Tema 3 Objetos Nativos en Javascript

Practica 03-06b getElementById() vs querySelector

Nombre		Curso	
Apellidos		Fecha	

Document.querySelector()

Devuelve el primer elemento del documento (utilizando un recorrido primero en profundidad pre ordenado de los nodos del documento) que coincida con el grupo especificado de selectores.

Sintaxis

element = document.querySelector(selectores);

Donde:

- element es un objeto de tipo element.
- selectores es una cadena de caracteres que contiene uno o más selectores CSS separados por coma.

Ejemplo

En este ejemplo, obtendremos el primer elemento del documento con la clase "miClase":

```
var el = document.querySelector(".miClase");
```

Ejemplo más útil

Los Selectores pueden ser muy útiles como se demostrará en el siguiente ejemplo. Aquí, será retornado el primer elemento <input name="login" /> dentro de .

```
var el = document.querySelector("div.user-panel.main input[name='login']");
```

Notas

- Devuelve null si no se encuentran coincidencias, de lo contrario, retorna el primer elemento encontrado.
- Si el selector coincide con un ID y este ID es usado erróneamente varias veces en el documento, devuelve el primer elemento encontrado.
- Lanza una excepción de tipo SYNTAX_ERR si el grupo de selectores especificado no es válido.

- querySelector() se introdujo en la API Selectors.
- La cadena de caracteres que se pasa como argumento a querySelector debe seguir la sintaxis CSS.
- Las Pseudo-clases CSS nunca devolverán elementos, tal y como está especificado en la API Selectors.
- Para que coincidan ID's o selectores que no siguen la sintáxis CSS (usando inapropiadamente dos puntos o un espacio por ejemplo), se debe 'escapar' el carácter con una barra invertida (\). Como la barra invertida es un carácter de 'escape' en JavaScript, si estás indicando una cadena de caracteres literal, debes 'escaparla' dos veces (una para la cadena de caracteres JavaScript y otra para el querySelector):

```
<div id="foo\bar"></div>

<div id="foo:bar"></div>


<script>
  console.log("#foo\bar"); // "#fooar"
  document.querySelector("#foo\bar"); // No coincide con nada
  console.log("#foo\\bar"); // "#foo\bar"
  console.log("#foo\\\\bar"); // "#foo\\bar"
  document.querySelector("#foo\\\\bar"); // Coincide con el primer div


  document.querySelector("#foo:bar"); // No coincide con nada
  document.querySelector("#foo\\:bar"); // Coincide con el segundo div
</script>
```

getElementById() vs querySelector

La eleccion depende de qué necesites hacer y cómo estés seleccionando elementos.

document.getElementById()

Uso:

```
const elemento = document.getElementById('miId');
```

Características:

- Selecciona **solo un elemento** del DOM (el que tenga el ID indicado).
- Es **muy rápido**, ya que los navegadores están optimizados para buscar por id.
- Solo sirve para elementos con atributo id.
- No requiere el # al pasar el nombre del ID.

Ventajas:

- **Rendimiento superior** (más rápido que querySelector).
- Muy claro y directo cuando sabes el id del elemento.
- No sirve para clases, etiquetas u otros selectores.

Cuándo usarlo:
Cuando sabes exactamente el id del elemento que necesitas y solo necesitas uno, por ejemplo:

```
const boton = document.getElementById('btnEnviar');
```

document.querySelector()

Uso:

```
const elemento = document.querySelector('#miId');
const otro = document.querySelector('.miClase');
```

Características:

- Permite usar **selectores CSS completos** (#id, .clase, div > p, [name="email"], etc.).
- Devuelve **solo el primer elemento** que coincide con el selector.
- Es más **versátil**, aunque un poco más **lento** (aunque en la práctica la diferencia es mínima en la mayoría de casos).

Ventajas:

- Muy flexible (acepta cualquier selector CSS).
- Permite buscar dentro de un contenedor, no solo en document.
- Ligeramente más lento (pero casi imperceptible en proyectos pequeños/medianos).
- Puede ser menos claro si abusas de selectores complejos.

Cuándo usarlo:
Cuando necesitas seleccionar por **clase, etiqueta o combinaciones CSS**, por ejemplo:

```
const boton = document.querySelector('.btn.primario');
const input = document.querySelector('input[name="email"]');
```

En resumen:

Criterio	getElementById()	querySelector()
Selector	Solo por ID	Cualquier selector CSS
Retorna	1 elemento	1er elemento que coincide
Velocidad	Más rápida	Un poco más lenta
Flexibilidad	Limitada	Muy alta
Sintaxis	getElementById('id')	querySelector('#id')

Recomendación práctica:

- Usa **getElementById()** si solo necesitas un elemento por ID → más simple y eficiente.
- Usa **querySelector()** si quieres flexibilidad o si tu selector no es un id.

Y si necesitas **más de un elemento**, entonces usa querySelectorAll():

```
const botones = document.querySelectorAll('.btn');
```

¿Quieres que te muestre un **ejemplo comparativo en código** mostrando el rendimiento o uso combinado de ambos?