ✓ Az Observer általában szinkron módon van megvalósítva Ez a válasz helyes, de nem jelölted meg. Az Observer egy Creational Design Patterns ✓ Az Observer estében a komponensek szoros kapcsolódnak (tight coupling) Ez a válasz helyes, de nem jelölted meg. ✓ Az Observer egy Gang of Four tervezési minta Ez a válasz helyes, de nem jelölted meg.

# Magyarázat

Az Observer és a PubSub minta hasonlóak, de nem ugyanazok. Természetesen mind a kettő Behavior Design Patterns.

A különbségek röviden összefoglalva:

- Az Observer Gang of Four minta a PubSub nem
- A Observer általában szinkron az PubSub aszinkron van megvalósítva
- Míg az Observer esetében szorosan kapcsolódnak a komponensek a PubSub esetében lazán, egy harmadik félen keresztül történik a kommunikáció

# **12. feladat** 0/1 pont

Válaszd ki azokat az állításokat, melyek igazak a PubSub tervezési mintára!

#### Válaszok

<b>✓</b>	A PubSub minta gyakran használt elosztott rendszereknél a kommunikáció során
	Ez a válasz helves, de nem jelölted meg.





A PubSub estében a komponensek lazán kapcsolódnak (loose coupling) **Ez a válasz helyes, de nem jelölted meg.** 

A PubSub egy Gang of Four tervezési minta

# Magyarázat

Az Observer és a PubSub minta hasonlóak, de nem ugyanazok. Természetesen mind a kettő Behavior Design Patterns.

A különbségek röviden összefoglalva:

- Az Observer Gang of Four minta a PubSub nem
- A Observer általában szinkron az PubSub aszinkron van megvalósítva
- Míg az Observer esetében szorosan kapcsolódnak a komponensek a PubSub esetében lazán, egy harmadik félen keresztül történik a kommunikáció

# 13. feladat 0/1 pont

Mi lesz az alábbi kód lefutása után a konzolon?

```
1 let arr;
2 for (var i = 5; i > 0; i--) {
3   arr.push(i);
4 }
5 console.log(arr);
```

- [1, 2, 3, 4, 5]
- [5, 4, 3, 2, 1]
- Error

Ez a válasz helyes, de nem jelölted meg.

undefined

# Magyarázat

Az arr változónak nem adtunk kezdőértéket, így undefined lesz, és mint ilyen nincs *push()* metódusa, ezért *Error*-t kapunk.

# **14. feladat** 0/1 pont

Mi lesz az alábbi kód lefutása után a konzolon?

```
const fruits = {
  dragonfruit: {
    name: 'dragonfruit',
    price : 10
  }
}

const { dragonfruit: { name: n = 'sárkánygyümölcs' } } = fruit;

console.log(n);
```

#### Válaszok



# 

# Magyarázat

A *sort()* metódus az UTF-16 kódok alapján rendez. Abban pedig az "Á" és az "É" betű az angol karakterek után kullog, tehát "Ádám" és "Éva" marad a tömb végén. Mivel az első elem "Bendegúz" lesz, és destructuringgal az első elemet vesszük ki a tömbből, így a konzolon a "Bendegúz" string fog megjelenni.

# 16. feladat 0/1 pont

```
const numbers = [];
for (var i = 0; i < 4; i += 1); {
   numbers.push(i + 1);
}</pre>
```

[1, 2, 3]

[5]

Ez a válasz helyes, de nem jelölted meg.

[1, 2, 3, 4]

[4]

# Magyarázat

A trükk abban rejlik, hogy a for záró zárójele után volt egy pontosvessző, tehát üres volt a ciklusmag.

Emiatt a *numbers.push()* csak egyszer fut le, mikor már a for ciklus futása végzett.

Mivel az *i* ciklusváltozó a var kulcsszóval lett létrehozva, ezért a cikluson kívül is elérhető, értéke a kilépési feltételnek megfelelő érték, azaz 4. A tömbben így egy érték lesz a 4+1, azaz 5.

Kedves Versenyzők!

A feladatban szándékosan volt a pontosvessző a for loop végén.

Ugyan a console.log() kimaradt, de a választható opciók közül mindegyik egy tömb volt, így egyértelmű, hogy a numbers értékét kellett volna kiíratni.

# **17. feladat** 0/1 pont

Mi lesz az alábbi kód lefutása után a konzolon?

```
const creature = Object.create({
  name: 'Winnie-the-Pooh',
  age: 96
});

delete creature.age;

console.log(creature);
```

#### Válasz



Ez a válasz helyes, de nem jelölted meg.

