JavaScript (ES6) manipular DOM

Diego Muñoz

9 de septiembre de 2025

Introducción

- Repaso de JS moderno con interacción mínima en el DOM.
- Practicar funciones, closures, arrays y eventos.
- Objetivo: jugar en HTML y reforzar JS.

1. ¿Qué es el DOM?

- **DOM = Document Object Model**.
- Representación en memoria de la página web.
- Cada etiqueta HTML es un **nodo/objeto** manipulable con JS.
- Permite: leer/modificar texto, atributos y reaccionar a eventos.

```
<h1 id="titulo">Hola</h1>
<script>
  const t = document.getElementById("titulo");
  console.log(t.textContent); // "Hola"
</script>
```

2. DOM como árbol

```
<body>
  <h1 id="titulo">Hola</h1>
  <p class="texto">Texto de ejemplo</p>
</body>
```

Se transforma en nodos:

```
└─ body
     ├─ h1#titulo
     └─ p.texto
```

3. Operaciones básicas

- Seleccionar: `getElementById`, `querySelector`.
- Texto: `el.textContent`.
- Inputs: `el.value`.
- Eventos: `addEventListener`.

```
const el = document.querySelector("p");
el.textContent = "Nuevo contenido";
```

4. Modificar texto

```
<div id="msg">Hola</div>
<script>
  const msg = document.getElementById("msg");
  msg.textContent = "Hola JS + DOM";
</script>
```

5. Inputs y eventos

```
<input id="name" placeholder="Nombre">
<button id="say">Saludar</button>
<output id="out"></output>

<script>
  const $ = s => document.querySelector(s);
  $("#say").addEventListener("click", () => {
    const name = $("#name").value.trim();
    $("#out").textContent = name ? "Hola " + name : "Falta nombre";
  });
</script>
```

6. Closures repaso

```
function makeCounter(initial=0){
  let value = initial;
  return {
    next: () => ++value,
    reset: () => (value = initial),
    get: () => value
  };
}
```

7. Ejemplo A: contador con DOM

```
<button id="inc">Sumar</button>
<button id="reset">Reset</button>
<output id="val">0</output>

<script>
  const c = makeCounter(0);
  const $ = s => document.querySelector(s);
  const render = () => { $("#val").textContent = c.get(); };

  $("#inc").addEventListener("click", () => { c.next(); render(); });
  $("#reset").addEventListener("click", () => { c.reset(); render(); });
  render();
</script>
```

8. Funciones puras para render

```
const Card = ({ title, body }) =>
  "<div class='card'><h3>" + title + "</h3><p>" + body + "</p></div>";
```

En React esto será JSX. Aquí strings.

9. Lista de tareas (modelo)

```
let tasks = [
  { id: 1, text: "Leer", done: false },
  { id: 2, text: "Practicar JS", done: true }
];

function addTask(list, text){
  const id = list.length ? Math.max(...list.map(t => t.id)) + 1 : 1;
  return [...list, { id, text, done:false }];
}

function toggleTask(list, id){
  return list.map(t => t.id === id ? { ...t, done: !t.done } : t);
}
```

10. Lista de tareas (HTML base)

```html
<form id="form">
  <input id="newText" placeholder="Nueva tarea">
  <button type="submit">Agregar</button>
</form>

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

11. Lista de tareas (render)

```
const $ = s => document.querySelector(s);

function TaskItem({ id, text, done }){
 const mark = done ? "[x]" : "[ ]";
 return "<div data-id='"+id+"' class='item'>" + mark + " " + text + "</div>";
}

function renderList(list){
  $("#list").innerHTML = list.map(TaskItem).join("");
}
```

12. Lista de tareas (interacción)

```
$("#form").addEventListener("submit", (e) => {
  e.preventDefault();
  const text = $("#newText").value.trim();
  if(!text) return;
  tasks = addTask(tasks, text);
  $("#newText").value = "";
  renderList(tasks);
});
```

12. Lista de tareas (interacción)

```
$("#list").addEventListener("click", (e) => {
  const item = e.target.closest(".item");
  if(!item) return;
  const id = Number(item.dataset.id);
  tasks = toggleTask(tasks, id);
  renderList(tasks);
});

renderList(tasks);
```

13. Estilos básicos

```
<style>
  .item { padding: 6px; border-bottom: 1px solid #eee; cursor: pointer; }
  .item.done { color: #777; text-decoration: line-through; }
</style>
```

14. Render condicional

```html
<label>
  <input type="checkbox" id="onlyActive">
  Mostrar solo pendientes
</label>
```

```js
$("#onlyActive").addEventListener("input", () => {
  const only = $("#onlyActive").checked;
  const view = only ? tasks.filter(t => !t.done) : tasks;
  renderList(view);
});
```

15. Buenas prácticas

1. Funciones puras para formateo y estado.
2. No mutar arrays/objetos originales.
3. Delegación de eventos para listas dinámicas.
4. Separar render de la lógica de datos.
5. Mantener código simple y legible.

16. Resumen

- DOM es el modelo de la página como árbol de objetos.
- Podemos seleccionar nodos, leer/escribir, escuchar eventos.
- Practicamos closures, arrays funcionales e inmutabilidad.
- Simulamos estado y render, igual que React lo hará.