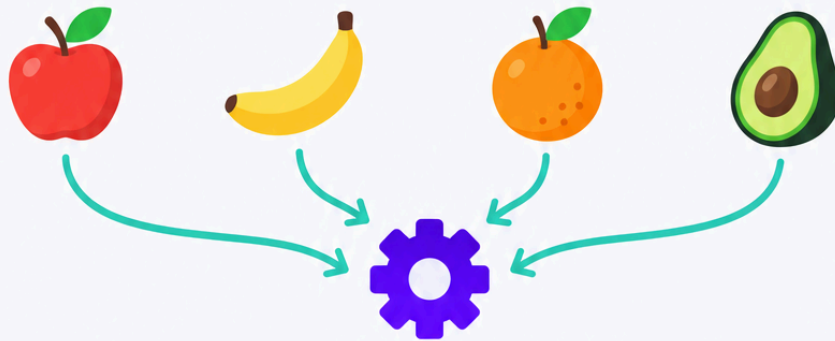


مختصر كتابي: forEach and Pure Function

التكرار باستخدام forEach

forEach and Pure Function



تنفذ الدالة مرة مع كل عنصر

```
fruits.forEach((item, index, arr) => {})
```

العنصر

رقم العنصر

المصفوفة الأصلية



السابق



الحالي



التالي

الدالة النقية

Pure Function

لا تغير شيئاً خارجها



تعمل في أي مشروع

سهولة الاختبار



اعتمد على المعاملات فقط

مختصر كتابي: forEach and Pure Function

كيف نمر على عناصر المصفوفة باستخدام forEach؟

ال **forEach** من أشهر طرق التكرار (Looping) في JavaScript. نعطيها **دالة استدعاء (Callback function)** فتنفذ **مرة واحدة مع كل item** في المصفوفة. ويمكن استخدامها مع ال array ومع ال NodeList التي نحصل عليها عند اختيار العناصر (select elements). وفي آخر الدرس نتعرف على معنى **الدالة النقية (Pure function)**.

أبسط استخدام ل forEach

```
const fruits = ["apple", "banana", "orange", "avocado"];

fruits.forEach((item) => console.log(`current fruit is ${item}`));
// current fruit is apple
// current fruit is banana
// current fruit is orange
// current fruit is avocado
```

سمّ المعامل **item** لو كنت تمر على array، و **element** لو كنت تمر على عناصر اخترتها من الصفحة. ولترى ما يحدث خطوة بخطوة ضع الكلمة **debugger** داخل الدالة: ستنتقل إلى تبويب Sources، وترى قيمة **item** تتغير من apple إلى banana ثم orange ثم avocado. هذه الطريقة مفيدة جدًا عندما تكون لديك حسابات معقدة ولا تفهم سبب النتيجة.

معاملات الدالة: item و index و array

المطلوب في التمرين: اطبع الفاكهة الحالية، والتي قبلها، والتي بعدها. لهذا نحتاج المعاملات الثلاثة التي تستقبلها دالة ال callback بالترتيب: **القيمة (value)** وهي ال item، ثم ال **index** الخاص به، ثم **المصفوفة الأصلية (original array)**:

```
const fruits = ["apple", "banana", "orange", "avocado"];

fruits.forEach((item, index, arr) => {
  if (arr[index - 1] === undefined) {
    console.log("This is the start of the array");
  } else {
    console.log(`previous fruit is ${arr[index - 1]}`);
  }

  console.log(`current fruit is ${item}`);

  if (arr[index + 1] === undefined) {
    console.log("This is the end of the array");
  } else {
    console.log(`next fruit is ${arr[index + 1]}`);
  }

  console.log("-----");
});
```

- `arr[index + 1]` يعطي العنصر التالي، و `arr[index - 1]` يعطي العنصر السابق.
- بدون الشرط يظهر `undefined` قبل أول عنصر وبعد آخر عنصر، لأنه لا يوجد شيء هناك. والرسالة الواضحة أفضل للمستخدم من `undefined`.
- انتبه: لو كتبت `item[index - 1]` بالخطأ ستحصل على حرف من الكلمة نفسها، لأن `item` هنا `string`.
- يمكنك استخدام `fruits` مباشرة بدل `arr` وستحصل على نفس النتيجة، لكن الأفضل الاعتماد على المعامل `arr`.
- رسالتا البداية والنهاية تحققان مطلوبين آخرين في التمرين: اطبع `hi` في البداية و `bye` في النهاية، فيكفي أن تضعهما مكان الرسالتين.

تحسين الكود (Refactoring)

لأن `undefined` قيمة `falsy`، نستطيع استبدال `=== undefined` بعلامة النفي `!`. ثم نستخدم `ال` **ternary operator** ليصبح كل شرط في سطر واحد، ونخرج الدالة في متغير مستقل اسمه `logFruit`:

```
const fruits = ["apple", "banana", "orange", "avocado"];

const logFruit = (item, index, arr) => {
  console.log(!arr[index - 1] ? "hi" : `previous fruit is ${arr[index - 1]}`);
  console.log(`current fruit is ${item}`);
  console.log(!arr[index + 1] ? "bye" : `next fruit is ${arr[index + 1]}`);
  console.log("-----");
};

fruits.forEach(logFruit);
```

بدلاً من نحو عشرين سطرًا صار الكود أربعة أسطر داخل الدالة. والنسختان صحيحتان، فاختر الأسهل في القراءة بالنسبة لك.

الدالة النقية (Pure function)

الدالة `logFruit` تعتمد على معاملاتها (inputs) فقط، ولا تغيّر أي شيء خارجها، أي **ليس لها أثر جانبي (Side effect)**. وهذه هي فكرة الدالة النقية. أما لو غيّرت الدالة متغيرًا موجودًا خارجها فلن تكون نقية:

```
let x = 10;

const changeX = () => {
  x = 12; // Pure تغيّر متغيرًا خارجها، فهي ليست
};

changeX();
console.log(x); // 12
```

- من الأمثلة المشهورة أيضًا: دالة تعدّل في الـ DOM، مثل إضافة عناصر أو حذفها أو استخدام `addEventListener`، فلها تأثير خارجها.
- من أكبر مزايا الدالة النقية أنها **سهولة الاختبار (test)**، ويمكنك نسخها إلى أي مشروع آخر فتعمل كما هي، لأنها غير مرتبطة بأي شيء خارجها.

الخلاصة

- `forEach` تنفّذ دالة الـ callback مرة واحدة مع كل عنصر، وتعمل مع الـ array والـ `NodeList`.
- الـ callback تستقبل بالترتيب: الـ item ثم الـ index ثم المصفوفة الأصلية.
- القيم `falsy` والـ `ternary` تجعل الكود أقصر وأوضح.
- الدالة النقية تعتمد على معاملاتها فقط ولا تغيّر شيئًا خارجها، فهي سهلة الاختبار وإعادة الاستخدام.

مصطلحات الدرس

المصطلح	بالإنجليزية	المعنى
دالة الاستدعاء	Callback function	دالة نمررها لـ <code>forEach</code> فتُنقَذ مع كل عنصر
المصفوفة الأصلية	Original array	المصفوفة التي نمر عليها، وهي المعامل الثالث
تحسين الكود	Refactoring	إعادة كتابة الكود ليصبح أقصر وأوضح بنفس النتيجة
الدالة النقية	Pure function	دالة تعتمد على معاملات فقط ولا تغيّر شيئاً خارجها
الأثر الجانبي	Side effect	أي تغيير تحدثه الدالة خارجها، مثل تعديل متغير أو الـ DOM