- { name: Winnie-the-Pooh', age: 96 }
- ( ) nu
- { name: Winnie-the-Pooh' }

# Magyarázat

Az Object.create() egy meglévő objektumot használ fel, mint prototípust. Azaz nem a *creature* hanem a *creature* prototípusa tartalmazza a name és age tulajdonságokat. Mivel a *delete*-el a *creature age* tulajdonságát akarjuk törölni, de közvetlenül nincs neki, így nem is történik semmi. Maga a person pedig továbbra is egy empty object marad.

Természetesen a prototípust lekérhetjük, és látszódnak a tulajdonságok: Object.getPrototypeOf(person)

# 18. feladat 0/1 pont

Mi lesz az alábbi kód lefutása után a konzolon?

```
function* generate() {
  yield 1;
  yield 2;
  return 3;
}

console.log([...generate()]);
```

#### Válasz

() [1]

[3]

[1

Ez a válasz helyes, de nem jelölted meg.

[1, 2, 3]

# Magyarázat

A generate() meghívása egy Generator-t ad vissza.

Mivel ez iterálható object a spread operátor segítségével kibontjuk, és egy tömbbe tesszük az elemeket.

A spread nem a teljes objektumot, hanem csak a value-kat helyezi el a tömbben.

A spread egészen addig megy, a *done* értéke *true* nem lesz, tehát a *return* utáni érték már nem fog belekerülni.

<u>Link</u>

# 19. feladat 0/1 pont

Mi lesz az alábbi kód lefutása után a konzolon?

```
1 const obj = {
  value0f() { return 42 },
  toString() { return 'forty-two'},
  };
  console.log(String(obj) + obj + `${obj}` + (obj));
```

- 42424242
- 4242forty-two42
- 84-twoforty-two42

forty-two42forty-two42

Ez a válasz helyes, de nem jelölted meg.

# Magyarázat

A String constuctor a toString() metódust használja, ugyanúgy, mint a template string is, míg a + a valueOf()-ot használja a háttérben.

#### Tehát:

- String(obj) = forty-two
- (String(obj) + obj = forty-two42
- String(obj) + obj + \${obj} = forty-two42forty-two
- String(obj) + obj + \${obj} + (obj) = forty-two42forty-two42

Link

# 20. feladat 0/1 pont

Melyik függvény fog Error-t dobni (nem fut le a catch ág kódja) ha meghívjuk?

```
async function returnWithAwait () {
     try {
       return await Promise.reject(new Error());
   } catch (error) {
       return 'OK!';
    }
   }
   async function returnWithoutAwait () {
    try {
       return Promise.reject(new Error());
   } catch (error) {
       return 'OK!';
15 }
```

returnWithAwait()

returnWithoutAwait() Ez a válasz helyes, de nem jelölted meg.

Mindkettő

Egyik sem

# Magyarázat

Ez azon kivétel, amikor van különbség a return és a return await között.

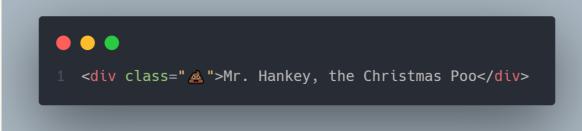
A returnWithAwait() esetében az await miatt megvárja, míg rejected lesz a Promise. Mivel hibát dobott, ezért a catch ágra ugrik. A Promise resolved lesz az 'OK' értékkel. A returnWithoutAwait(), mivel nincs ott az await, nem várja meg a futást, és Error-t dob. Tehát cask akkor ugrik a catch ágra, ha ott az await.

# 21. feladat 0/1 pont

Jelöld be a helyes állításokat!

#### Válaszok





Ez a válasz helyes, de nem jelölted meg.

Használhatok inline stílusmegadásnál css variable-t az alábbi módon:



Ez a válasz helyes, de nem jelölted meg.

A transform: *translate(50%, 50%)* helyett írhatom azt, hogy: *translate: 50% 50%* **Ez a válasz helyes, de nem jelölted meg.** 

A natív CSS *@apply* már széleskörben támogatott a böngészők körében

# Magyarázat

Amennyiben Unicode karakterkódolást használunk (mi mást?) használhatjuk a " a " és egyéb unicode karaktereket akár classname-ben vagy bárhol.

Inline stílusmegadás esetében ugyanúgy elérjük a változókat, mintha beágyazva, vagy belinkelve, és ezek JavaScriptből is lekérdezhetők a styles propertyn keresztül. Pl.: document.querySelector('.timmy').style.getPropertyValue('--textcolor')

A különböző transzformációk külön propertyként történő megadása már a legtöbb böngészőben támogatott, így nem kell mindig megadni a *transform*ot. <u>Translate - caniuse</u>

Az @apply rule sajnos még csak proposalban van: @apply - caniuse

# **22. feladat** 0/1 pont

Az alábbiak közül melyik a legspecifikusabb szelektor?

