INTRODUCCIÓN.

¿Cómo introducir nuestro archivo de JS?

• Código incrustado en el HTML
<script>
alert('Hello, world!');
</script>
• Script externo:
<script src="/path/to/script.js"></script>

Ahora conviene añadir los scripts en <head>. Podemos usar (ponerlo al final de la línea):

- Atributo **defer**: LO MEJOR (en grl). Se pausa la carga del HTML hasta que se carga JS. Se ejecuta HTML una vez cargado el JS. Esto es importante cuándo interactúan HTML y JS.
- Atributo **async**: se cargan ambos (HTML y JS) a la vez. 1º puede que se ejecute el HTML, por lo que se puede cargar JS.

INTERACCIÓN.

Tenemos una serie de funciones predefinidas.

- alert: le muestra un mensaje al usuario. Salen como pop-ups. Ej: alert("Hola");
- prompt: le pide al usuario que introduzca información. NO deberíamos usarlo en producción porque para la página por completo. Nos puede ser útil para un input o tb podemos usar el console.log para mostrar texto.
- console.log: nos muestra / pide información por pantalla. Ej: console.log("Esto se imprime en la consola");
- confirm: le solicita al usuario que confirme. Ej: let str = confirm("¿Estás de acuerdo?"); console.log(str) //true o false

NO USARLOS EN PRODUCCIÓN PORQUE SON MUY INTRUSIVOS. Bloquean completamente la página. Usaremos “console.log()” para pruebas.

VARIABLES.

Tenemos dos tipos de variables: let y const. También existe var pero NO se debe usar.

- let: podemos almacenar un dato con nombre, o declaración múltiple. Ejemplos:
  let message = ‘Hello’;
  let user = 'John',
  age = 25,
  message = 'Hello';
- const: variable que NO cambia de valor. NO se puede reasignar su valor. Además, TIENE que asignarse un valor al declararlas.

TIPOS DE DATOS – STRING.

"string".length -> devuelve el número total de caracteres de una cadena.

"string".charAt(index) -> devuelve el carácter en una posición específica (índice) de una cadena. Los índices comienzan desde 0.

"string".split(pattern) -> divide una cadena en un array de substrings, utilizando un patrón (separador) especificado.

"string".slice(start, [end]) -> extrae una parte de una cadena, desde el índice start (incluido) hasta el índice end (excluido). Si se omite end, se extrae hasta el final de la cadena.

- **length**: Obtiene la longitud de una cadena.
- **charAt**: Obtiene un carácter específico.
- **split**: Divide una cadena en un array.
- **slice**: Extrae una parte de una cadena.

COMPARACIONES.

Para que no se haga la conversión, usar la igualdad estricta ===

Si los tipos son distintos, se devuelve false con ===

CONDICIONALES.

- If: se pone como -> .... ? .... : ..... Ej: let accessAllowed = (age > 18) ? true : false; (condition) ? (true) : (else-false)
- OR and AND. OJO!!! En el AND, si se llega al final, se devuelve el valor del último operando.

NULLISH OPERATOR.

Si es null, me devuelve otra cosa que le haya puesto.

Ej: const miUsuario = localStorage.get(usuario) ?? usuarioPorDefecto (tb puedo hacerlo sin definir la ult variable)

Otro: pp = [1,null]

const numero = pp[0] ?? 0

const numero1 = pp[1] ?? 0

➔ Si pongo numero, me devuelve 1. Si pongo numero1, me devuelve 0. Si no tuvieran la interrogación "?", numero1 em devolvería "null"

ARRAYS.

• Crear arrays: let fruits = ['Apple', 'Banana'];

• Acceder a un elemento: console.log(fruits[0]);

• Longitud: console.log(fruits.length);

• Recorrerlo: fruits.forEach(function(item, index, array) { console.log(item, index); });

• Añadir un element: let newLength = fruits.push('Orange');

• Eliminar el último elemento: let last = fruits.pop();

• Buscar un índice de un elemento: let pos = fruits.indexOf('Banana');

• Eliminar un elemento: let removedItem = fruits.splice(pos, 1);

CALLBACK.

Def: funciones pasadas como argumento a otra función. Pueden ser síncronas, pero habitualmente son asíncronas.

Como todo, las variables globales SÍ son accesibles desde fuera, pero las que declaro como argumento de una función NO lo son.

Tengo dos formas de declarar el callback:

> [1,2,3,4,].forEach(callback); -> lo hace de forma SÍNCRONA
>
> setTimeout(callback, 1000); -> lo hace de forma ASÍNCRONA, nos permite invocar a la función una vez hayan pasado 1000 mseg (1 seg)

LISTA DE EVENTOS COMUNES EN ELEMENTOS HTML.

