REST API with Express 5.3 · Node.js 26 LTS

بناء REST API باستخدام Express.js

من مفهوم HTTP والـRouting لحد بناء Courses API بخطوات واضحة، ثم اختبار المدخلات والأخطاء وحالات الحدود.

الإصدار المستهدف
Node.js 26 LTS / Express 5.3.0

إعداد
محمود منصور (Mahmoud Mansour)

تاريخ الإصدار
24 نوفمبر 2026

المنصة
khatwa tech

مذكرة مراجعة عملية للمبتدئين والمتوسطين، بأمثلة قابلة للتجربة، ومشروع، وتدريبات وحلول.

youtube.com/@khatwatech

NODE26_EXPRESS53_REST_API / 02

خريطة المذكرة

خطة مراجعة منظمة؛ الأرقام تشير إلى أرقام صفحات النسخة المطبوعة.

طريقة استخدام المذكرة

1) اقرأ الفكرة بسرعة. 2) توقّع الناتج قبل ما تبص على الحل. 3) طبّق المشروع وعدّل عليه.

القسم / الموضوع	الصفحات
REST وHTTP	3–4
إعداد Express وأول Server	5–6
Routes وRequest وResponse	7–12
CRUD Operations	13–16
Middleware وValidation	17–18
Router وControllers	19–20
Errors وAsync Handlers	21–22
مشروع Courses REST API	23–26
أخطاء مشهورة وCheat Sheet	27–28
اختبر نفسك	29–30
الإجابات والحالات الحدّية	31–34
المراجع	35

نصيحة: كل مثال Express مستقل بنسخته المصغرة. اطلب الـendpoint المذكور بعد تشغيل الملف، وراجع Status Code وJSON سوا.

NODE26_EXPRESS53_REST_API / 03

01. REST API من البداية

API هو اتفاق لتبادل البيانات، وREST أسلوب تصميم شائع فوق HTTP.

Client
متصفح أو Mobile App أو Postman بيبعث Request.

Server
تطبيق Express بيستقبل الطلب ويحدد الرد.

Endpoint
مسار زي /api/courses بيحدد موردًا.

JSON
صيغة بيانات معتادة في REST APIs.

طلب	رد مناسب
GET /api/courses	200 + قائمة JSON
GET /api/courses/99	404 + رسالة خطأ
POST /api/courses	201 + بيانات الكورس الجديد
DELETE /api/courses/1	204 بدون Response Body

الفرق اللي يهمك: REST مش مكتبة. Express Framework بيساعدك تنفّذ HTTP API بطريقة منظمة.

NODE26_EXPRESS53_REST_API / 04

HTTP Request & Response .01

أي Request فيه Method وURL وHeaders وقد يحتوي Body، والـResponse فيه Status وBody.

مثال HTTP تعليمي

```
GET /api/courses/2 HTTP/1.1
Host: localhost:3000
Accept: application/json
```

رد السيرفر المتوقع

```
HTTP/1.1 200 OK
Content-Type: application/json

{"id":2,"title":"SQL Fundamentals","price":450}
```

Method	الاستخدام
GET	قراءة
POST	إنشاء
PUT	استبدال بيانات المورد
PATCH	تعديل جزئي
DELETE	حذف

انتبه: الأكواد 200 و201 و204 و400 و404 و500 مش كلمات تزيين؛ اختار كود مناسب للحالة الفعلية.

NODE26_EXPRESS53_REST_API / 05

02. إعداد المشروع بـnpm

هنستخدم ES Modules وExpress 5.3.0 بدل CommonJS في المشروع الجديد.

أوامر التهيئة

```
mkdir courses-api
cd courses-api
npm init -y
npm install express@5.3.0
```

الأثر المتوقع

يُنشئ package.json ويثبت Express عند توفر الإنترنت.

أهم أجزاء package.json

```
{
  "type": "module",
  "scripts": { "start": "node src/app.js" },
  "dependencies": { "express": "5.3.0" }
}
```

الفرق اللي يهمك: مع `"type":"module"` تقدر تستخدم import/export داخل ملفات .js.

