JAVASCRIPT AVANZADO

NOELIA CÁMARA

2024-2025

Contenido

EVENTOS 4

EVENTOS EN LINEA 4

EVENTOS TRADICIONAL 5

EVENTOS W3C 6

EVENTOS INFORMACIÓN 6

EJERCICIOS 9

WebStorage 12

LOCALSTORAGE 12

SESIONSTORAGE 12

DOM 13

TRAVERSING 13

TIPOS DE NODOS 13

1a PARTE MOVERSE VERTICALMENTE 13

childNodes 13

children 14

firstChild 14

FirstElementChild 14

RECORRER EL ÚLTIMO NODO 14

LastChild 14

LastElementChild 15

HasChildrenNodes() 15

2a PARTE MOVERSER HORIZONTALMENTE 15

3a PARTE 15

getElementByID() 15

getElementByClassName() 15

getElementsByTagName() 15

getElementsByName() 15

querySelector() 15

querySelectorAll() 16

Diferencias entre nodeList y HTMLCollection 16

nodeList 16

HTMLCollection 16

4a PARTE 17

console.dir 17

className 17
nodeName 17
innerHTML 17
outerHTML 17
textContent 17
5ª PARTE 17
setAttribute() 17
getAttribute() 17
hasAttribute() 18
removeAttribute() 18
Eventos DOM 18
EVENT BUBBLING O PROPAGACIÓN DE EVENTOS 18
FASE DE CAPTURA 19
matches() y closest() 20
EVENT DELEGATION 21
CLASES 21
classList 21
método item() 21
método add() 21
método remove() 21
método toggle() 22
método contains() 22
método replace() 22
CREATE ELEMENT 22
createElement() 23
appendChild() 23
append() 23
prepend() 23
style 24
property () 24
getPropertyValue() 25
getComputedStyle () 25

EVENTOS

```
 // EVENTO mecanismo que se acciona cuando el usuario realiza un cambio sobre
la pagina web.Captura el evento es programar una accion sobre un elemento. Se
encarga de gestionarlo el DOM.

        // MANEJADOR: la acción que se va a manejar. Por ejemplo el hacer
click.
        // Manejadores de eventos del raton
        // - onclick
        // - ondbclick
        // - onmousedown/up
        // - onmouseover/onmouseout
        // - onmousemove

        // Manejadores de eventos de teclado
        // - onkeydown: cuando de pulsa o se mantiene pulsada una tecla
        // - onkeypress:
        // - onkeyup: cuando se suelta la tecla
        // https://www.w3schools.com/jsref/dom_obj_event.asp




        // El nombre del evento sería "on" más algo (pe onclick)

<!-- Para ejecutar los script es mejor tener cargada la parte del DOM, por eso
es mejor ponerlo al final del body el script  -->
   <script src="eventosLinea.js"> </script>
```

EVENTOS EN LINEA

NO HACER NUNC EVENTOS EN LINEA

```
<!-- NO HACER NUNCA  EVENTOS EN LINEA -->
    <!-- Manejadores como atributo -->
    <h3 id="titulo" onclick="this.innerHTML='Hola'"
onmouseover="this.style.background='red'">Pulsa aquí</h3>
    <h3 id="titulo2"
onclick="document.getElementById('titulo2').innerHTML='Adios'">Hasta
luego</h3>
    <!-- Manejadores con funciones externas -->
     <h3 onclick="cambiar(this)">Pulsa aquí</h3>
     <script>
        function cambiar(elem){
            elem.innerHTML = "Funcion externa";
        };
     </script>
```

```
    <!-- Existen acciones que desencadenan varios eventos.Ejemplo submit que
desencadena varios manejadores.
     Para evitar que se desencadenen por defecto un manejador, en el ejemplo
anterior evitamos que vaya a google -->
     <a href="https://www.google.es" onclick="alertar(); return
false">Google</a> <br><br>
     <script>
       function alertar(){
           alert("Vamos a google");
       }
     </script>
     <a href="https://www.google.es" onclick=" return preguntar()">Preguntar
al usuario</a>
     <script>
           function preguntar(){
               return confirm("Quieres ir a google?");
           };
     </script>
```

