EXPRESS.JS

- 1- إيه هوExpress.js ؟ : إطار عمل (Framework) مبني على Node.js لتطوير تطبيقات ويب و APIبسهولة وسرعة. بيوفر أدوات لإدارة طلباتHTTP ، التوجيه(Routing) ، والMiddleware
- لماذا Express ؟ :
  - خفيف ومرن.
  - يدعم بناء RESTful APIs
  - يستخدمه شركات كبيرة زي MySpace و Storify
- متطلبات :
  - معرفة بـ JavaScript و Node.js
  - تثبيت Node.js و npm (Node Package Manager)

2- إعداد Express.js :

- تثبيت Express

1. افتح Terminal واكتب :

```
npm init -y

npm install express
```

2. إنشاء ملف أساسي مثل app.js

```
const express = require('express');

const app = express();

app.listen(3000, () => console.log('Server running on port 3000'));
```

- تشغيل السيرفر :

```
node app.js
```

افتح المتصفح على http://localhost:3000

3- التوجيه (Routing) :

- إيه هو؟ : تحديد استجابات السيرفر للطلبات على مسارات (URLs) معينة.
- المهم :
  - GET Request

```
app.get('/', (req, res) => {
  res.send('Hello, World!');
});
```

يرجع نص "Hello, World!" للمسار الرئيسي.

  - POST Request

```
app.post('/submit', (req, res) => {
  res.send('Data received!');
});
```

    - معاملات (Parameters)

```
app.get('/user/:id', (req, res) => {
  res.send(`User ID: ${req.params.id}`);
});
```

يظهر ID المستخدم مثل http://localhost:3000/user/123

- نصيحة : جرب إنشاء مسارات مختلفة واختبرها بـ Postman

4- Middleware

- إيه هو؟ : دوال تُنفذ قبل أو أثناء معالجة الطلبات.
- المهم :
  - Middleware مدمج :

```
app.use(express.json()); // لتحليل JSON في الطلبات
```

  - Middleware مخصص :

```
app.use((req, res, next) => {

  console.log(`Request at ${new Date()}`);

  next(); // الانتقال للدالة التالية

});
```

  - أمثلة :
    - express.static : لخدمة ملفات ثابتة مثل HTML ، CSS
    - body-parser : لتحليل بيانات الـ POST
- نصيحة : جرب إضافة Middleware يسجّل وقت كل طلب.

5- إدارة الطلبات والاستجابات :

- Request (req) : يحتوي على بيانات الطلب مثل Headers ، Parameters، Body
  - مثال req.body : للحصول على بيانات الـ POST
- Response (res) : يحدد الاستجابة مثل نص، JSON، أو صفحة HTML
  - مثال :

```
app.get('/api', (req, res) => {

  res.json({ message: 'API response' });

});
```

- Status Codes :
  - 200 : نجاح.
  - 404 : المسار غير موجود.
  - 500 : خطأ في السيرفر.
- نصيحة : استخدم Postman لاختبار استجابات API

6- التعامل مع قواعد البيانات :

- التكامل مع MongoDB مثال شائع :
  - تثبيت mongoose :

```
npm install mongoose
```

  - الاتصال بقاعدة البيانات :

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/mydb', { useNewUrlParser: true });
```

  - إنشاء نموذج (Model) :

```
const User = mongoose.model('User', { name: String, email: String });
```

  - حفظ بيانات :

```
app.post('/user', async (req, res) => {

  const user = new User(req.body);

  await user.save();

  res.send('User saved!');

});
```

- نصيحة :جرب إنشاء API يخزن بيانات في MongoDB

7- بناءRESTful API :

- إيه هو؟ : واجهة برمجية لتبادل البيانات باستخدام HTTP
- مثال API بسيط :

```
let users = [];

app.get('/users', (req, res) => res.json(users));

app.post('/users', (req, res) => {

  users.push(req.body);

  res.status(201).send('User added');

});

app.get('/users/:id', (req, res) => {

  const user = users.find(u => u.id === req.params.id);

  if (user) res.json(user);

  else res.status(404).send('User not found');

});
```

- نصيحة : جرب بناء API بسيط واختبره بـ Postman

8- الأمان في Express :

- المهم :
  - Helmet : لإضافة هيدرز أمان :

```
npm install helmet

javascript

const helmet = require('helmet');

app.use(helmet());
```

  - CORS : للسماح بالوصول من مصادر مختلفة :

```
npm install cors

javascript

const cors = require('cors');

app.use(cors());
```

  - تجنب XSS وCSRF : تحقق من المدخلات واستخدم مكتبات زي csurf
- نصيحة : جرب تثبيت Helmet وشوف الهيدرز في المتصفح.
