**Javascript alapok 2.**

**4. Operátorok**

1. **Aritmetikai és értékadás**
   * A matches tömb segítségével számold ki, hogy a harcos hány győzelmet szerzett!
   * Tárold el a wins változóban.
2. **Logikai és összehasonlító**
   * Az előző feladat megoldását bővítsd ki úgy, hogy a harcos vereségeit is számolja meg! Írd ki, hogy a harcos győzelmeinek száma nagyobb-e a vereségeinél!
   * Az összehasonlításból állíts elő egy true/false értéket és tárold el a isGoodFighter változóba!
3. **Ternary operator (?)**
   * Adj a fighter.matches tömbhöz még 10 véletlenszerű meccset!
     + használd a Math.random() függvényt,
     + a meccs dátuma legyen 2025.09.01 és 2025.09.30 közötti,
     + a meccs eredménye legyen *win*, *loss* vagy *draw*.
   * Írj ki a konzolra egy üzenetet:

"Bajnok esélyes", ha több mint 10 győzelme van,

"Erős harcos", ha legalább 5 győzelme van,

különben "Edzeni kell még".

**5. Vezérlési szerkezetek**

1. **if...else**  
   Döntsd el a fighter.age alapján, hogy a harcos

junior (<20),

prime (20–30) vagy

veteran (30<),

és írd az eredményt az ageGroup változóba!

1. **switch**  
   Írj egy switch szerkezetet, ami a fenti felosztás szerint harcos kora alapján kiírja a megfelelő korcsoportot (*junior*, *prime* vagy *veteran*)!

**6. Ciklusok**

1. **for/for in/for of ciklus**
   * Írd ki a fighter.matches minden elemét mindhárom for ciklus fajtával! Mit tapasztaltál?
2. **while ciklus**
   * Írd ki a fighter.matches minden elemét while ciklussal!
3. **Extra: az Iterator osztály**
   * Kérdezd le a fighter.matches iterátorát:

const iterator = fighter.matches[Symbol.iterator]();

* + Írd ki a tömb minden elemét az iterator segítségével while ciklusban! Ehhez használd az IteratorResult = Iterator.next() metódust! Az IteratorResult egy objektum done és value mezőkkel.

**7. Függvények, metódusok**

1. **Paraméter, alapérték**  
   Írj egy függvényt addMatch(fighter, date, result), amely hozzáad egy új mérkőzést a meccsek tömbhöz. Ha nincs a paraméter megadva, akkor az értéke legyen *win*! Hozz létre 2 új meccset a függvény segítségével!
2. **Névtelen és nyílfüggvény**
   * Készíts egy névtelen függvényt, ami megszámolja a győzelmeket (function(matches) {...}). A függvény legyen a countWins változó értéke. Hogyan lehet meghívni így a függvényt? Meg lehet-e hívni változóban tárolás nélkül?
   * Írd át nyílfüggvény formájába: (matches) => {...}.
   * Számold meg a győztes meccseket az Array.reduce metódussal!
   * Készíts egy új tömböt a matches-ből az Array.map metódussal, ami a meccsek eredményét (loss, draw, win) a 0, 1, 2 számokra cseréli.

**8. Extra: összetett feladat**

A mellékelt JSON fájlt szúrd be mint *konstans* változót a main.js fájlba!

Implementáld a következő programozási tételeket!

* + firstByName(name): az első egyező nevű harcos
  + firstByAgeGroup(ageGroup): az első egyező korcsoportú harcos
  + getAgeGroups(): harcosok listája korcsoportok szerint, az eredmény legyen egy objektum – {"junior": [], "prime": [], "veteran": []}

Implementáld az általános first(list, predicate) függvényt, aminek az első paramétere egy iterálható objektum (pl. tömb), a második paramétere egy állítást kifejező függvény, azaz predikátum. A predikátum függvény lényegét tekintve teljesen megegyezik a C# Predicate<T> delegate típusával:

bool Predicate(listItem);

A függvény a paraméterként kapott objektumról egy igaz-hamis állítást tesz, pl. listItem.name == "Conor McGregor"