EVENTOS TRADICIONAL

```
<script>
       // IMPORTANTE  no poner parentesis a la funcion, si no la ejecuta
directamente no cuando se de al click
       document.getElementById("tradicional").onclick=cambiar;

       function cambiar(){
           alert("Entramos en la función cambiar");
           document.getElementById("tradicional").innerHTML = "Modelo de
registro de eventos tradicional";
           document.getElementById("tradicional").onclick=null;
       }
   </script>

   <h3 id="tradicional2">Pulsa aqui</h3>
   <script>
       window.onload = function(){
           alert("La página se ha cargado correctamente");
           document.getElementById("tradicional2").onclick = miMensaje;
       }
       function miMensaje(){
           document.getElementById("tradicional2").innerHTML = "Modelo
tradiconal";
       }
   </script>
```

EVENTOS W3C

ES COMO SE ACONSEJA HACER

```
 <script>
        // elemento.addEventListener("evento sin on",funcion)
        // La funcion sin paréntesis

        document.getElementById("w3c").addEventListener("click",saludarUnaVez)
;
        document.getElementById("w3c").addEventListener("click",colorear);
        document.getElementById("w3c").addEventListener("mouseover",cambiarFon
do);


        function saludarUnaVez(){
            alert("Hola bienvenido");
            document.getElementById("w3c").removeEventListener("click",saludar
UnaVez);
        };

        function colorear(){
            document.getElementById("w3c").style.color = "red";
        };

        function cambiarFondo(){
            document.getElementById("w3c").style.backgroundColor = "pink";
        };

        // Funciones anónimas

        document.getElementById("w3canonima").addEventListener("click",functio
n(){
                this.style.backgroundColor ="skyblue";
        });
    </script>
```

EVENTOS INFORMACIÓN

Pasar siempre el evento (e), en la función del escuchador, para posteriormente poder obtener información

INFORMACIÓN INTERNA DEL EVENTO

e.type → devuelve el tipo de manejador (click, mouseover…)

e.target → devuelve la etiqueta entera, del elemento donde ha ocurrido el evento(<button id=”contenido” class=”contenedor”>)

e.currentTarget → te devuelve hasta donde se propaga el evento, el elemento más externo.

e.target.id → devuelve el id que tenga el elemento que ha activado el evento (ejemplo anterior → contenido).

e.target.nodeName → devuelve la etiqueta en mayúsculas (ejemplo anterior (BUTTON).

e.target.getAttribute(‘id-data’)→ devuelve el atributo que tenga tu elemento <button id=”contenido” id-data=’principal’ class=”contenedor”>

e.target.value → devuelve el valor que tiene tras el evento

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Eventos información</title>
  </head>
  <body>
    <p id="parrafo">Parrafo 1</p>
    <h1 id="eventos">Obtener información de un evento</h1>
    <h2 id="eventos2">Evento 2</h2>
```

```
    <h2 id="eventos3">Evento 3</h2>

    <script>
      document.getElementById("parrafo").addEventListener("click", manejador);
      document
        .getElementById("eventos")
        .addEventListener("mouseover", manejador);
      document
        .getElementById("eventos")
        .addEventListener("mouseout", manejador);
      document.getElementById("eventos2").addEventListener("click", saludo);
      document.getElementById("eventos3").addEventListener("click",  saludo);

      // el "e" nos devuelve informacion interna del evento, con type nos
devuelve el tipo de manejador (el evento) en este caso mouseover o mouseout en
cuanto el h1. e.type=> que evento se esta produciendo
      function manejador(e) {
        console.log(e.type);
        switch (e.type) {
          case "mouseover":
            this.style.color = "red";
            break;
          case "mouseout":
            this.style.color = "green";
            break;
          case  "click":
            this.style.color = "yellow";
            break;
        }
      }
      function saludo(e){
        // e.target.id; te devuevle el identificador del elemento en este caso
(eventos2/eventos3)
        console.log(e.target.id);
        if (e.target.id=="eventos2"){
            alert("Has pulsado eventos 2")
        } else if (e.target.id=="eventos3"){
            alert("Has pulsado eventos 3")
        }
      }
    </script>
```

EJERCICIOS

"CRONO"