NODE26_EXPRESS53_REST_API / 06

02. أول Express Server

Express بيستقبل الطلب في Route Handler وبيبعت الرد بـres.

GET http://localhost:3000/health

```
import express from 'express';
const app = express();
app.get('/health', (_req, res) => {
  res.json({ status: 'ok' });
});
app.listen(3000);
```

الرد المتوقع بعد طلب GET

```
HTTP 200
{"status":"ok"}
```

طريقة التشغيل: احفظ الكود في app.js وشغّل `node app.js` بعد npm install. السيرفر يظل شغّال لحد ما توقفه.

انتبه: ده مشروع Console Server مش صفحة ويب؛ جرّب الـRoute بـcurl أو Postman أو المتصفح لطلبات GET.

NODE26_EXPRESS53_REST_API / 07

GET Routes .03

لكل HTTP Method ممكن تعمل Handler مستقل على نفس المسار.

GET /api/courses

```
import express from 'express';
const app = express();
app.get('/api/courses', (_req, res) => {
  res.json([{ id: 1, title: 'Python' }]);
});
app.listen(3000);
```

الرد المتوقع

```
HTTP 200
[{"id":1,"title":"Python"}]
```

جرّب بنفسك: زود عنصر تاني في الـArray وخلي الـClient يقارن النتيجتين.

NODE26_EXPRESS53_REST_API / 08

Route Parameters .03

استخدم req.params لو جزء من مسار URL متغير زي id.

GET /courses/12

```
import express from 'express';
const app = express();
app.get('/courses/:id', (req, res) => {
  res.json({ id: req.params.id });
});
app.listen(3000);
```

الرد المتوقع

```
HTTP 200
{"id":"12"}
```

انتبه: req.params.id بييجي String؛ لو هتستخدمه كرقم، حوّله وتأكد إنه Integer صحيح قبل البحث.

NODE26_EXPRESS53_REST_API / 09

Query Parameters .03

Query String مناسب للبحث والفلترة مش لتعريف مورد بعينه.

GET /search?term=python

```
import express from 'express';
const app = express();
app.get('/search', (req, res) => {
  res.json({ term: req.query.term ?? '' });
});
app.listen(3000);
```

الرد المتوقع

```
HTTP 200
{"term":"python"}
```

Params	Query
`/courses/:id`	`/search?term=...`
لتحديد مورد واحد	للفلترة والبحث

NODE26_EXPRESS53_REST_API / 10

03. JSON Parser و Request Body

req.body مش مضمون يكون موجود لحد ما تضيف Middleware مناسب.

POST /echo مع Body: {"title":"Python"}

```
import express from 'express';
const app = express();
app.use(express.json());
app.post('/echo', (req, res) => {
  res.status(201).json({ title: req.body?.title });
});
app.listen(3000);
```

الرد المتوقع

```
HTTP 201
{"title":"Python"}
```

خطأ مشهور: لو نسيت `express.json()` هتلاقي req.body undefined لطلبات JSON. افحص نوع المدخل قبل استخدامه.

NODE26_EXPRESS53_REST_API / 11

Response Helpers .03

اختار res.json وres.status وres.sendStatus حسب محتوى الرد.

مثالان مستقلان في نفس Server

```
import express from 'express';
const app = express();
app.post('/created', (_req, res) => {
  res.status(201).json({ created: true });
});
app.delete('/empty', (_req, res) => {
  res.status(204).end();
});
app.listen(3000);
```

الردان المتوقعان

```
POST /created → HTTP 201, {"created":true}
DELETE /empty → HTTP 204, no body
```

الفرق اللي يهمك: مع 204 مفيش Response Body. كتابة res.json بعد status(204) مش مفيدة.

NODE26_EXPRESS53_REST_API / 12

04. Status Codes اللي هتقابلها

كود الحالة بيقول للـClient نوع نتيجة الطلب.

