Az Express.js keretrendszer elsajátítása több lépésen keresztül vezet, amelyek az alapoktól egészen a haladó funkciókig terjednek. Az alábbiakban bemutatom, milyen lépéseket érdemes követni, ha hatékonyan szeretnéd megismerni és használni az Express.js-t:

# Alapok elsajátítása: Node.js és JavaScript

## JavaScript alapok.

## Node.js alapok

# Express telepítése és első alkalmazás

## Express telepítése

Kezdd egy új Node.js projekt létrehozásával, majd telepítsd az Express-t   
npm install express   
paranccsal.

## Első szerver létrehozása:

Készíts egy alapvető Express szervert, ami egy egyszerű "Hello World" üzenetet ad vissza.

# Routing (útvonalkezelés)

* **Routing**: Tanuld meg, hogyan kezelheted a különböző útvonalakat (app.get(), app.post(), stb.) az Express-ben.
* **Dinamikus útvonalak**: Ismerd meg a dinamikus útvonalak használatát, például hogyan lehet paramétereket átadni az URL-ben (/users/:id).

**Gyakorlati feladat:**

* Készíts egy szervert, amely különböző útvonalakon különböző adatokat szolgáltat.

# Middleware használata

* **Middleware koncepciója**: Az Express egyik kulcsa a middleware, amely segít a bejövő kérések feldolgozásában (pl. hitelesítés, logolás, hibakezelés).
* **Beépített és saját middleware**: Használd az Express beépített middleware-jeit, mint a express.json() vagy a express.static(), és tanuld meg saját middleware-t készíteni.

**Gyakorlat:**

* Implementálj egy middleware-t, ami minden kérés előtt naplózza a kérés időpontját.

# Sablonmotorok használata

* **Sablonmotorok bevezetése**: Express támogat több sablonmotort, mint a Pug, EJS, Handlebars. Tanuld meg, hogyan használhatsz sablonmotorokat dinamikus HTML oldalak generálásához.
* **Sablonmotor beállítása és használata**: Készíts egy egyszerű sablont egy adott motorral, és adj vissza dinamikus tartalmat.

**Példa Pug sablonmotorral:**

# REST API készítése

* **RESTful API-k**: Express remekül alkalmas RESTful API-k készítésére. Tanuld meg, hogyan kezelheted a különböző HTTP metódusokat (GET, POST, PUT, DELETE), és hogyan strukturálhatod API-jaidat.
* **Adatkezelés és JSON válaszok**: Készíts egy REST API-t, amely JSON adatokat szolgáltat.

**Gyakorlat:**

* Hozz létre egy egyszerű CRUD (Create, Read, Update, Delete) alkalmazást, amely pl. felhasználók adatait kezeli.

# Adatbázis integráció

* **Adatbázis csatlakoztatása**: Tanuld meg, hogyan integrálhatsz adatbázist (pl. MongoDB, MySQL) az Express alkalmazásodba.
* **ORM használata**: Opcionálisan tanulj meg egy ORM-et (pl. Mongoose vagy Sequelize), hogy könnyebben dolgozhass az adatbázissal.

**Gyakorlat:**

* Hozz létre egy alkalmazást, amely adatokat kér le és tárol egy adatbázisban.

# Hibakezelés és biztonság

* **Hibakezelés**: Tanuld meg, hogyan kezeld a hibákat az Express-ben, mind szinkron, mind aszinkron módon (Promise-ok).
* **Biztonsági intézkedések**: Ismerkedj meg a biztonsági gyakorlattal, pl. CORS kezelése, input validáció, titkosítás.

**Gyakorlat:**

* Implementálj alapvető hibakezelést és biztonsági middleware-t (pl. Helmet).

# Autentikáció és jogosultságkezelés

* **Hitelesítés**: Tanuld meg, hogyan implementálhatsz hitelesítést (pl. JSON Web Token vagy session alapú hitelesítés).
* **Jogosultságok kezelése**: Kezeld a felhasználói jogosultságokat a különböző API végpontokon.

**Gyakorlat:**

* Hozz létre egy bejelentkezési rendszert, ahol felhasználók tokenek segítségével tudnak autentikálni.

# Tesztelés és telepítés

* **Tesztelés**: Tanuld meg, hogyan tesztelheted az Express alkalmazásod automatikusan (pl. Mocha, Chai vagy Jest segítségével).
* **Telepítés**: Ismerkedj meg a telepítési folyamatokkal, pl. hogyan telepíthetsz alkalmazást Heroku-ra vagy más cloud platformra.

**Gyakorlat:**

* Írj teszteket az API végpontjaidhoz, majd telepítsd az alkalmazásodat egy felhőszolgáltatóra.

# További források: