Javascript (DOM #02)

> **¿Quieres apoyar a los directores?** 😍
>
> Tienes varias jugosas alternativas:
>
> 1. Suscríbete al canal de Youtube (es gratis) click aquí
> 2. Si estás viendo un video no olvides regalar un 👍 me gusta y comentario 🙏🏽
> 3. También puedes ser miembro del canal de Youtube click aquí
> 4. Puedes adquirir cursos premium en Udemy 👇🏽👇🏽👇🏽 ¿Quiéres apoyar a los directores?
>   - Curso de HTML + CSS + Bootstrap 5 + Git y más UDEMY
>   - Curso de React + Firebase UDEMY
>   - Curso Vue.js + Firebase UDEMY

El burbujeo y la captura

- eventos
- agregarListener de eventos

El **burbujeo y la captura** de eventos son dos mecanismos que describen lo que sucede cuando dos controladores del mismo tipo de evento se activan en un elemento.

```html
<div class="container">
    <div class="border border-primary border-5 py-5 m-3">
        Lorem, ipsum dolor
        <div class="border border-secondary border-5 py-5 m-3">
            Lorem, ipsum dolor
            <div class="border border-danger border-5 py-5 m-3">
                Lorem, ipsum dolor
            </div>
        </div>
    </div>
</div>
```

```js
const primary = document.querySelector(".border-primary");
const secondary = document.querySelector(".border-secondary");
const danger = document.querySelector(".border-danger");

primary.addEventListener("click", (e) => console.log("primary"));
secondary.addEventListener("click", (e) => console.log("secondary"));
danger.addEventListener("click", (e) => console.log("danger"));
```

Fase de captura: Se propaga desde el elemento padre hasta el hijo.

```js
primary.addEventListener("click", (e) => console.log("primary"), true);
secondary.addEventListener("click", (e) => console.log("secondary"), true);
danger.addEventListener("click", (e) => console.log("danger"), true);
```

detener la propagación

- detener la propagación : evita la propagación adicional del evento actual en las fases de captura y burbujeo.

```js
const cajas = document.querySelectorAll(".container div");
cajas.forEach((item) => {
    item.addEventListener("click", (e) => {
        e.stopPropagation();
        console.log("click");
    });
});
```

prevenirPredeterminado

- prevenirPredeterminado : Cancela el evento si este es cancelable, sin detener el resto del funcionamiento del evento, es decir, puede ser llamado de nuevo.

```html
<form>
    <input type="text" name="nombre">
```

```js
const formulario = document.querySelector("form");
formulario.addEventListener("submit", (e) => {
    e.preventDefault();
    console.log("click");
});
```

> **CONSEJO**
>
> En las siguientes secciones veremos un especial de formularios con sus respectivas validaciones.

Sirve para cualquier comportamiento por defecto del navegador:

```html
<a href="#">ancla</a>
```

```js
const ancla = document.querySelector("a");
ancla.addEventListener("click", (e) => e.preventDefault());
```

Delegación de Eventos

La delegación de eventos es básicamente un patrón para gestionar eventos de manera eficiente.

En lugar de agregar un detector de eventos a todos y cada uno de los elementos similares, podemos agregar un detector de eventos a un elemento principal y llamar a un evento en un objetivo en particular utilizando la propiedad .target del objeto de evento.

Así evitamos la propagación 👌

```html
<div class="container">
    <div
      id="padre"
      class="border border-primary border-5 py-5 m-3"
      data-div="divPadre"
```

```
            id="hijo"
            class="border border-secondary border-5 py-5 m-3"
            data-div="divHijo"
          >
            Lorem, ipsum.
            <div
              id="nieto"
              class="border border-danger border-5 py-5 m-3"
              data-div="divNieto"
            >
              Lorem, ipsum.
            </div>
          </div>
        </div>
      </div>
```

```
const container = document.querySelector(".container");
container.addEventListener("click", (e) => {
    console.log(e.target);
});
```

¿Pero como activo un evento para un elemento en específico?

- partidos : El métodomatches() comprueba si el elemento sería seleccionable por el selector CSS especificado en la cadena; en caso contrario, retornará falso.
- conjunto de datos

```
const container = document.querySelector(".container");
container.addEventListener("click", (e) => {

    // console.log(e.target.id);
    if (e.target.id === "hijo") {
        console.log("diste click en el mijo");
    }

    // console.log(e.target.matches(".border-danger"));
    if (e.target.matches(".border-danger")) {
        console.log("diste click en el nieto");
    }
```

```
    // console.log(e.target.dataset.div);
    if (e.target.dataset["div"] === "divPadre") {
        console.log("diste click en padre");
    }
});
```

> **CONSEJO**
>
> Puedes seleccionar todo el documento, así no tienes que estar detectando el componente principal 💪

```js
document.addEventListener()
```

Práctica

- ver ejemplo

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Carrito Objeto</title>
    <link crossorigin="anonymous" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2Qv2
</head>

