جافاسكربت - الوحدة الثامنة: جافاسكربت غير المتزامنة وأداة fetch

بعض المهام تستغرق وقتا، مثل طلب بيانات من خادم، ويجب ألا تتجمد الصفحة وهي تنتظر. والكود غير المتزامن يتيح لجافاسكربت أن تبدأ مثل هذه المهمة وتتابع أعمالا أخرى حتى يصل الجواب. وتجعل أربع خطوات ذلك سهل الإدارة: الانتظار، والوعد، والتريث، والجلب. وتجيب كل خطوة عن سؤال بسيط، وتستطيع نافذة console في المتصفح تشغيل كل سطر. وفي هذه الوحدة يستخدمها بسام، مطور واجهات في شركة شحن، ليبني لوحة تتبع مباشرة تظهر حالة طرد.

1. الانتظار

تشغل الصفحة كود جافاسكربت على مسار واحد، أي مهمة واحدة في كل مرة، فقد تجمد مهمة بطيئة كل شيء. والسؤال هو: هل يمكن للعمل الآخر أن يستمر أثناء الانتظار؟ وفي الكود غير المتزامن تبدأ جافاسكربت مهمة بطيئة، مثل مؤقت أو طلب إلى خادم، ثم تمضي. وعندما تنتهي المهمة البطيئة يشغل المتصفح الكود الذي أعد للنتيجة. ولذلك تبقى الصفحة مستجيبة والبيانات في الطريق. ومن الأخطاء الشائعة أن نكتب السطر التالي كأن الجواب وصل، وهو لم يصل بعد.

مثال: يضع الطباخ الأرز على النار ويضبط مؤقتا. وبدلا من التحديق في القدر يقطع الخضروات ويجهز المائدة. فالأرز هو المهمة البطيئة، والمؤقت هو الكود الذي أعد للنتيجة، والتقطيع هو بقية الصفحة التي تتابع عملها. وعندما يدق المؤقت يعود الطباخ إلى القدر، ولم تضع أي دقيقة أثناء الانتظار.

في العمل: تسأل لوحة التتبع عند بسام خدمة التتبع في الشركة عن حالة طرد، وهذا يستغرق لحظة. ويطرح سؤال الانتظار ويترك باقي الصفحة يعمل. ويستطيع الزائر أن يتصفح ويفتح القائمة ويقرأ تفاصيل أخرى. وصندوق الحالة وحده ينتظر جوابه.

2. الوعد

الوعد promise كائن يمثل نتيجة لم تجهز بعد. يبدأ معلقا pending، ثم يصير منجزا fulfilled بقيمة أو مرفوضا rejected بخطأ. والسؤال هو: هل انتهى الوعد، وهل نجح أم فشل؟ ونرفق الدالة then لاستقبال القيمة عند النجاح، والدالة catch لمعالجة الخطأ عند الفشل. وتعمل الدالة finally في الحالتين، وهي مكان جيد لإزالة رسالة التحميل. ويمكن سلسلة الوعود، فتصير نتيجة خطوة مدخل الخطوة التالية.

مثال: يعطي المطعم الزبون جهاز تنبيه بعد تسجيل الطلب. الجهاز هو الوعد: يبقى معلقا ما دام الطعام قيد التحضير، ويستطيع الزبون أن يجلس ويقرأ. وعندما تجهز الوجبة يرن الجهاز، وهذا وعد منجز. وإذا نفد صنف يشرح الطاقم ذلك بدل الرنين، وهذا وعد مرفوض، فيحصل الزبون دائما على جواب واحد واضح.

في العمل: يظهر بسام رسالة تحميل فور بدء الطلب. ويطرح سؤال الوعد ويستخدم then لعرض الحالة عند الإنجاز و catch لإظهار رسالة ودية عند الرفض. وتخفي الدالة finally رسالة التحميل في الحالتين. ولا يحدق الزوار أبدا في صندوق فارغ أو في مؤشر لا يتوقف.