```
<!-- Tres botones. uno cada vez que pinche va a incrementar si pulsa
decrementar resta y tercer boton que es el resert que lo pone a 0  -->
    <p id="contador"></p>
    <button id="incrementar">Incrementar</button>
    <button id="disminuir">Disminuir</button>
    <button id="reset">Reset</button>
    <button id="doble">x2</button>
    <button id="mitad">/2</button>
  </body>
  <script>
    document.getElementById("incrementar").addEventListener("click", crono);
    document.getElementById("disminuir").addEventListener("click", crono);
    document.getElementById("reset").addEventListener("click", crono);
    document.getElementById("doble").addEventListener("click", crono);
    document.getElementById("mitad").addEventListener("click", crono);
    let cont = 0;
    function crono(e) {
      switch (e.target.id) {
        case "incrementar":
          cont++;
          document.getElementById("contador").innerHTML = cont;
          break;
        case "disminuir":
          cont--;
          document.getElementById("contador").innerHTML = cont;
          break;
        case "reset":
          cont = 0;
          document.getElementById("contador").innerHTML = cont;
          break;
        case  "doble":
          cont = cont * 2;
          document.getElementById("contador").innerHTML = cont;
          break;
        case  "mitad":
          cont = cont / 2;
          document.getElementById("contador").innerHTML = cont;
          break;
      }
    }
  </script>
```

EJERCICIO COCHE

HTML+JS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejercicio prueba</title>
    <link rel="stylesheet" href="ejercicioImagen.css">
</head>
<body>
    <img id="coche" src="coche.png" alt="">



    <script>
        document.getElementById("coche").addEventListener("dblclick",
manejador);
        document.getElementById("coche").addEventListener("click", manejador);
        document.getElementById("coche").addEventListener("mouseover",
manejador);
        document.getElementById("coche").addEventListener("wheel",  manejador);

        function manejador(e){

            switch(e.type){
                case "dblclick":
                    this.classList.add("acelerar");
                    this.classList.remove("volver");
                    this.classList.remove("escala");
                    this.classList.remove("caer");
                    break;
                case "click":
                    this.classList.add("volver");
                    this.classList.remove("acelerar");
                    this.classList.remove("escala");
                    this.classList.remove("caer");
                    break;
                case "mouseover":
                    this.classList.add("escala");
                    this.classList.remove("acelerar");
                    this.classList.remove("voler");
                    this.classList.remove("caer");
                    break;
                    break;
                case "wheel":
                    this.classList.add("caer");
                    break;
            }
```

```
        }
    </script>

</body>
</html>
```

CSS

```
img {
  width: 400px;
  height: 200px;
  margin: 200px;
}
.acelerar {
  transition: translate, 10s;

  transform: translate(700px);
}
.volver {
  transition: translate, 5s;
  transform: translate(-100px);
}
.escala {
  transition: scale, 5s;
  transform: scale(2, 2);
}
.caer {
  transition: rotate, 3s;
  transform: rotate(45deg);
}
```

WebStorage

```
<!-- WEB STORAGE
     - Permite almacenar datos localmente en el ordenador del usuario
     - Es más seguro y almacena más informacion que las cookies
     -->
```

Pueden emplearse, por ejemplo, para guardar las puntuaciones de un juego.

```
// Primero en un if general, verificamos si el navegador soporta Storage, si si
que lo soporta metemos todo el código dentro

if(typeof(Storage)!=="undefined"){
alert("El navegador soporta webstorage");

}else{

    alert("El navegador no soporta webstorage");
}
```

LOCALSTORAGE

```
NO TIENEN FECHA DE EXPIRACION (localstorage)

// Creamos un elemento localstorage
        window.localStorage.setItem("usuario","Oscar");

// Acceder a los datos de localstorage
        alert(localStorage.getItem("usuario"));

// Borrar datos a nivel individual ( un elemento )
        localStorage.removeItem("apellido");

// Eliminar todos los elementos

localStorage.clear();
```

SESIONSTORAGE

```
/ Creamos un elemento sesionstorage
         sessionStorage.setItem("usuario2","Oscar2");

// Acceder a los datos de sesionStorage

        alert(sessionStorage.getItem("usuario2"));

// Borrar datos a nivel individual ( un elemento )
        sessionStorage.removeItem("apellido2");

// Eliminar todos los elementos

        // AL CERRAR LA SESION SE HACE UN CLEAR AUTOMATICAMENTE. ES
ALMACENAMIENTO DE SESION

sessionStorage.clear();
```