Status	المعنى	متى تستخدمه
OK 200	نجاح	GET أو PUT/PATCH ناجح
Created 201	تم الإنشاء	POST ناجح
No Content 204	نجاح بلا Body	DELETE ناجح
Bad Request 400	مدخلات غير صالحة	JSON خطأ أو قيمة غير صحيحة
Not Found 404	المورد مش موجود	id غير موجود
Server Error 500	مشكلة غير متوقعة	خطأ داخلي

انتبه: 404 مش معناه Server وقع. فرّق بين مسار مش موجود ومورد غير موجود داخل مسار معروف.

NODE26_EXPRESS53_REST_API / 13

04. GET All وGET by ID

قبل أي عملية تعديل، جهّز القراءة والتحقق من ID.

قراءة كل العناصر وعنصر واحد

```
import express from 'express';
const app = express();
const courses = [{ id: 1, title: 'Python' }];
app.get('/courses', (_req, res) => res.json(courses));
app.get('/courses/:id', (req, res) => {
  const id = Number(req.params.id);
  const item = courses.find(c => c.id === id);
  if (!item) return res.status(404).json({ error: 'Not found' });
  return res.json(item);
});
app.listen(3000);
```

الردان المتوقعان

```
GET /courses/1 → HTTP 200, {"id":1,"title":"Python"}
GET /courses/99 → HTTP 404, {"error":"Not found"}
```

جرّب بنفسك: جرّب ID موجود وID غير موجود وشوف الفرق بين 200 و404.

NODE26_EXPRESS53_REST_API / 14

04. POST: أنشئ عنصرًا

POST يستقبل بيانات جديدة، ويتحقق منها، وبعد النجاح يرجع 201.

الإنشاء بعد فحص title

```
import express from 'express';
const app = express();
app.use(express.json());
const courses = [];
app.post('/courses', (req, res) => {
  if (typeof req.body?.title !== 'string' || !req.body.title.trim())
    return res.status(400).json({ error: 'Invalid title' });
  const item = { id: 1, title: req.body.title.trim() };
  courses.push(item);
  return res.status(201).json(item);
});
app.listen(3000);
```

الرد المتوقع

```
POST {"title":"SQL"} → HTTP 201
{"id":1,"title":"SQL"}
```

انتبه: ده مثال مصغر في الذاكرة؛ في التطبيق الكامل هنولّد ID غير مكرر لكل كورس.

NODE26_EXPRESS53_REST_API / 15

04. PUT وPATCH

PUT غالبًا يستبدل بيانات المورد المطلوبة كاملة، بينما PATCH يعدّل بعض الحقول.

PUT ثم PATCH على نفس المورد

```
import express from 'express';
const app = express();
app.use(express.json());
const course = { id: 1, title: 'Python', price: 600 };
app.put('/courses/1', (req, res) => {
  if (typeof req.body?.title !== 'string' ||
      typeof req.body?.price !== 'number')
    return res.status(400).json({ error: 'Invalid body' });
  course.title = req.body.title;
  course.price = req.body.price;
  return res.json(course);
});
app.patch('/courses/1', (req, res) => {
  if (typeof req.body?.price !== 'number')
    return res.status(400).json({ error: 'Invalid price' });
  course.price = req.body.price;
  return res.json(course);
});
app.listen(3000);
```

الردان المتوقعان

```
PUT {"title":"Python Pro","price":800} → HTTP 200
{"id":1,"title":"Python Pro","price":800}
ثم PATCH {"price":900} → HTTP 200
{"id":1,"title":"Python Pro","price":900}
```

الفرق اللي يهمك: في المشروع النهائي سنمنع Unknown Fields ونتحقق من السعر غير السالب.

NODE26_EXPRESS53_REST_API / 16

04. DELETE: احذف موردًا

لو الحذف نجح ورجعت 204، ما تبعتش Body.

حذف مرة ثم تكراره

```
import express from 'express';
const app = express();
let courses = [{ id: 1, title: 'SQL' }];
app.delete('/courses/:id', (req, res) => {
  const id = Number(req.params.id);
  if (!courses.some(c => c.id === id))
    return res.status(404).json({ error: 'Not found' });
  courses = courses.filter(c => c.id !== id);
  return res.status(204).end();
});
app.listen(3000);
```

الردان المتوقعان

```
DELETE /courses/1 → HTTP 204, no body
DELETE /courses/1 مرة ثانية → HTTP 404
{"error":"Not found"}
```

NODE26_EXPRESS53_REST_API / 17

05. Middleware: قبل الـRoute وبعده

Middleware Function بتستقبل req وres وnext؛ تقدر تسجل الطلب أو تمنعه.

