Uso de métodos de Arrays con NodeList

📘 1️⃣ ¿Qué es un NodeList?

Un **NodeList** es una **colección de nodos del DOM** (elementos HTML), que obtenemos cuando usamos funciones como:

```
document.querySelectorAll("div");
document.getElementsByClassName("tarjeta");
document.getElementsByTagName("p");
```

📘 Ejemplo:

```
<div>Primero</div>
<div>Segundo</div>
<div>Tercero</div>
```

```
<script>
  const nodos = document.querySelectorAll("div");
  console.log(nodos);
</script>
```

🧩 En consola verás algo como:

NodeList(3) [div, div, div]

🧩 2️⃣ ¿En qué se parece un NodeList a un Array?

✅ **Similitudes:**

- Se puede **recorrer con un bucle for o for...of**.
- Tiene la **propiedad .length**.
- Es **iterable** (se puede usar en forEach() en la mayoría de navegadores modernos).

❌ **Diferencias:**

- **No es un array real**, por lo que **no tiene todos los métodos de los arrays**.
- No puedes usar directamente map(), filter(), reduce(), join(), etc.

⚙️ 3️⃣ Métodos que sí puedes usar directamente en un NodeList

Método	Disponible	Descripción
forEach()	✅ Sí	Recorre todos los nodos
length	✅ Sí	Devuelve el número de nodos
item()	✅ Sí	Devuelve el nodo de un índice
entries(), keys(), values()	⚠️ A veces	Solo en navegadores modernos
map(), filter(), reduce(), join()	❌ No	Solo funcionan en arrays

---

💡 Ejemplo práctico de uso permitido:

```
const divs = document.querySelectorAll("div");

divs.forEach(d => d.style.color = "blue");
console.log(divs.length); // cantidad de divs
```

✅ Funciona perfecto.
Pero si intentas hacer:

divs.map(d => d.textContent);

❌ Te dará error:

TypeError: divs.map is not a function

🧩 4️⃣ Cómo convertir un NodeList en un Array real

Para poder usar **todos los métodos de arrays** (map, filter, join, etc.),
debemos **convertir el NodeList a un array real**.

🔷 Método 1: Array.from()

```
const nodos = document.querySelectorAll("div");
const arrayDivs = Array.from(nodos);
console.log(arrayDivs.map(d => d.textContent));
```

📘 Array.from() toma una colección iterable (como NodeList) y crea un **array real**.

🔷 Método 2: Operador Spread [...]

```
const nodos = document.querySelectorAll("div");
const arrayDivs = [...nodos];
console.log(arrayDivs.map(d => d.textContent));
```

💡 El **spread operator** (...) copia los elementos del NodeList dentro de un array nuevo.
Es la forma más moderna y limpia.

🧠 5️⃣ Ejemplo completo de uso combinado

Supongamos que tienes varios productos en el DOM:

```
<ul>
  <li class="producto">Teclado</li>
  <li class="producto">Ratón</li>
  <li class="producto">Monitor</li>
</ul>

<script>
  // 1️⃣ Capturar los elementos
  const nodos = document.querySelectorAll(".producto");

  // 2️⃣ Convertir a array
  const productos = [...nodos].map(nodo => nodo.textContent);

  // 3️⃣ Usar métodos de arrays
  const lista = productos.join(" | ");

</script>
  console.log(lista); // "Teclado | Ratón | Monitor"
</script>
```

💡 Aquí usas:

- querySelectorAll() → obtiene un NodeList.
- map() → aplicado tras convertirlo a array.
- join() → para unir los textos en una sola cadena.

🧩 6️⃣ Otro ejemplo — Filtrar nodos según su contenido

```
<p>JavaScript</p>
```

```
<p>Python</p>
<p>HTML</p>
<p>CSS</p>

<script>
  const parrafos = [...document.querySelectorAll("p")];

  const filtrados = parrafos.filter(p => p.textContent.includes("J"));
  filtrados.forEach(p => p.style.color = "red");
</script>
```

📘 Resultado:
Solo los párrafos que contienen “J” (JavaScript) se pondrán en rojo.

🧠 7️⃣ Métodos útiles una vez convertido el NodeList

Método	Qué hace	Ejemplo
map()	Transformar nodos en valores	[...nodos].map(n => n.textContent)
filter()	Filtrar elementos del DOM	[...nodos].filter(n => n.textContent.includes("a"))
reduce()	Combinar valores de nodos	[...nodos].reduce((t, n) => t + n.textContent, "")
join()	Unir el texto de los nodos	[...nodos].map(n => n.textContent).join(", ")
sort()	Ordenar elementos del DOM (por texto o valor)	[...nodos].sort((a,b) => a.textContent.localeCompare(b.textContent))

⚠️ 8️⃣ Diferencia con HTMLCollection

👉 Métodos como getElementsByClassName() o getElementsByTagName() devuelven un **HTMLCollection**, no un NodeList.

Propiedad	NodeList	HTMLCollection
Iterador (for...of)	✅ Sí	✅ Sí
forEach()	✅ Sí	❌ No
length	✅ Sí	✅ Sí
Métodos de Array (map, filter)	❌ No	❌ No

📘 **Conclusión:**
Tanto NodeList como HTMLCollection **deben convertirse a Array** si quieres usar métodos avanzados.

🧩 9️⃣ Ejercicio práctico para tus alumnos

👉 **Objetivo:** recorrer elementos del DOM, filtrarlos, transformarlos y combinarlos con join().

```html
<ul>
  <li class="producto">Teclado</li>
  <li class="producto">Ratón</li>
  <li class="producto">Monitor</li>
  <li class="producto">Auriculares</li>
</ul>

<script>
  // 1 Capturar NodeList
  const nodos = document.querySelectorAll(".producto");

  // 2 Convertir a array real
  const productos = Array.from(nodos);

  // 3 Filtrar los que contienen la letra "o"
  const filtrados = productos.filter(p => p.textContent.includes("o"));

  // 4 Crear texto combinado
  const texto = filtrados.map(p => p.textContent).join(" - ");

  // 5 Mostrar el resultado
  console.log("Productos filtrados:", texto);
</script>
```

📘 **Salida:**

Productos filtrados: Teclado - Ratón - Monitor - Auriculares

🎯 **10 Conclusión**

Pregunta	Respuesta
¿Un NodeList es un array?	❌ No, pero se parece
¿Qué métodos puedo usar directamente?	✅ forEach(), length
¿Cómo uso map(), filter() o join()?	🔷 Convirtiendo a array con Array.from() o [...nodelist]
¿Para qué sirve?	Para manipular listas de elementos del DOM fácilmente

---

💡 **Regla de oro para tus alumnos:**
"Si quieres usar métodos de arrays (map, filter, join, reduce) sobre elementos del DOM, primero conviértelos a array real."