Az alábbiakban gyakorlati feladatokat ajánlok az egyes témákhoz, amelyek segíthetnek a tanulóknak elmélyíteni JavaScript-ismereteiket és fejleszteni gyakorlati készségeiket.

**1. Objektumok és Tömbök Kezelése (ES6)**

* **Feladat 1**: Hozz létre egy tömböt, amely diákok nevét tartalmazza, és használd a .map(), .filter(), és .reduce() függvényeket a következőkre:
  + Módosítsd minden diák nevét, hogy nagybetűs legyen (.map()).
  + Szűrd ki azokat a diákokat, akiknek a neve "A" betűvel kezdődik (.filter()).
  + Számold meg, hogy hány diák van a listában (.reduce()).
* **Feladat 2**: Hozz létre egy objektumot, amely egy könyvet reprezentál (cím, szerző, kiadás éve), majd használj **destructuring**-et, hogy külön változóba mentsd az egyes tulajdonságokat.
* **Feladat 3**: Kombinálj két objektumot a **spread operator** segítségével, hogy egy új objektum jöjjön létre, amely mindkettő tulajdonságait tartalmazza.

**2. Hibakeresés a JavaScript Kódban, Kód Tesztelése**

* **Feladat 1**: Írj egy egyszerű programot, amely két számot ad össze, és szándékosan hagyj benne egy hibát (pl. egy null vagy undefined változó). Használj **try-catch** blokkot a hiba kezelésére.
* **Feladat 2**: Tanítsd meg a tanulókat a **Chrome DevTools** használatára. Adj egy rövid JavaScript kódot, amely hibát tartalmaz, és kérd meg a tanulókat, hogy a böngésző eszközeivel találják meg és javítsák ki a hibát.

**3. Aszinkron Futás és Callback-ek**

* **Feladat 1**: Írj egy programot, amely a **setTimeout()** függvénnyel késleltetve írja ki egyesével a hét napjait a konzolra.
* **Feladat 2**: Készíts egy egyszerű "to-do" alkalmazást, ahol minden hozzáadott feladat után egy **callback** függvény fut le, amely kiírja, hogy "Új feladat hozzáadva: {feladat neve}".

**4. Promise-ok, async/await**

* **Feladat 1**: Írj egy **Promise**-t, amely szimulálja egy API kérés sikeres vagy sikertelen eredményét (pl. resolve() vagy reject()). Adj hozzá .then(), .catch() és .finally() kezelőket.
* **Feladat 2**: Készíts egy **async/await** alapú függvényt, amely egy meghatározott idő után visszatér egy értékkel. Töltsd meg a kódot hibakezelőkkel is (try-catch).
* **Feladat 3**: Készíts egy "háttérfolyamat szimulátort", amely időnként megjeleníti a folyamat állapotát (pl. % completion) await és **setTimeout()** használatával.

**5. JSON Kezelése**

* **Feladat 1**: Írj egy JavaScript programot, amely JSON formátumban tartalmaz egy listát könyvekről (cím, szerző, kiadás éve). Készíts egy függvényt, amely egy adott évhez szűri a könyveket.
* **Feladat 2**: Olvass be JSON adatokat egy fájlból vagy statikus változóból, és alakítsd át egy HTML táblázattá, amely megjeleníti az adatokat.

**6. REST API Architektúra**

* **Feladat 1**: Írj egy JavaScript kódot, amely egy fiktív **REST API**-t használ, hogy könyvek listáját (GET kérés), új könyv hozzáadását (POST kérés), vagy könyv törlését (DELETE kérés) szimulálja.
* **Feladat 2**: Használj egy ingyenes, nyilvános API-t (pl. JSONPlaceholder), és írd meg a kódot, amely kér és megjelenít adatokat egy listából (pl. felhasználók vagy posztok).

**7. AJAX és fetch() függvény**

* **Feladat 1**: Használj **fetch()** függvényt egy nyilvános API hívására, majd jelenítsd meg az adatokat egy egyszerű HTML oldalon. Készíts hibakezelést a fetch sikertelen lefutására is.
* **Feladat 2**: Mutasd be az **AJAX** történeti hátterét az XMLHttpRequest objektum használatával, majd készíts egy egyszerű AJAX kérést, amely egy API adatokat kér le.
* **Feladat 3**: Készíts egy felhasználói űrlapot, amely adatokat kér be, és a beküldés után **POST** kérés segítségével továbbítja az adatokat egy REST API végpontra (fetch használatával).

**Kiegészítő Feladatok**

* **Event Loop megértése**: Készíts egy példát, amely bemutatja az **event loop** működését. Pl. egy setTimeout() és egy szinkron függvény (console.log) kombinálásával figyeljétek meg, hogy mi jelenik meg először.
* **Local Storage használata**: Hozz létre egy egyszerű alkalmazást, amely a felhasználói adatokat menti **Local Storage**-ba, majd később visszaolvassa azokat.

Ezek a gyakorló feladatok segítenek a diákoknak elmélyíteni az alapvető JavaScript technikákat és megérteni az aszinkron programozást, valamint a kliensoldali fejlesztés gyakorlati aspektusait. Ha további részletekre vagy konkrét megoldásokra van szükséged, szívesen segítek!