Logging Middleware ثم GET /health

```
import express from 'express';
const app = express();
app.use((req, _res, next) => {
  console.log(req.method, req.path);
  next();
});
app.get('/health', (_req, res) => res.json({ ok: true }));
app.listen(3000);
```

الناتج المتوقع

```
Console: GET /health
HTTP 200
{"ok":true}
```

خطأ مشهور: لو Middleware ما بعتتش Response ولا نادت next()، الطلب ممكن يفضل معلّق.

NODE26_EXPRESS53_REST_API / 18

Input Validation .06

الـClient ممكن يبعث أي بيانات؛ فحص المدخلات مسؤولية السيرفر.

Function مستقلة قابلة للتشغيل بـNode.js

```
function validateCourse(body) {
  if (!body || typeof body !== 'object' || Array.isArray(body))
    return 'Body must be an object';
  if (typeof body.title !== 'string' || !body.title.trim())
    return 'Invalid title';
  if (!Number.isFinite(body.price) || body.price < 0)
    return 'Invalid price';
  return null;
}
console.log(validateCourse({title:'SQL',price:450}));
console.log(validateCourse({title:'',price:450}));
```

الناتج الفعلي

```
null
Invalid title
```

انتبه: Number.isFinite بيستبعد NaN وInfinity. لازم كمان تتحقق من أنواع الحقول وصلاحيات المستخدم في المشاريع الحقيقية.

NODE26_EXPRESS53_REST_API / 19

Express Router .07

قسّم المسارات حسب المورد بدل ما تحط كل Route في app.js.

ملفان مستقلان في مشروع واحد

```
// src/routes/courses.routes.js
import { Router } from 'express';
const router = Router();
router.get('/', (_req, res) => res.json([]));
export default router;

// src/app.js
import express from 'express';
import coursesRouter from './routes/courses.routes.js';
const app = express();
app.use('/api/courses', coursesRouter);
app.listen(3000);
```

الرد المتوقع

```
GET /api/courses → HTTP 200
[]
```

جرّب بنفسك: الـRouter لما يتسجل على /api/courses، المسار / داخله يساوي /api/courses خارجًا.

NODE26_EXPRESS53_REST_API / 20

Controllers .07

الـController بيجمع منطق الطلب، والـRouter بيحدد الـEndpoint.

مثال ملفات مترابطة

```
// controller.js
export function listCourses(_req, res) {
  res.json([{ id: 1, title: 'Python' }]);
}
// routes.js
import { Router } from 'express';
import { listCourses } from './controller.js';
const router = Router();
router.get('/', listCourses);
export default router;
```

الرد المتوقع

```
GET للمسار المركّب → HTTP 200
[{"id":1,"title":"Python"}]
```

Router	Controller
يختار المسار وطريقة HTTP	ينفذ منطق القراءة أو التعديل
ملف للمسارات	ملف لدوال معالجة الطلبات

NODE26_EXPRESS53_REST_API / 21

08. 404 وError Middleware

ضع Middleware عدم وجود Route بعد كل Routes؛ ومعالجة الأخطاء بأربع Parameters.

الترتيب الصحيح للـMiddleware

```
import express from 'express';
const app = express();
app.get('/health', (_req, res) => res.json({ ok: true }));
app.use((_req, res) => res.status(404).json({error:'Route not found'}));
app.use((err, _req, res, _next) => {
  console.error(err.message);
  res.status(500).json({error:'Internal server error'});
});
app.listen(3000);
```

الرد المتوقع

```
GET /missing → HTTP 404
{"error":"Route not found"}
```

انتبه: الـ404 عادي مش لازم يبقى Exception. سجّل الخطأ الحقيقي على السيرفر، ولا تسرّب تفاصيله الداخلية للـClient.

