مختصر كتابي: Map, Find, Filter

التحويل والبحث والتصفية
Map, Find, Filter

map



تحول كل عنصر وترجع مصفوفة جديدة

filter



ترجع كل العناصر المطابقة

find



ترجع أول عنصر مطابق



```
employees.filter((employee) => employee.age >= 29)
```

الموظفون من 29 سنة فأكثر



ادمج العمليات: 3 دورات بدل 12



اختر الطريقة حسب ما تريد إرجاعه

مختصر كتابي: Map, Find, Filter

متى نستخدم map و find و filter؟

نكمل مع أشهر methods التكرار (Looping) في JavaScript. الـ **map** تحوّل كل عنصر وترجع **مصفوفة جديدة**، والـ **find** تبحث عن **عنصر واحد** يحقق شرطًا، والـ **filter** ترجع **كل العناصر** التي تحقق الشرط. والهدف ليس حفظ كل method وحدها، بل طريقة استخدامها معًا في الشغل اليومي.

map: مصنع يحوّل البيانات

الـ map مثل **المصنع (Factory)** أو المحوّل (Transformer): تعطيها المادة الخام فتحوّلها لك بالطريقة التي تحددها. أول مطلوب: أضف كلمة basha إلى كل اسم:

```
const people = ["mohamed", "ahmed", "hossam"];

const newNames = people.map((item) => `${item} basha`);
console.log(newNames); // [ 'mohamed basha', 'ahmed basha', 'hossam basha' ]
```

ما الفرق بينها وبين `forEach`؟ الـ map **ترجع نسخة جديدة** من المصفوفة، أما forEach فتمر على العناصر فقط **ولا ترجع شيئًا**. لذلك لتحصل على نفس النتيجة بـ forEach تحتاج كودًا أكثر:

```
const people = ["mohamed", "ahmed", "hossam"];

const newNames = [];
people.forEach((item) => {
  newNames.push(`${item} basha`);
});
console.log(newNames); // [ 'mohamed basha', 'ahmed basha', 'hossam basha' ]
```

سلسلة من التحويلات

المطلوب التالي: اجعل **أول حرف فقط** كبيرًا (Capital). لو استخدمنا `toUpperCase()` على الاسم كله ستصبح الكلمة كلها كبيرة، مثل `MOHAMED BASHA`. لذلك نأخذ أول حرف `name[0]` ونكبّره، ثم نضيف بقية الاسم بـ `slice(1)`، أي من الحرف رقم 1 حتى النهاية:

```
const capitalizedNames = newNames.map(
  (name) => `${name[0].toUpperCase()}${name.slice(1)}`
);
// [ 'Mohamed basha', 'Ahmed basha', 'Hossam basha' ]
```

ثم نحوّل كل اسم إلى object فيه `id` فريد (نستخدم الـ index) و `birthdate` . ولاحظ أمرين:

- عندما يكون اسم الخاصية واسم المتغير متطابقين نكتبه مرة واحدة: `name` بدل `name: name` .
- لترجع object من arrow function مختصرة ضعه بين قوسين `({ ... })` ، وإلا ستفهم JavaScript أن القوسين `{}` هما جسم الدالة.

```
const employees = capitalizedNames.map((name, index) => ({
  id: index,
  name,
  birthdate: 2000 + index,
}));

const employeesWithAge = employees.map((person) => {
  const currentYear = new Date().getFullYear();
  person.age = currentYear - person.birthdate;
  return person;
});
```

في الـ map الأخيرة استخدمنا الأقواس الكاملة `{}` ، وهنا **يجب أن تكتب return بنفسك**، وإلا ستحصل على مصفوفة من `undefined` . الإرجاع التلقائي (automatic return) يعمل فقط في الشكل المختصر.

دمج الحلول في map واحدة

الحل السابق مرّ على المصفوفة **4 مرات × 3 عناصر = 12 دورة**. لو جمعنا كل المطلوب في map واحدة تكفي 3 دورات فقط، والكود يصبح أسهل في القراءة والتعديل. هذه التفاصيل الصغيرة هي التي تفرّق بين مبرمج يكتب كودًا يعمل ومبرمج يكتب **كودًا نظيفًا (Clean code)**:

```
const people = ["mohamed", "ahmed", "hossam"];
const year = 1995;

const newEmployeesArray = people.map((name, index) => {
  const currentYear = new Date().getFullYear();
  const birthdate = year + index;

  return {
    id: index,
    name: `${name[0].toUpperCase()}${name.slice(1)} basha`,
    birthdate,
    age: currentYear - birthdate,
  };
});
```

وضعنا سنة البداية في المتغير `year` حتى نعدّلها من مكان واحد.

find للبحث عن عنصر واحد، وfilterg لكل المطابقين

نستخدم `find` عندما نريد عنصرًا واحدًا فقط، مثل موظف بـ ID معين. وبدل كتابة الرقم يدويًا كل مرة نضعها داخل function، ونرجع رسالة واضحة بدل `undefined` حتى يعرف المستخدم ما حدث:

```
const findEmployee = (id) => {
  const result = newEmployeesArray.find((employee) => employee.id === id);

  if (!result) {
    return "There is no employee with this ID";
  }
  return result;
};

console.log(findEmployee(1)); // employee object with id 1 (Ahmed basha)
console.log(findEmployee(3)); // There is no employee with this ID
```

أما `filter` فترجع مصفوفة بكل العناصر التي تحقق الشرط، مثل الموظفين الذين عمرهم 29 سنة أو أكثر، ومن عمره أقل من ذلك لن يظهر في النتيجة. (الأعمار تُحسب من السنة الحالية، لذلك تتغير النتيجة حسب سنة التشغيل.)

```
const employeesOlderThan29 = newEmployeesArray.filter(
  (employee) => employee.age >= 29
);
```

ويمكنك أيضًا **ربط الـ methods (Chaining)**، أي كتابتها وراء بعضها كحلقات السلسلة. مثلًا: `filter` للموظفين من 29 سنة فأكثر، ثم `find` داخلهم بشرط آخر. الطريقتان صحيحتان: الربط، أو فصل كل خطوة في متغير مستقل.

الخلاصة

- `map` تحوّل كل عنصر وترجع مصفوفة جديدة، أما `forEach` فلا ترجع شيئًا.
- مع الأقواس الكاملة `{}` اكتب `return` بنفسك، ولإرجاع object مباشرة ضعه بين `()`.
- `find` ترجع أول عنصر يطابق الشرط، و`filter` ترجع كل العناصر المطابقة.
- ادمج العمليات في loop واحدة متى أمكن لكود أسرع وأوضح.

مصطلحات الدرس

المصطلح	بالإنجليزية	المعنى
التحويل	map	تطبيق دالة على كل عنصر وإرجاع مصفوفة جديدة
البحث	find	إرجاع أول عنصر يحقق الشرط
التصفية	filter	إرجاع كل العناصر التي تحقق الشرط
الإرجاع التلقائي	Automatic return	إرجاع القيمة دون كتابة return في الـ arrow function المختصرة
الربط	Chaining	كتابة أكثر من method وراء بعضها
الكود النظيف	Clean code	كود سهل القراءة والتعديل ويهتم بالأداء