نقاط فنية و تفاصيل مهمة (Technical Details)

1. لماذا نتحقق من `isNaN`؟

إذا لم نتحقق من ذلك، فإن التعبير `(celsius * 9/5)` سيُنتج `NaN` إذا كانت قيمة `celsius` نفسها غير رقمية، مما يجعل الناتج النهائي غير قابل للقراءة من قبل المستخدم.

---

2. لماذا نستخدم `parseFloat` بدل `parseInt`؟

لأن `parseFloat` يدعم الأرقام العشرية (مثل `12.5`)، بينما `parseInt` يقص الجزء العشري ويحتفظ فقط بالعدد الصحيح.

---

3. ماذا لو المستخدم أدخل فراغات؟

- الدالة `parseFloat("  12.3  ")` تتجاهل الفراغات وتعطي `12.3`.
- لكن `parseFloat("")` أو `parseFloat("abc")` سترجع `NaN`.

---

4. لماذا نستخدم `const`؟

`const` تعني أن المتغير لن يُعاد تعيينه إلى مرجع جديد داخل نفس النطاق، وهو أمر جيد للثوابت أو القيم التي لا تتغير داخل الدالة. إذا كنت تحتاج تغيير القيمة لاحقًا، يمكنك استخدام `let` بدلاً من ذلك.

---

5. دعم الفاصلة كفاصل عشري

في بعض اللغات، يُكتب الرقم العشري باستخدام فاصلة `,` بدل النقطة `.` (مثل `12,5`). لكن `parseFloat` لا يتعرف على الفاصلة، لذلك يمكن استبدالها بنقطة قبل التحليل:

```
const raw = document.getElementById("celsiusInput").value.trim();
const normalized = raw.replace(',', '.');
const celsius = parseFloat(normalized);
```

---

6. عن `toFixed()`

الدالة `toFixed()` تُرجع قيمة نصية (string)، وليس رقمية. هذا مفيد عند العرض للمستخدم. ولكن إذا كنت تحتاج التعامل مع النتيجة حسابيًا لاحقًا، فيجب تحويلها مرة أخرى باستخدام `parseFloat()`.

---

7. الأمان وتجربة المستخدم

- يمكن إضافة خصائص مثل `min` و `max` إلى عنصر `<input type="number">` لتقييد القيم.
- من الأفضل استبدال خاصية `onclick` في الـ HTML بحدث Event Listener داخل الجافاسكربت لفصل المنطق عن البنية:

```
document.getElementById("convertBtn").addEventListener("click", convert);
```

أمثلة توضيحية سريعة

إدخال (C°)	النتيجة
25	`25°C = 77.00°F and 298.15K.`
0	`0°C = 32.00°F and 273.15K.`
-40	`-40°C = -40.00°F and 233.15K.` ← ملاحظة: عند `-40` تكون C° وF° متطابقتين.