NODE26_EXPRESS53_REST_API / 22

08. Async Errors في Express 5

Express 5 بيمرر أخطاء Promise المرفوضة تلقائيًا لـError Middleware.

Async Route يفشل بأمان

```
import express from 'express';
const app = express();
app.get('/fail', async () => {
  throw new Error('Database unavailable');
});
app.use((err, _req, res, _next) => {
  res.status(500).json({ error: 'Internal server error' });
});
app.listen(3000);
```

الرد المتوقع

```
GET /fail → HTTP 500
{"error":"Internal server error"}
```

الفرق اللي يهمك: مش محتاج asyncHandler خارجي لكل Route في Express 5؛ لكن لسه محتاج تفكير في Logging وResponse Error Design.

NODE26_EXPRESS53_REST_API / 23

المشروع: Courses REST API — البنية

الملفات دي جزء من المشروع الكامل المرفق؛ ابدأ بـnpm install ثم npm start.

package.json

```
{
  "name": "khatwa-courses-rest-api",
  "version": "1.0.0",
  "private": true,
  "type": "module",
  "scripts": {"start": "node src/app.js", "dev": "node --watch src/app.js", "test": "node --test"},
  "dependencies": {"express": "5.3.0"},
  "engines": {"node": ">=26"}
}
```

src/app.js — السيرفر وربط الـRoutes

```
import express from 'express';
import coursesRouter from './routes/courses.routes.js';
import { notFound, errorHandler } from './middleware/errorHandler.js';

const app = express();
app.use(express.json({ limit: '10kb' }));
app.get('/health', (_req, res) => res.json({ status: 'ok' }));
app.use('/api/courses', coursesRouter);
app.use(notFound);
app.use(errorHandler);

const port = Number(process.env.PORT) || 3000;
app.listen(port, (error) => {
  if (error) { console.error(error); process.exitCode = 1; return; }
  console.log(`API running on port ${port}`);
});
```

ملاحظة: البيانات في الذاكرة فقط. هترجع لحالتها الأصلية عند إعادة تشغيل البرنامج.

NODE26_EXPRESS53_REST_API / 24

المشروع: تخزين وفحص البيانات

الـController بيتعامل مع قواعد واضحة للـID والبيانات؛ ده اللي يمنع رسائل خطأ عشوائية.

src/data/courses.js

```
export const courses = [
  { id: 1, title: 'Python Basics', price: 600 },
  { id: 2, title: 'SQL Fundamentals', price: 450 }
];
export function resetCourses() {
  courses.splice(0, courses.length,
    { id: 1, title: 'Python Basics', price: 600 },
    { id: 2, title: 'SQL Fundamentals', price: 450 });
}
```

src/controllers/courses.controller.js — الجزء الأول

```
import { courses } from '../data/courses.js';

function parseId(value) {
  const id = Number(value);
  return Number.isSafeInteger(id) && id > 0 && String(id) === value ? id : null;
}

function validateCourse(body, partial = false) {
  if (!body || typeof body !== 'object' || Array.isArray(body)) {
    return 'Body must be a JSON object';
  }
  const fields = Object.keys(body);
  if (fields.some(key => !['title', 'price'].includes(key))) {
    return 'Unknown field';
  }
  if (!partial || 'title' in body) {
    if (typeof body.title !== 'string' || !body.title.trim()) {
      return 'title must be a non-empty string';
    }
  }
  if (!partial || 'price' in body) {
    if (!Number.isFinite(body.price) || body.price < 0) {
      return 'price must be a non-negative number';
    }
  }
  if (partial && fields.length === 0) return 'At least one field is required';
  return null;
}

function locate(req, res) {
  const id = parseId(req.params.id);
  if (id === null) { res.status(400).json({ error: 'Invalid id' }); return null; }
  const course = courses.find(item => item.id === id);
  if (!course) { res.status(404).json({ error: 'Course not found' }); return null; }
  return course;
}

export function listCourses(_req, res) { res.json(courses); }

export function getCourse(req, res) {
  const course = locate(req, res);
  if (course) res.json(course);
}
```

انتبه: ملف الـController بيتكمل في الصفحة التالية؛ انسخه كله من ZIP بدل نسخ جزء منفصل.