DOM

TRAVERSING

TIPOS DE NODOS

```
// // // - De tipo elemento (etiquetas HTML)
// // // - De tipo texto (textos,intros,tabulaciones)
// // // - De tipo atributo (class,id,href...)
// // // - De tip comentario (<!--  -->)
// // // - De objeto raiz (document, solo existe uno por cada documento, es la
raiz)
```

1a PARTE MOVERSE VERTICALMENTE

```
// // // childNodes vs children
// // // firstChild vs firstElementChild
// // // lastChild vs lastElementChild
// // // hasChildNodes()
```

childNodes

```
// childNodes (nos devuelve todo tipo de nodos, incluidos textos, tabulaciones, intros...)
```

- Ejemplo

```
// let body=document.body.childNodes;
// // console.log(body);
// // let wrapper=document.body.childNodes[9];
// // console.log(wrapper);
```

children

```
// children (nos devuelve unicamente elementos de tipo HTML)
```

- Ejemplo

```
// // let tipoElemento = document.body.children;
// // console.log(tipoElemento);
// // console.log(tipoElemento[0]);
```

firstChild

```
// firstChild nos devuelve elementos que no son HTML(tabulaciones,intros..)
```

- Ejemplo

```
// let primerNodo = document.body.firstChild;
// // console.log(primerNodo);
```

FirstElementChild

```
// firstElementChild, devuelve solo HTML
```

- Ejemplo

```
/ let primerElemento = document.body.firstElementChild;
```

RECORRER EL ÚLTIMO NODO

LastChild

```
/ lastChild, devuelve elementos que no son HTML (tabulaciones,intros..)
```

- ejemplo

```
// let ultimoNodo= document.body.lastChild;
```

LastElementChild

```
// lastElementChild, devuelve solo HTML
```

HasChildrenNodes()

```
// Metodo hasChildNodes() devuelve true o false en funcion de si el elemento tiene hijos, incluye todo tipo de elementos
```

2a PARTE MOVERSER HORIZONTALMENTE

```
// nextSibling vs nextElementSibling
// // // previousSibling vs previousElementSibling
// // // parentNode vs parentElement
```

CONSEJO :

```
//  La idea es ir guardandonos en variables los elementos par ano tener que recorrerlos cada vez.
```

3a PARTE

getElementByID()

```
/ let wrapper=document.getElementById("wrapper");
```

getElementByClassName()

```
// let links= document.getElementsByClassName("link");
```

getElementsByTagName()

```
// let div =document.getElementsByTagName("div");
```

getElementsByName()

Orientado a formularios donde esta el atributo name

```
// let inputTelefono=document.getElementsByName("telefono");
```

querySelector()

```
// let wrapper2 = document.querySelector(".wrapper");
```

```
// let wrapper3 = document.querySelector("#wrapper");
// let etiquetaDiv= document.querySelector('div');
// let queryWrapper= document.querySelector('.link.bold');
```

querySelectorAll()

Con querySelectorAll(), te devuelve un nodeList

```
// let divA= document.querySelectorAll('div > a');
```

Diferencias entre nodeList y HTMLCollection

nodeList

Con nodeList se pueden utilizar forEach

```
// let linksQuery = document.querySelectorAll('.link');
/ linksQuery.forEach(link =>{
//     console.log(link);
// })
```

Si el DOM cambia, la nodeList no se actualiza, en cambio HTMLCollection si se actualiza.

HTMLCollection

Si cambia el DOM HTMLCollection si que se actualiza.

```
// let crearA = document.createElement('a');
// crearA.setAttribute('class','link');
// crearA.textContent="Hola buenos días";
// document.body.append(crearA);
// let linksHTML = document.getElementsByClassName('link');
```

Pueden utilizar el método item para referirse a un elemento

```
// console.log(linksHTML.item(1));
```

Convertir nodeList en array, para emplear métodos array

```
// let linksArray = Array.from(document.querySelectorAll('.link'));
// console.log(linksArray);
```