<body>

    <main class="container mt-5">
        <div class="row text-center">
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Frutilla 🍓</h5>
                        <p class="lead">$300</p>
                        <button class="btn btn-primary btn-sm" data-fruta="frutilla"
```

HTML, CSS y Bootstrap 5 (bluuweb)

```
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Banana 🍌</h5>
                        <p class="lead">$100</p>
                        <button class="btn btn-primary btn-sm" data-fruta="banana" da
                    </div>
                </div>
            </article>
            <article class="col-sm-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Manzana 🍏</h5>
                        <p class="lead">$200</p>
                        <button class="btn btn-primary btn-sm" data-fruta="manzana" d
                    </div>
                </div>
            </article>
        </div>
    </main>

    <section class="container mt-3">
        <ul class="list-group" id="carrito"></ul>
    </section>

    <footer id="footer" class="container mt-3">
        <template id="templateFooter">
            <div class="card">
                <div class="card-body d-flex justify-content-between align-items-cent
                    <p class="lead mb-0">TOTAL: $<span>1500</span></p>
                    <button class="btn btn-outline-primary">Finalizar Compra</button>
                </div>
            </div>
        </template>
    </footer>

    <template id="template">
        <li class="list-group-item text-uppercase bg-secondary text-white">
            <span class="badge bg-primary rounded-pill align-middle">14</span>
            <span class="lead align-middle">A list item</span>
        </li>
        <li class="list-group-item d-flex justify-content-between align-items-center
            <div>
```

HTML, CSS y Bootstrap 5 (bluuweb)

```
                <div>
                    <button class="btn btn-sm btn-success">Agregar</button>
                    <button class="btn btn-sm btn-danger">Quitar</button>
                </div>
            </li>
        </template>

        <script src="app.js"></script>
    </body>

    </html>
```

js

```
const carrito = document.querySelector("#carrito");
const template = document.querySelector("#template");
const footer = document.querySelector("#footer");
const templateFooter = document.querySelector("#templateFooter");
const fragment = document.createDocumentFragment();
let carritoArray = [];

// Delegación de eventos:
document.addEventListener("click", (e) => {
    // console.log(e);
    // console.log(e.target.dataset.fruta);
    // console.log(e.target.matches(".card button"));
    if (e.target.matches(".card button")) {
        agregarCarrito(e);
    }

    // console.log(e.target.matches(".list-group-item .btn-success"));
    if (e.target.matches(".list-group-item .btn-success")) {
        btnAumentar(e);
    }

    // console.log(e.target.matches(".list-group-item .btn-danger"));
    if (e.target.matches(".list-group-item .btn-danger")) {
        btnDisminuir(e);
    }
});

const agregarCarrito = (e) => {
    // console.log(e.target.dataset);
    const producto = {
```

```
            cantidad: 1,
            precio: parseInt(e.target.dataset.precio),
        };

        // buscamos el indice
        const index = carritoArray.findIndex((item) => item.id === producto.id);

        // si no existe empujamos el nuevo elemento
        if (index === -1) {
            carritoArray.push(producto);
        } else {
            // en caso contrario aumentamos su cantidad
            carritoArray[index].cantidad++;
        }

        // console.log(carritoArray);
        pintarCarrito();
    };

    const pintarCarrito = () => {
        carrito.textContent = "";

        // recorremos el carrito y pintamos elementos:
        carritoArray.forEach((item) => {
            const clone = template.content.cloneNode(true);
            clone.querySelector(".text-white .lead").textContent = item.titulo;
            clone.querySelector(".rounded-pill").textContent = item.cantidad;
            clone.querySelector("div .lead span").textContent =
                item.precio * item.cantidad;
            clone.querySelector(".btn-success").dataset.id = item.id;
            clone.querySelector(".btn-danger").dataset.id = item.id;
            fragment.appendChild(clone);
        });
        carrito.appendChild(fragment);

        pintarFooter();
    };

    const pintarFooter = () => {
        footer.textContent = "";

        const total = carritoArray.reduce(
            (acc, current) => acc + current.precio * current.cantidad,
            0
```

```
    // console.log(total);

    const clone = templateFooter.content.cloneNode(true);
    clone.querySelector("p span").textContent = total;

    // fragment.appendChild(clone);
    footer.appendChild(clone);
};

const btnAumentar = (e) => {
    // console.log(e.target.dataset.id);
    carritoArray = carritoArray.map((item) => {
        if (item.id === e.target.dataset.id) {
            item.cantidad++;
        }
        return item;
    });
    pintarCarrito();
};

const btnDisminuir = (e) => {
    // console.log(e.target.dataset.id);
    carritoArray = carritoArray.filter((item) => {
        // console.log(item);
        if (item.id === e.target.dataset.id) {
            if (item.cantidad > 0) {
                item.cantidad--;
                // console.log(item);
                if (item.cantidad === 0) return;
                return item;
            }
        } else {
            return item;
        }
    });
    pintarCarrito();
};
```

Última actualización: 27/11/2021, 13:07:12