NODE26_EXPRESS53_REST_API / 25

المشروع: عمليات الإنشاء والتعديل

الـPOST ينشئ عنصرًا جديدًا، والـPUT وPATCH لازم يتصرفوا بشكل مختلف.

src/controllers/courses.controller.js — تكملة الملف

```
export function createCourse(req, res) {
  const error = validateCourse(req.body);
  if (error) return res.status(400).json({ error });
  const id = Math.max(0, ...courses.map(item => item.id)) + 1;
  const course = { id, title: req.body.title.trim(), price: req.body.price };
  courses.push(course);
  return res.status(201).json(course);
}

export function replaceCourse(req, res) {
  const course = locate(req, res);
  if (!course) return;
  const error = validateCourse(req.body);
  if (error) return res.status(400).json({ error });
  course.title = req.body.title.trim();
  course.price = req.body.price;
  return res.json(course);
}

export function updateCourse(req, res) {
  const course = locate(req, res);
  if (!course) return;
  const error = validateCourse(req.body, true);
  if (error) return res.status(400).json({ error });
  if ('title' in req.body) course.title = req.body.title.trim();
  if ('price' in req.body) course.price = req.body.price;
  return res.json(course);
}

export function deleteCourse(req, res) {
  const course = locate(req, res);
  if (!course) return;
  courses.splice(courses.indexOf(course), 1);
  return res.status(204).end();
}
```

جرّب بنفسك: ابدأ بعمليات قراءة ثم إنشاء؛ اختبر عنوان فاضي وسعر سالب وID غير صحيح.

NODE26_EXPRESS53_REST_API / 26

المشروع: Router وError Handling والناتج

نربط الـRoutes بملفات الـControllers، ثم نختبر الـEndpoints بـcurl.

src/routes/courses.routes.js

```
import { Router } from 'express';
import {
  listCourses, getCourse, createCourse,
  replaceCourse, updateCourse, deleteCourse
} from '../controllers/courses.controller.js';

const router = Router();
router.get('/', listCourses);
router.get('/:id', getCourse);
router.post('/', createCourse);
router.put('/:id', replaceCourse);
router.patch('/:id', updateCourse);
router.delete('/:id', deleteCourse);
export default router;
```

src/middleware/errorHandler.js

```
export function notFound(_req, res) {
  res.status(404).json({ error: 'Route not found' });
}

export function errorHandler(err, _req, res, _next) {
  if (err.type === 'entity.parse.failed') {
    return res.status(400).json({ error: 'Invalid JSON' });
  }
  console.error(err);
  return res.status(500).json({ error: 'Internal server error' });
}
```

الطلب	HTTP	Body
GET /health	200	`{"status":"ok"}`
GET /api/courses/2	200	`{"id":2,"title":"SQL Fundamentals","price":450}`
POST /api/courses	201	`{"id":3,"title":"Node.js Basics","price":750}`
DELETE /api/courses/1	204	بدون Body

طوّر المشروع: (1) أضف بحث بالاسم. (2) أضف Pagination. (3) اربط Database بدل Array.

NODE26_EXPRESS53_REST_API / 27

الأخطاء الشائعة في Express

ابدأ فحص المشكلة من URL وMethod وBody، وبعدها Middleware.

المشكلة / الرسالة	السبب المحتمل	تعمل إيه
Cannot find package express	الحزمة مش مثبّتة	شغّل npm install
Cannot use import statement	ES Modules مش متفعلة	أضف type: module
req.body is undefined	نسيت express.json أو Content-Type	أضف Parser واستخدم JSON
Route not found 404	الـMethod أو المسار غلط	راجع مسار Router وPrefix
Invalid id 400	ID غير صالح	أرسل Integer موجب
Internal server error 500	خطأ في معالجة الطلب	راجع Logs وError Middleware
EADDRINUSE	البورت مشغول	غيّر PORT أو أوقف العملية القديمة

انتبه: بعض الشروحات القديمة تستخدم Express 4 signatures؛ راجع دليل الترقية للإصدار 5.