4a PARTE

console.dir

Con console.dir, devuelve todas las propiedades que tiene

className

Se puede usar como lectura o como escritura, si se usa como escritura **SOBREESCRIBIRÁ** todas las clases que tenga class.

```
wrapper.className="container";
```

nodeName

devuelve el nombre del nodo en el cual se llama a la propiedad

```
/ console.log(title.nodeName);
```

innerHTML

Tanto de escritura como de lectura, actúa sobre lo que hay dentro del elemento, no sobre el propio elemento, si es un actúa dentro de ese div, sin modificarlo. **Cuando se quiere añadir solo tenxto dentro de algún elemento es preferible utilizar textContent antes que innerHTML, ya que este ultimo puede añadir etiquetas HTML.**

```
const wrapper=document.querySelector('.title');
wrapper.innerHTML ='<h2>Sobreescribo el DOM </h2>';
```

outerHTML

A diferencia de innerHTML, si que actúa sobre el elemento.

```
wrapper.outerHTML = `<h2>Nuevo HTML</h2>`;
```

textContent

Unicamente inyecta **TEXTO** no se puede utilizar ni HTML ni JavaScript

5ª PARTE

setAttribute()

Se pueden añadir atributos al igual que los objetos

```
// title.id="encabezado";
```

Otra forma es con **setAttribute().** Este método sobreescribe.

```
wrapper.setAttribute('id','primero');
```

de modo que cambiará el id que tuviera, en el caso de que tuviese, y pondrá ‘primero’.

getAttribute()

Primero nos aseguramos de que si existe ese atributo si devuelve null es que

no existe

```
// console.dir(title.getAttribute('href'));
// console.dir(title.getAttribute('class'));
```

hasAttribute()

para saber si el elemento tiene ese atributo o no

```
// console.log(imagen.hasAttribute('alt'));
```

removeAttribute()

para borrar un atributo

```
// imagen.removeAttribute('alt');
```

```
      // // Te devuelve los atributos que tiene ese elemento

// console.log(imagen.attributes);
```

El resutlado de estas dos sentencias es el mismo

```
// // let imagen2=document.querySelector(`[identificador="img"]`);
// // console.log(imagen2);

// // console.log(imagen2.getAttribute('identificador'));
```

BUENAS PRACTICAS

```
// // <!-- Como buenas practicas a los atributos inventados por nosotros (en el
HTML) se pone data- delante -->

// let imagen3=document.querySelector(`[data-identificador="img"]`);

// console.log(imagen3.getAttribute('data-texto-mostrar'));
```

Eventos DOM

EVENT BUBBLING O PROPAGACIÓN DE EVENTOS

Recomendación → poner (e) en las funciones como parámetro, aunque creamos que no lo vamos a usar, porque se puede obtener mucha información del evento.

Ponemos el escuchador en el body, de modo que todo lo que este contenido tendrá el escuchador (evento bubbling).

```
document.body.addEventListener('click',handleEventTarget);
```

e.target.nodeName → nos muestra donde estoy haciendo click, cada elemento.(BUTTON,SECTION …)

```
function handleEventTarget(e){
```

```
    console.log(`Has dado click en ${e.target.nodeName}`);
};
```

e.currentTarget.nodeName → Devuelve el contenedor más alto de manera jerárquica.
En este caso devolverá BODY, da igual si haces click encima de un button.

```
document.body.addEventListener('click',handleEventCurrentTarget);

function handleEventCurrentTarget(e){
    console.log(`Has dado click en ${e.currentTarget.nodeName}`);
};
```

En el caso de que el evento este en el BODY, dentro de éste en un SECTION y dentro de este en un BUTTON, al dar al BUTTON mostrará de dentro a fuera, aunque el orden sea diferente, de esta manera:

```
document.body.addEventListener('click',handleEventCurrentTarget);
botonShow.addEventListener('click',handleEventCurrentTarget);
section.addEventListener('click',handleEventCurrentTarget);

function handleEventCurrentTarget(e){
    console.log(`Has dado click en ${e.currentTarget.nodeName}`);
};
```

```
Has dado click en BUTTON
Has dado click en SECTION
Has dado click en BODY
```

