جافاسكربت - الوحدة التاسعة: الوحدات ومعالجة الأخطاء وتصحيحها

تكبر البرامج وتكبر معها مشاكلها. وتبقي أربع خطوات البرنامج المتنامي سليما: التقسيم، والحماية، والفحص، والإصلاح. وتجيب كل خطوة عن سؤال بسيط، وتكفي للتدرب عليها أدوات المطور المجانية التي يضمها كل متصفح. وفي هذه الوحدة تستخدمها ريم، مهندسة برمجيات في فريق الفواتير، لتعرف لماذا يظهر NaN إجماليا لبعض الفواتير.

1. التقسيم

الوحدة البرمجية module ملف جافاسكربت يبقي متغيراته خاصة، ولا يشارك إلا ما يختار تصديره export. ويجلب ملف آخر هذه الأجزاء بتعليمة import، فيسمي ما يحتاجه والملف الذي يأتي منه. وفي صفحة الويب نحمل الملف الرئيسي بوسم script نوعه module، وينبغي تقديم الصفحة من عنوان ويب، لأن المتصفحات لا تحمل الوحدات من ملفات نفتحها مباشرة. والسؤال هو: ما الذي ينتمي إلى بعضه، وما الذي ينبغي أن تستطيع الملفات الأخرى استخدامه؟ والوحدات الصغيرة ذات المهمة الواضحة أسهل في الاختبار وإعادة الاستخدام، ولا يكتب بعضها فوق متغيرات بعض بالخطأ. والوحدة التي لا تصدر إلا دوال قليلة حسنة التسمية تعطي باقي البرنامج واجهة بسيطة.

مثال: تحتفظ الشركة بأدراج منفصلة للعقود والفواتير والرسائل، وعلى كل درج لافتة. ولا يختلط شيء من درج العقود بالرسائل بالخطأ. ويفتح الموظفون الدرج الذي يحتاجونه فقط ويخرجون منه ملفا واحدا. وهكذا تعمل الوحدات البرمجية: يبقي كل ملف متغيراته خاصة، ولا يشارك إلا الدوال التي يصدرها.

في العمل: يقع كود الفواتير عند ريم في ملف واحد طويل جدا. وتطرح سؤال التقسيم وتفصله إلى ثلاث وحدات: واحدة لقواعد الضريبة، وواحدة لتنسيق المبالغ، وواحدة لبناء الفاتورة. وتصدر كل وحدة دوالها الرئيسية فقط، وتستورد وحدة الفاتورة الوحدتين الأخريين. ويستطيع الزملاء الآن تغيير قواعد الضريبة دون لمس تصميم الفاتورة.

2. الحماية

ستقع أخطاء أثناء التشغيل، والكود الجيد يقرر سلفا ماذا يفعل. تحمل كتلة try الكود الذي قد يفشل، وتستقبل كتلة catch كائن الخطأ عندما يحدث ذلك، وفيه اسم ورسالة. وتعمل كتلة finally في كل الأحوال، وهي مكان التنظيف. ونستطيع أن نطلق خطأ خاصا بنا، مثل `throw new Error("Line without a price")`، فينشئ كائن خطأ برسالة واضحة. والسؤال هو: ما الذي قد يفشل هنا، وماذا ينبغي أن يحدث عندئذ؟ وكتلة catch التي تتجاهل الخطأ تخفي المشكلة وهي أسوأ من عدم المعالجة أصلا. والأخطاء ينبغي أن تلتقط حيث نعرف كيف نرد عليها، وأن تصل إلى المستخدم رسالة ودية.

مثال: يتحقق الصيدلي من غياب الجرعة في كل وصفة قبل تجهيزها. وعندما ينقص شيء يتصل بالطبيب ولا يخمن، فيتلقى الزبون جوابا واضحا بدل دواء خاطئ. والفحص يستغرق ثواني قليلة، أما التخمين الخاطئ فقد يكلف كثيرا. وهذه هي فكرة الحماية: نتوقع الفشل ونقرر سلفا ماذا نفعل عند حدوثه.

في العمل: تفحص دالة الإجمالي عند ريم الآن كل سطر في الفاتورة. وتطرح سؤال الحماية وتطلق خطأ رسالته سطر بلا سعر حين يغيب السعر. ويلتقط الكود الذي يبني الفاتورة ذلك الخطأ ويظهر للمحاسب رسالة واضحة بدل إجمالي هو NaN. وتخفي كتلة finally مؤشر التحميل في كل الأحوال.