#### Válasz

nav .mainMenu > a:hover::before

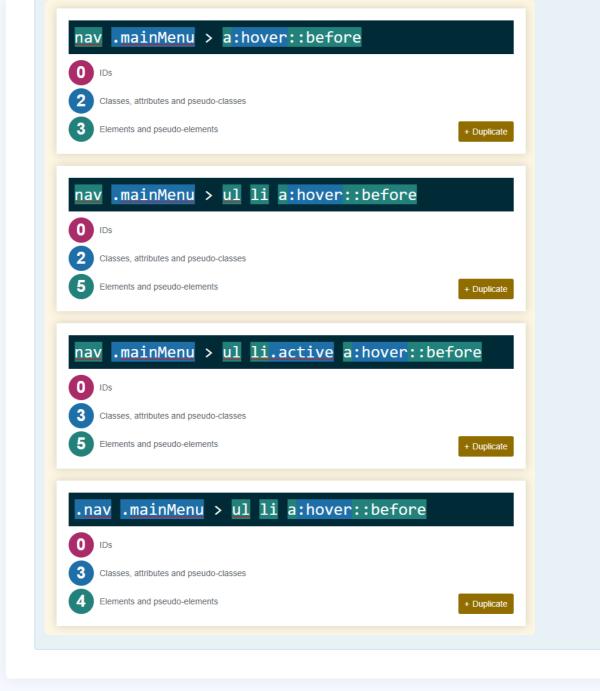
nav .mainMenu > ul li a:hover::before

nav .mainMenu > ul li.active a:hover::before Ez a válasz helyes, de nem jelölted meg.

.nav .mainMenu > ul li a:hover::before

# Magyarázat

CSS specificity kalkulátor alapján:



# **23. feladat** 0/1 pont

Adott a következő html kód:



és 3 CSS kód:

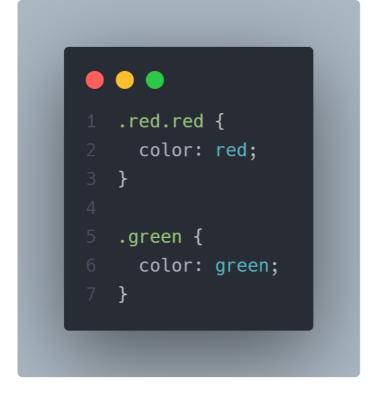
A.

```
1 .green {
2  color: green;
3 }
4
5 .red {
6  color: red;
7 }
```

В

```
1 .red {
2   color: red;
3  }
4
5 .green {
6   color: green;
7  }
```

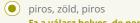
és **C.** 



Milyen színű lesz sorban, **A**, **B**, és **C** kódok esetén a *div*en belül a szöveg?

#### Válasz

piros,	zöld	zöld
pii OJ,	ZOIG,	2010



Ez a válasz helyes, de nem jelölted meg.

- zöld, piros, zöld
- zöld, piros, piros

# Magyarázat

Az **A** és **B** esetben amelyik kód később van a CSS-ben az fog érvényesülni, tehát piros, zöld.

A **C** esetben a specifikusság miatt piros lesz a szövegszín.

# **24. feladat** 0/1 pont

Az alábbiak közül melyik **nem** egy szabvány szerinti media query typus?

# Válasz



projector

Ez a válasz helyes, de nem jelölted meg.

- braille
- aural
- speech

# Magyarázat

A szabványos media queryk:

all

• print
• screen
• tty
• tv
• projection
• handheld
• braille
• embossed
• aural
• speech
Media Queries Level 4

# 25. feladat 0/1 pont Hogy érzed, ennyi kérdés ennyi idő alatt elég volt? Netán sok(k)? Esetleg kevés? Válaszok ✓ Csak ennyi? Kérek még! Ez a válasz helyes, de nem jelölted meg. ✓ Elég volt, köszi. Ez a válasz helyes, de nem jelölted meg. ✓ Ez elég erősre sikerült. Ez a válasz helyes, de nem jelölted meg. ✓ Na elmész a... Ez a válasz helyes, de nem jelölted meg. ✓ "Az ötöst szádra ne vedd..." azaz nem érdekel az ingyen egy pont - sem a gyilkos nyúl - , a véleményem megtartom magamnak. Magyarázat Az most nincs.:)

'I'

Legfontosabb tudnivalók ☑ Kapcsolat ☑ Versenyszabályzat ☑ Adatvédelem ☑
© 2023 Human Priority Kft.

KÉSZÍTETTE C⊜NE

Megjelenés

• Világos ≎