Se muestra en la consola de dentro a fuera, del más bajo al más alto.
De tal manera que poniendo los escuchadores en los mismos elementos, y empleando la misma función,esto se mostrará al hacer click en el mismo button que el ejemplo anterior.

```
document.body.addEventListener('click',handleEventCurrentTarget, {capture:
true});
botonShow.addEventListener('click',handleEventCurrentTarget, {capture: true});
section.addEventListener('click',handleEventCurrentTarget, {capture: true});
```

FASE DE CAPTURA

Te devuelve de fuera hacia dentro, al contrario que el evento bubbling.
Hay que añadirle un tercer parámetro {captura:true}.

```
document.body.addEventListener('click',handleEventCurrentTarget, {capture:
true});
botonShow.addEventListener('click',handleEventCurrentTarget, {capture: true});
section.addEventListener('click',handleEventCurrentTarget, {capture: true});
```

```
function handleEventCurrentTarget(e){
    console.log(`Has dado click en ${e.currentTarget.nodeName}`);
};
```

```
Has dado click en BODY
Has dado click en SECTION
Has dado click en BUTTON
```

Resumen → pulsando el mismo botón :

- Con bubbling, nos devuelve los elementos de "dentro hacia fuera"
  - Button,section,body. Esto es por defecto siempre así
- Con captura, nos devuevle de "fuera a dentro"
  - Body,section,button. Hay que especificar el parámetro {captura:true}
  -

Para parar la propagación, **stopPropagation()**

matches() y closest()

matches(), nos devuelve true o false dependiendo de si encuentra el elemento o no.
Por ejemplo:
En este ejemplo recorremos todos los elementos que tienen el atributo [data-id], y el que coincida su valor de atributo con "button-show" (es cuando el valor de currentElement , se convierte en true, evaluado en el if), imprimirá en la consola el mensaje.

```
const elements= document.querySelectorAll('[data-id]');

for(element of elements){
    const currentElement = element.matches('[data-id="button-show"]');

    if (currentElement){

        console.log(`El boton ${element.nodeName} contiene el boton botonShow`);
    };
};
```

closest(), nos devuelve el padre más cercano de la clase que hemos indicado ejemplo

```
let input = document.querySelector('[type="text"]');

console.log(input.closest('.wrapper') );
```

En este caso el padre más cercano al elemento input con la clase wrapper, si lo encuentra devuelve al elemento.
Si no lo encuentra devuelve null, aquí lo ha encontrado por lo que devuelve el elemento.

```
<main class="wrapper" data-id="wrapper">
```

EVENT DELEGATION

Consiste en delegar la acción, en el ejemplo, se ve que aunque haya sido el section quien tiene el evento, va a ser currentTarget, pero obteniendo el valor de e.target, que será al elemento que se ha pulsado, en este caso un button dentro del section.

```
let section = document.querySelector('.section');

section.addEventListener('click',cambioColor);

function cambioColor(e){
    e.currentTarget.style.backgroundColor = e.target.getAttribute('data-
color');

    // El target te dice el elemento que has pulsado, el currentTarget te dice
hasta donde se propaga es decir en este caso el section.
}
```

CLASES

classList

```
// Es solamente de lectura y devuelve una coleccion DOMTokenList

console.log(modal.classList);
console.log(modal.classList[0]);
```

método item()

```
 // metodo item: devuelve un elemento o una clase de una lista de clases. Como
parámetro le pasamos el indice de la clase que queramos
console.log(modal.classList.item(2));
console.log(modal.classList[2]);
```

método add()

NO MODIFICA al resto de clases. Se pueden añadir más de una clase

```
//  metodo add(): nos permite agregar una clase al elemento SIN MODIFICAR al
resto de clases que existen actualmente

button.addEventListener("click", function(){
    modal.classList.add('show');
});

modal.classList.add('show','clase-nueva'). //Se pueden pasar más de una clase
```

método remove()

Eliminar clases de un elemento

```
close.addEventListener("click", function(){
    modal.classList.remove('show');
    console.log(modal.classList);
});
```

método toggle()

Añade y quita, si la clase está, la quita y si no está, la añade

```
button.addEventListener('click',function(){
        button.classList.toggle('red');
    }
);
```

Sustituto de:

```
// button.addEventListener('click',function(){
//     if(button.matches('.red')){
//         button.classList.remove('red');
//     } else {
//         button.classList.add('red');
//     }
// });
```

método contains()