3. الفحص

لإصلاح مشكلة يجب أن نراها أولا. تطبع الدالة console.log قيمة، وتطبع console.table مصفوفة كائنات في جدول، وهذا مثالي لقوائم أسطر الفواتير. وتظهر لوحة Console في أدوات المطور الأخطاء مع اسم الملف ورقم السطر،

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)

ويبين سجل الاستدعاء stack أي الدوال كانت تعمل. وفي لوحة Sources نضع نقطة توقف breakpoint على سطر، فيتوقف البرنامج عنده ونقرأ قيمة كل متغير ونتقدم سطرا سطرا. والتعليمة debugger توقف البرنامج بالطريقة نفسها عندما تكون أدوات المطور مفتوحة. والسؤال هو: ماذا يفعل الكود حقا، مقارنة بما نظن أنه يفعله؟ والنظر إلى القيم الفعلية ينهي التخمين.

مثال: لا يستبدل فني السيارات القطع عشوائيا، بل يوصل جهاز تشخيص ويقرأ ما يخبر به المحرك، مثل القطعة التي تفشل. وعندئذ يتجه الإصلاح إلى السبب الحقيقي لا إلى التخمين. ونقطة التوقف تعطينا التوصيل نفسه مع الكود، لأنها توقف البرنامج عند سطر وتتيح لنا قراءة كل قيمة. وهذا هو سبب أن النظر إلى القيم الفعلية ينهي التخمين.

في العمل: تطبع ريم أسطر الفاتورة بالدالة console.table فترى أن سطرا واحدا ليس له سعر إطلاقا. وتطرح سؤال الفحص وتضع نقطة توقف في دالة الإجمالي لتراقب القيم سطرا سطرا. ويبين سجل الاستدعاء في رسالة الخطأ أي الدوال قادت إلى السطر الفاشل. وخلال دقائق تعرف أي سطر يعطل الإجمالي.

4. الإصلاح

يبدأ الإصلاح بإعادة إنتاج المشكلة في كل مرة، بأصغر مثال يفشل. وقراءة رسالة الخطأ بعناية كثيرا ما تعطي الجواب: فالخطأ ReferenceError يعني أن اسما غير معرف، والخطأ TypeError يعني أن قيمة ليست من النوع الذي توقعه الكود، كقراءة خاصية من undefined، والخطأ SyntaxError يعني أن الكود غير صالح فلا يعمل الملف أصلا. ثم نكون تخمينا عن السبب، ونغير شيئا واحدا، ونختبر من جديد. والسؤال هو: ما أصغر تغيير يصلح السبب، لا العرض فقط؟ وبعد الإصلاح نعيد الاختبار الذي فشل، ونبقيه فحصا للمستقبل. وتغيير عدة أشياء دفعة واحدة يجعل من المستحيل معرفة أي تغيير نفع.

مثال: يسأل الطبيب أولا متى بدأ الألم وماذا يزيده، ثم يختبر فكرة واحدة في كل مرة. وإعطاء ثلاثة أدوية معا يخفي أيها نفع، فلا يعرف أحد ماذا يكرر في المرة القادمة. وبالطريقة نفسها يعيد المبرمج إنتاج المشكلة، ويغير شيئا واحدا، ثم يختبر من جديد. وهذا هو سبب أننا نغير شيئا واحدا في كل مرة.

في العمل: تعيد ريم إنتاج المشكلة بفاتورة واحدة فيها سطر واحد. وتبين رسالة الخطأ والجدول أن اسم الخاصية كتب بإملاء مختلف في البيانات، فصار السعر undefined ونتج عن الحساب NaN. وتطرح سؤال الإصلاح وتصحح ربط البيانات حيث ينشأ الاسم، بدل إخفاء NaN في النهاية. وتبقي الفاتورة الفاشلة اختبارا دائما، فلا تعود المشكلة دون أن يلحظها أحد.

الخاتمة

أصبح كود الفواتير عند ريم مقسما إلى وحدات واضحة، محميا من البيانات الناقصة، سهل الفحص. فالتقسيم يسأل ما الذي ينتمي إلى بعضه وما الذي يستخدمه الآخرون، والحماية تسأل ما الذي قد يفشل وماذا يحدث عندئذ. والفحص يسأل ماذا يفعل الكود حقا، والإصلاح يسأل عن أصغر تغيير يزيل السبب. وفي الوحدة التالية نخرج من المتصفح ونشغل جافاسكربت بالأداة Node.js ومدير الحزم npm.

المصادر

- شبكة مطوري موزيلا (MDN): وحدات جافاسكربت – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules
- شبكة مطوري موزيلا (MDN): جملة try...catch – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch
- شبكة مطوري موزيلا (MDN): الكائن Error – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error
- Chrome for Developers: تصحيح أخطاء جافاسكربت – https://developer.chrome.com/docs/devtools/javascript

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)