3. التريث

تجعل الكلمتان async و await كود الوعود يقرأ ككود عادي خطوة خطوة. نكتب async قبل الدالة، وداخلها نكتب await أمام وعد لتتوقف تلك الدالة إلى أن ينتهي الوعد. والتوقف يؤثر في تلك الدالة فقط وليس في الصفحة كلها. والسؤال

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

هو: أي خطوة يجب أن تنتهي قبل أن يعمل السطر التالي؟ ولمعالجة الإخفاق نضع الخطوات المنتظرة داخل كتلة try ونلتقط الخطأ في كتلة catch. والدالة async تعيد وعدا دائما، فيستطيع الكود الذي يستدعيها أن ينتظرها بدورها. والطلبات المستقلة لا ينبغي أن ينتظر بعضها بعضا، فالدالة Promise.all تنتظر عدة وعود تعمل معا حتى تنتهي كلها.

مثال: ينتظر الخباز ارتفاع العجين قبل الخبز، لأن الخبز يعتمد عليه. وفي أثناء ارتفاع العجين يسخن الخباز الفرن ويدهن الصينية، لأن أيا من الأمرين لا يعتمد على العجين. فلا ينتظر الخباز إلا حيث تحتاج الخطوة إلى سابقتها، ويبدأ كل ما سواها معا. وهذا الفرق بين الخطوات المعتمدة والمستقلة هو ما يعبر عنه await و Promise.all.

في العمل: يحتاج بسام إلى حالة الطرد وبيانات السائق، ولا يعتمد أحدهما على الآخر. ويطرح سؤال التريث، ويستخدم await لكل خطوة داخل كتلتي try و catch، ويلف الطلبين في Promise.all. فتجهز اللوحة بعد الأبطأ من الطلبين لا بعد مجموعهما. وإذا فشل أحد الطلبين تظهر كتلة catch رسالة واحدة واضحة.

4. الجلب

ترسل الدالة fetch طلبا إلى عنوان ويب وتعيد وعدا بالاستجابة. والسؤال هو: ماذا أرسل الخادم، وهل هو سليم؟ لا يرفض الوعد إلا إذا تعذر إرسال الطلب أصلا، كانقطاع الشبكة، لكنه لا يرفض عند حالة خطأ مثل صفحة غير موجودة. ولذلك نفحص الخاصية ok في الاستجابة أولا. ثم تحول الدالة json جسم الاستجابة إلى كائن جافاسكربت، وهي تعيد وعدا أيضا، فتحتاج إلى await هي الأخرى. فمثلا `const response = await fetch(trackingUrl)` ينتظر الاستجابة من العنوان المحفوظ في trackingUrl. وعندئذ فقط نستخدم الكائن لملء الصفحة.

مثال: يفتح قارئ صفحة أخبار، فيطلب المتصفح من خادم آخر العناوين. وإذا رد الخادم باعتذار مثل صفحة غير موجودة بدل العناوين، فلا ينبغي أن تعرض الصفحة الاعتذار كأنه خبر. لذلك تفحص الصفحة أولا هل الجواب سليم، وهذا ما تخبرنا به الخاصية ok، ثم تقرأ العناوين وتعرضها. والصفحة التي تتجاوز هذا الفحص قد تعرض رسالة خطأ مكان الأخبار.

في العمل: يجلب بسام عنوان التتبع لطرد. ويطرح سؤال الجلب ويفحص الخاصية ok قبل أي شيء، فينتج الطرد المفقود رسالة واضحة بدل لوحة معطلة. وحين تكون الاستجابة سليمة يقرؤها بالدالة json ويملأ صندوق الحالة. وتظهر اللوحة الآن بيانات مباشرة وتصمد أمام طلب فاشل.

الخاتمة

تحمل لوحة التتبع عند بسام الآن بيانات مباشرة دون أن تجمد الصفحة ودون أن تربك الزائر. فالانتظار يسأل هل يستمر العمل الآخر في الأثناء، والوعد يسأل هل انتهت النتيجة وهل نجحت. والتريث يسأل أي خطوة يجب أن تنتهي قبل أن يعمل السطر التالي، والجلب يسأل ماذا أرسل الخادم وهل هو سليم. وفي الوحدة التالية ننظم الكود في وحدات برمجية modules ونتعلم معالجة الأخطاء وتصحيحها.

المصادر

- شبكة مطوري موزيلا (MDN): الكائن Promise – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
- شبكة مطوري موزيلا (MDN): الدالة async – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function
- شبكة مطوري موزيلا (MDN): العامل await – https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await
- شبكة مطوري موزيلا (MDN): استخدام واجهة Fetch – https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- شبكة مطوري موزيلا (MDN): الدالة json والخاصية ok في استخدام واجهة Fetch – https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

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