NODE26_EXPRESS53_REST_API / 28

Express & REST API Cheat Sheet

أهم الصيغ اللي هترجع لها أثناء بناء الـAPI.

عايز أعمل	الصيغة المختصرة
تثبت Express	`npm install express@5.3.0`
تفعّل ESM	`"type": "module"`
تنشئ Server	`const app = express()`
تقرأ JSON	`app.use(express.json())`
تعمل Route	`app.get('/path', handler)`
تقرأ ID	`req.params.id`
تقرأ Search	`req.query.term`
تقرأ JSON Body	`req.body`
ترجع JSON	`res.json(data)`
ترجع Status	`res.status(201).json(data)`
تحذف بدون Body	`res.status(204).end()`
تعمل Router	`Router()`
تسجل Router	`app.use('/api', router)`
تتعامل مع أخطاء	`(err, req, res, next) => ...`

خطوة صغيرة: خلي عندك جدول لكل Endpoint: Method، URL، Input، Success، Failure.

NODE26_EXPRESS53_REST_API / 29

اختبر نفسك: 6 أسئلة توقّع الناتج

الأسئلة دي JavaScript logic جوه Node.js؛ شغّل كل كود بعد ما تتوقع نتيجته.

سؤال 1 — JSON Output

```
console.log(JSON.stringify({ status: 'ok' }));
```

سؤال 2 — Query Param

```
const p = new URL('http://localhost/courses?limit=2');
console.log(p.searchParams.get('limit'));
```

سؤال 3 — Filter logic

```
const grades = [60, 80, 100];
console.log(grades.filter(x => x >= 80).length);
```

سؤال 4 — Validation

```
const body = { price: 0 };
console.log(Number.isFinite(body.price) && body.price >= 0);
```

سؤال 5 — Lookup

```
const list = [{id:1},{id:2}];
console.log(list.find(x=>x.id===3) ?? 'Missing');
```

سؤال 6 — PATCH-like update

```
const c = {id:1,title:'SQL',price:450};
console.log(JSON.stringify({...c,price:500}));
```

NODE26_EXPRESS53_REST_API / 30

اختبر نفسك: 6 تمارين اكتب الحل

اكتب Route Handler كاملًا لكل حالة. اختبر Method وURL وBody وStatus.

تمرين 1: GET /hello يرجع JSON

Input: GET /hello → Output: {"message":"Hello"}

تمرين 2: GET /courses/:id يقرأ رقم

Input: GET /courses/7 → Output: {"id":7}

تمرين 3: GET /search يقرأ term

Input: GET /search?term=SQL → Output: {"term":"SQL"}

تمرين 4: POST /echo يرجع 201

Input: POST {"title":"Python"} → Output: 201

تمرين 5: 404 لمورد غير موجود

Input: GET /courses/99 → Output: 404

تمرين 6: DELETE يرجع 204

Input: DELETE /courses/1 → Output: no body

طريقة الحل: ابدأ بكل تمرين في ملف .js مستقل فيه import express وإنشاء app وlisten.

NODE26_EXPRESS53_REST_API / 31

الإجابات: توقّع الناتج (1–6)

النتيجة هنا مع تفسير مختصر؛ دي أكواد JavaScript مستقلة وتم تشغيلها بـNode.js.

السؤال	الناتج الفعلي	التفسير
1	`{"status":"ok"}`	JSON Output
2	`2`	Query Param
3	`2`	Filter logic
4	`true`	Validation
5	`Missing`	Lookup
6	`{"id":1,"title":"SQL","price":500}`	PATCH-like update

تفصيل مثال سؤال 3

```
const grades = [60, 80, 100];
console.log(grades.filter(x => x >= 80).length);
```

ناتج تشغيل Node

```
2
```

الفرق اللي يهمك: دالة find بتعيد undefined لو مفيش تطابق؛ ?? بتبدلها بالقيمة البديلة.

