1. REPLACECHILDREN

El método `replaceChildren` en JavaScript es una forma eficiente de reemplazar todos los nodos hijos de un elemento con nuevos nodos o contenido. Fue introducido en el DOM como una alternativa más clara y directa a otras técnicas como eliminar manualmente los nodos con `removeChild` o sobrescribir con `innerHTML`.

¿Qué hace replaceChildren?

1. Elimina todos los nodos hijos existentes del elemento.
2. Agrega nuevos nodos o texto opcionales que le pases como argumentos.

Si no se proporcionan argumentos, simplemente vacía el contenido del elemento.

Sintaxis:

```
element.replaceChildren(nodes);
```

Parámetros:

- `nodes`: Uno o más nodos o cadenas de texto que serán agregados como hijos. Si pasas una cadena, se convierte automáticamente en un nodo de texto.

Ejemplo 1: Reemplazar hijos existentes con texto:

```
const div = document.getElementById('miDiv');
div.replaceChildren(document.createTextNode('Nuevo contenido'));
```

Antes:

```
<div id="miDiv">
    <p>Contenido viejo</p>
    <span>Más contenido</span>
</div>
```

Después:

```
<div id="miDiv">Nuevo contenido</div>
```

Ejemplo 2: Reemplazar con nuevos elementos:

```
const div = document.getElementById('miDiv');
const nuevoParrafo = document.createElement('p');
nuevoParrafo.textContent = 'Este es un párrafo nuevo';
div.replaceChildren(nuevoParrafo);
```

Antes:

```
html
<div id="miDiv">
```

```
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>
</div>
```

Después:

```
<div id="miDiv">
    <p>Este es un párrafo nuevo</p>
</div>
```

Ejemplo 3: Vaciar un elemento:

```
const div = document.getElementById('miDiv');
div.replaceChildren(); // Elimina todos los hijos sin agregar nada
nuevo
```

Resultado:

```
<div id="miDiv"></div>
```

Equivalente con RemoveChild:

```
const div = document.getElementById("miDiv");
// Mientras el contenedor tenga hijos, elimina el primero.
 while (div.firstChild) { div.removeChild(div.firstChild); }
```

Ejemplo 4: Reemplazar por varios elementos:

```
const div = document.getElementById('miDiv');

// Crear nuevos nodos
const parrafo1 = document.createElement('p');
parrafo1.textContent = 'Este es el primer párrafo';

const parrafo2 = document.createElement('p');
parrafo2.textContent = 'Este es el segundo párrafo';

const texto = document.createTextNode('Texto adicional sin
envoltura');

// Reemplazar hijos existentes con varios nuevos
div.replaceChildren(parrafo1, parrafo2, texto);
```

HTML Antes:

```
<div id="miDiv">
    <ul>
        <li>Elemento antiguo 1</li>
        <li>Elemento antiguo 2</li>
    </ul>
</div>
```

HTML Después:

```
<div id="miDiv">
```

```
    <p>Este es el primer párrafo</p>
    <p>Este es el segundo párrafo</p>
    Texto adicional sin envoltura
</div>
```

Ventajas de replaceChildren:

1. **Eficiencia:** Realiza la limpieza y la adición de nuevos nodos en una única operación.
2. **Simplicidad:** Código más claro comparado con `innerHTML` o bucles con `removeChild`.
3. **Seguridad:** No ejecuta código HTML ni interpreta contenido como lo hace `innerHTML`. Evita que se inyecte un script malicioso en esta entrada
4. **Ineficiencia**: genera nuevamente el árbol DOM del contenido del elemento que se está manipulando y se pierden los eventos asociados

---

¿Cuándo usarlo?

- Cuando necesitas actualizar completamente el contenido de un elemento.
- Cuando buscas una alternativa más eficiente y moderna a `innerHTML` para reemplazar o vaciar hijos de un nodo.

2. TEXTCONTENT

El método o propiedad **`textContent`** en JavaScript se utiliza para obtener o establecer el contenido de texto de un nodo en el DOM. A diferencia de `innerHTML`, `textContent` no interpreta el contenido como HTML, lo que lo hace más seguro y adecuado para trabajar con texto plano.

Características principales de `textContent`:

1. **Obtiene texto plano:** Al leer `textContent`, se obtiene todo el texto contenido en un elemento, incluidos sus nodos hijos, sin incluir etiquetas HTML.

   Ejemplo:

   ```
   <div id="miDiv">
       <p>Hola <strong>mundo</strong></p>
   </div>
   javascript
   Copiar código
   const div = document.getElementById('miDiv');
   console.log(div.textContent); // "Hola mundo"
   ```

   Las etiquetas `<p>` y `<strong>` no se incluyen, solo el texto.

---

2. **Establece texto plano:** Al asignar un valor a `textContent`, todo el contenido existente del elemento se reemplaza con el texto proporcionado. Cualquier HTML dentro del elemento será tratado como texto literal, no como código ejecutable.

   Ejemplo:

   ```
   const div = document.getElementById('miDiv');
   div.textContent = "<b>Texto plano</b>";
   console.log(div.innerHTML); // "&lt;b&gt;Texto plano&lt;/b&gt;"
   ```

   El texto `<b>Texto plano</b>` aparece literalmente en lugar de interpretarse como HTML.

Ventajas de textContent sobre innerHTML:

1. **Mayor seguridad:** Como `textContent` no ejecuta ni interpreta código HTML o JavaScript, es inmune a ataques de inyección de código (como XSS).

2. **Más rápido:** `textContent` es más eficiente que `innerHTML` cuando solo se necesita trabajar con texto, ya que no requiere analizar ni construir nodos HTML.

3. **Sencillez:** Es ideal para limpiar el contenido de un elemento, ya que reemplaza todo con texto plano sin preocuparse por etiquetas o atributos.