مختصر كتابي

Requirements and How to Use the Lab



ما الذي تحتاجه لتشغيل المعمل وقراءة نتائجه بثقة؟

المعمل الجيد ليس الذي فيه أدوات أكثر، بل الذي تستطيع تشغيله، وتعرف ما الذي تغيّر فيه، وتحصل منه على النتيجة نفسها كل مرة. نبدأ ببرنامج Server وبرنامج Client على جهازك، ويكفيك أن تكتب functions وشروطًا وتفهم الـobjects التي تحمل البيانات، دون معرفة سابقة بداخل النظام أو تفاصيل protocols الشبكة.

الإصدار الأساسي Node 24، وهو جاهز في المعمل تحت الدرس، والمعمل الأول لا يحتاج مكتبات خارجية. والطرفية (Terminal) تبحث عن اسم الملف في **المجلد الحالي** (Current Directory)، فإذا قالت إن الملف غير موجود فراجع المجلد الذي تقف فيه والمسار الذي كتبته.

نفتح طرفيتين. في الأولى يبقى الـServer يعمل وينتظر الـRequests، فلا تقبل أوامر جديدة، وهذا لا يعني أنه معطّل. وفي الثانية يرسل الـClient Request ويطبع النتيجة ثم ينتهي. العنوان رقم محلي يشير إلى جهازك نفسه، و**المنفذ** (Port) يحدد الخدمة التي تستقبل الـRequest، ومنفذنا 8100. إذا كانت خدمة أخرى تمسك العنوان والمنفذ نفسيهما فلن يبدأ الـServer. لا تغلق برامج عشوائيًا؛ تأكد أولًا أنه لا توجد نسخة قديمة من الـServer تعمل، أو اختر منفذًا آخر واكتبه في الـServer والـClient معًا.

جسم الطلب (Request Body) بصيغة JSON، أي بيانات مكتوبة نصًا، مثل object فيه رقم المنتج وعدد القطع. والرقم `1` يختلف عن النص `"1"`. وخدمتنا تتحقق أن عدد القطع رقم صحيح من 1 إلى 10؛ فـJSON ينقل البيانات فقط، والتطبيق هو الذي يقرر هل القيمة مقبولة. ورسالة HTTP فيها الـmethod والمسار والـbody ومعلومات أخرى، والرد فيه **رمز الحالة** (Status Code) والـheaders والـbody.

قد يُفتح Connection سليم ثم يأتي رد يرفض البيانات، وهذا يختلف تمامًا عن غياب الـServer. وإذا ردّ التطبيق بالرفض فلن يظهر خطأ في الكود وحده، لذلك نطبع الـstatus code مع الـbody: وصول الرد لا يعني قبول العملية. وانتبه إلى الوحدات: `1000` في كود المعمل تعني ثانية واحدة، لأن الوقت بالملي ثانية. وللأوامر حدود انتظار مقصودة حتى ينتهي الأمر إذا لم يرد الـServer، فلا تحذفها.

```
node --version
node server.mjs
node client.mjs health
```

الأمران الأولان في الطرفية الأولى، والثالث في `Terminal 2`. المتوقع `status: 200` و`ok: true`. وسجّل ما حدث فعلًا، ولا تغيّر رقمًا ليشبه المثال، فالجهاز المشغول ببرامج أخرى يغيّر القياس. وملف الاختبار المرفق يشغّل Server مدة قصيرة ويجري عمليات محددة ثم يغلقه؛ نجاحه يثبت الحالات المكتوبة فيه فقط، لا الجاهزية لأي ضغط. والبيانات في الـMemory وتُمسح عند إيقاف الـServer، فاكتب ملاحظاتك في ملف مستقل.

وفي التمرين تصنّف ثلاث مشكلات: أمر لا تعرفه الطرفية مشكلة في الأداة أو مسارها، وملف غير موجود يعني مراجعة المجلد واسم الملف، ورد يقول إن عدد القطع خطأ يعني أن الـClient وصل إلى التطبيق والتطبيق رفض البيانات.

- Node 24 جاهز في المعمل، ولا حاجة إلى مكتبات خارجية.
- طرفية للـServer تبقى مشغولة، وطرفية للـClient ترسل وتنتهي.
- العنوان والمنفذ 8100 يجب أن يتطابقا في الـServer والـClient.
- اقرأ الـstatus code مع الـbody، فوصول الرد لا يعني القبول.
- سجّل الأمر والإصدار والحالة قبل التشغيل والنتيجة الفعلية.

جرّب في المعمل تحت الدرس: شغّل `node server.mjs` في الطرفية الأولى، ثم `node client.mjs` في `Terminal 2`، ثم أوقف الـServer وأعد الأمر لترى خطأ Connection بدل رد HTTP `health`.

الخلاصة: حدّد أولًا هل المشكلة في الأداة أم في الملفات أم في عقد التطبيق، ثم سجّل ما حدث فعلًا.