NODE26_EXPRESS53_REST_API / 32

حلول تمارين Express (1-3)

الملفات مستقلة؛ كل ملف اسمه app.js ويعمل على localhost:3000.

حل تمرين 1: GET /hello يرجع JSON

```
import express from 'express';
const app = express();
app.get('/hello',(_req,res)=>res.json({message:'Hello'}));
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 200
{"message":"Hello"}
```

حل تمرين 2: GET /courses/:id يقرأ رقم

```
import express from 'express';
const app = express();
app.get('/courses/:id',(req,res)=>res.json({id:Number(req.params.id)}));
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 200
{"id":7}
```

حل تمرين 3: GET /search يقرأ term

```
import express from 'express';
const app = express();
app.get('/search',(req,res)=>res.json({term:req.query.term??''}));
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 200
{"term":"SQL"}
```

NODE26_EXPRESS53_REST_API / 33

حلول تمارين Express (4-6)

ركّز على status(201) و404 و204، وكل واحدة ليها استخدام واضح.

حل تمرين 4: POST /echo يرجع 201

```
import express from 'express';
const app = express();
app.use(express.json());
app.post('/echo',(req,res)=>res.status(201).json({title:req.body.title}));
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 201
{"title":"Python"}
```

حل تمرين 5: 404 لمورد غير موجود

```
import express from 'express';
const app = express();
app.get('/courses/:id',(req,res)=>res.status(404).json({error:'Not found'}));
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 404
{"error":"Not found"}
```

حل تمرين 6: DELETE يرجع 204

```
import express from 'express';
const app = express();
app.delete('/courses/:id',(_req,res)=>res.status(204).end());
app.listen(3000);
```

الرد المتوقع عند طلب المثال

```
HTTP 204
(no body)
```

NODE26_EXPRESS53_REST_API / 34

اختبار المشروع: الحالات الحدّية

اختبارات الـController الفعلية بتراجع منطق العمل بدون الاعتماد على توفّر Express package محليًا.

الحالة	HTTP	الناتج
قائمة كورسات عند البداية	200	courses 2
GET id=99	404	`{"error":"Course not found"}`
GET id=1bad	400	`{"error":"Invalid id"}`
POST عنوان فارغ	400	`{"error":"title must be a non-empty string"}`
POST عنوان وسعر صالح	201	id=3
PATCH بجسم فاضي	400	`{"error":"At least one field is required"}`
PUT بيانات كاملة	200	Updated course
DELETE موجود	204	no body
JSON معطوب	400	`{"error":"Invalid JSON"}`

تشغيل اختبارات الـController على Node 22.16.0

```
npm test
```

ملخص التحقق

```
9 tests passed; 0 failed
```

حدود الاختبار: الـController Tests نجحت، وفحص Syntax لملفات المشروع نجح. تثبيت Express 5.3.0 وتشغيل HTTP End-to-End على Node 26 لم يتوفرا داخل بيئة التحقق الحالية.

NODE26_EXPRESS53_REST_API / 35

المراجع الرسمية والخطوة التالية

بعد المذكرة جرّب API حقيقي وأرسل إليه Requests بنجاح وفشل.

Node.js Releases / LTS
https://nodejs.org/en/about/previous-releases

Express 5.x API
https://expressjs.com/en/5x/api.html

Express Routing
https://expressjs.com/en/guide/routing.html

Express Error Handling
https://expressjs.com/en/guide/error-handling.html

Express 5 Migration
https://expressjs.com/en/guide/migrating-5.html

Express Middleware
https://expressjs.com/en/guide/using-middleware.html

npm package.json
https://docs.npmjs.com/cli/v11/configuring-npm/package-json

الخطوة العملية: نزّل المشروع، شغّل npm install وnpm start على Node.js 26، ثم اختبر 200 و201 و204 و400 و404 بـcurl أو Postman.

حدود البيئة: جرى اختبار منطق الـControllers والـJavaScript على Node.js 22.16.0، وليس على Node.js 26. حزمة Express 5.3 لم تتوفر للتثبيت محليًا أثناء التجهيز.