Devuelve true si la classList contiene la clase pasada por parámetro, si no devuelve false

```
console.log(modal.classList.contains('modal'));
```

método replace()

Te permite reemplazar una clase que existe por otra. Devuelve true si la clase que quieres reemplazar existe y la reemplaza, si no existe devuelve false y no hace nada.

```
console.log(modal.classList.replace('clase1','clase100'));
```

CREATE ELEMENT

```
// createElement()
// appendChild() vs append()
// prepend()
// insertBefore() vs before() vs after()
// removeChild() vs remove()
```

```
// replaceChild() vs replaceWhith()
// insertAdjacentHTML() vs insertAdjacentElement()
// cloneNode()
```

createElement()

Crea un elemento

```
let newTask = document.createElement('li');
```

appendChild()

para añadir al final de una lista dentro de un elemento padre.
elementoPadre.appendChild(elementoHijoAAñadir);

```
function createTask(value){
    let newTask = document.createElement('li');
    // Cuando vamos a añadir string mejor usar tesxtContent ante innerHTML, por
que este ultimo puede añadir codigo js html ...

    newTask.textContent = value;
    contenedor.appendChild(newTask);
    // Tambien podria utilizar el append: contenedor.append(newTask);
    // console.log(newTask);
    addEvents(newTask);
};
```

append()

Permite añadir también strings, mientras que appendChild(), solo permite añadir nodos

```
let div = document.createElement("div");
let p = document.createElement("p");
div.append("Algo de texto", p);
console.log(div.childNodes); // NodeList [ #text "Algo de texto", <p> ]


let div = document.createElement("div");
let p = document.createElement("p");
div.append(p);

console.log(div.childNodes); // NodeList [ <p> ]
```

prepend()

Inserta delante de los elementos que ya hubiese, colocándose el primero en la lista

```
let div = document.createElement("div");
let p = document.createElement("p");
let span = document.createElement("span");
div.append(p);
div.prepend(span);

console.log(div.childNodes); // NodeList [ <span>, <p> ]
```

a pesar de haber añadido antes <p>, al añadir span con prepend se coloca el primero

style

CSSOM – CSS OBJECT MODEL

Para ver los estilos que tiene cualquier documento, por ejemplo una variable llamada titulo

```
console.log(title.style);
```

Para acceder a un estilo en concreto

```
console.log(title.style.color);
```

```
// nomenclatura camellCase

title.style.fontFamily = "cursive";
title.style.color="darkred";
title.style.fontSize="4rem";
```

property ()

Se usa estilo css

```
setProperty(), es como crear variables globales de estilos
```

```
document.documentElement.style.setProperty('--color-title','hotpink');
title.style.color= "var(--color-title)";
```

De tal manera que si se cambia la variable global cambiará todos donde se ha aplicado

- Quitar el valor de una propiedad

1ª forma → se emplea nomenclatura css

```
title.style.removeProperty('font-family');
```

2ª forma → dejando el blanco el valor

```
title.style.color= "";
```

3ª forma → así se borran todos los estilos en línea con **setAttribute**

```
title.setAttribute("style","");
```

Ejemplos

```
inputColor.addEventListener('input',function(e){
    const newColor =e.target.value;
```

```
    title.style.color=newColor;
});
```

```
inputRage.addEventListener('input',function(e){
    const tamaño=e.target.value;
   title.style.setProperty('font-size',tamaño+"px",'important');
   console.log(title.style.getPropertyPriority('font-size'));
});
```

getPropertyValue()

Con property estilo css con guiones.

```
console.log(title.style.getPropertyValue('font-family'));
```

Con style se usa camelCase

```
console.log(title.style.fontFamily);
```

```
// Nos sirve para saber que codigo se esta aplicando

console.log(estilosTitle2.getPropertyValue('content'));
console.log(estilosTitle2.getPropertyValue('margin'));
```

getComputedStyle ()

```
// Te devuelve un obejto con los estilos finales que se está aplicando tanto en
linea como en css del elemento del cual se usa como parametro.

console.log(getComputedStyle(title));
const estilosTitle =getComputedStyle(title);

console.log(estilosTitle.color);

const estilosTitle2 = getComputedStyle(title,'::after');
```