## 1. <button>, <input type="button">, <input type="submit">

- click: Se dispara cuando se hace clic en el botón.

- dblclick: Se dispara cuando se hace doble clic en el botón.

- focus: Se dispara cuando el botón gana el foco.

- blur: Se dispara cuando el botón pierde el foco.

- mousedown: Se dispara cuando se presiona un botón del ratón sobre el botón.

- mouseup: Se dispara cuando se suelta el botón del ratón sobre el botón.

- mouseover: Se dispara cuando el ratón pasa sobre el botón.

- mouseout: Se dispara cuando el ratón sale del botón.

En estos casos, puedo comprobar si ha hecho click o no añadiendo un botón, y luego guardando las veces que hace click etc. Si quiero ver cuántas,

“addEventListener(‘click’, click)” -> añadir e.preventDefault(); por si el user no hace nada

## 2. <form>

- submit: Se dispara cuando se envía el formulario.

- reset: Se dispara cuando el formulario se reinicia.

- focusin: Se dispara cuando un elemento dentro del formulario gana el foco.

- focusout: Se dispara cuando un elemento dentro del formulario pierde el foco.

## 3. <input> (text, password, email, checkbox, radio, etc.)

- input: Se dispara cada vez que cambia el valor del campo.

- change: Se dispara cuando el valor del campo cambia y pierde el foco.

- focus: Se dispara cuando el campo gana el foco.

- blur: Se dispara cuando el campo pierde el foco.

Para ver si se están cumpliendo, debo hacer lo siguiente (xa los 3 eventos de teclado sig):

```
document.addEventListener(‘keydown’,logKey);
function logKey(e) {
        console.log(e.code); -> cdo pulso una tecla me dice cuál es
}
```

- keydown: Se dispara cuando una tecla es presionada.
- keyup: Se dispara cuando una tecla es soltada.
- keypress: Se dispara cuando una tecla es presionada y mantenida.
- click: Se dispara cuando se hace clic en el campo (por ejemplo, en un checkbox).
- dblclick: Se dispara cuando se hace doble clic.
- mousedown: Se dispara cuando se presiona un botón del ratón.
- mouseup: Se dispara cuando se suelta un botón del ratón.
- mouseover: Se dispara cuando el ratón pasa sobre el campo.
- mouseout: Se dispara cuando el ratón sale del campo.

## 4. <textarea>

- input: Se dispara cuando cambia el contenido del textarea.
- change: Se dispara cuando el contenido cambia y el textarea pierde el foco.
- focus: Se dispara cuando el textarea gana el foco.
- blur: Se dispara cuando el textarea pierde el foco.
- keydown: Se dispara cuando se presiona una tecla.
- keyup: Se dispara cuando se suelta una tecla.
- keypress: Se dispara cuando se presiona una tecla y se mantiene.

## 5. <select>

- change: Se dispara cuando el valor seleccionado cambia.
- focus: Se dispara cuando el select gana el foco.
- blur: Se dispara cuando el select pierde el foco.

## 6. <img>

- load: Se dispara cuando la imagen ha sido completamente cargada.
- error: Se dispara si hay un error al cargar la imagen.
- abort: Se dispara si la carga de la imagen es interrumpida.
- mouseover: Se dispara cuando el ratón pasa sobre la imagen.
- mouseout: Se dispara cuando el ratón sale de la imagen.

## 7. , , <p>, <section>, <article> (contenedores genéricos)

- click: Se dispara cuando se hace clic en el elemento.
- dblclick: Se dispara cuando se hace doble clic.
- mousedown: Se dispara cuando se presiona un botón del ratón.
- mouseup: Se dispara cuando se suelta un botón del ratón.
- mouseover: Se dispara cuando el ratón pasa sobre el elemento.
- mouseout: Se dispara cuando el ratón sale del elemento.
- mouseenter: Se dispara cuando el ratón entra en el área del elemento (sin burbujas).
- mouseleave: Se dispara cuando el ratón sale del área del elemento (sin burbujas).
- contextmenu: Se dispara cuando se abre el menú contextual (clic derecho).

## 8. <a> (enlace)

- click: Se dispara cuando se hace clic en el enlace.
- focus: Se dispara cuando el enlace gana el foco.
- blur: Se dispara cuando el enlace pierde el foco.
- mouseover: Se dispara cuando el ratón pasa sobre el enlace.
- mouseout: Se dispara cuando el ratón sale del enlace.

## 9. <audio>, <video>

- play: Se dispara cuando la reproducción empieza.
- pause: Se dispara cuando la reproducción se detiene.
- ended: Se dispara cuando la reproducción finaliza.
- timeupdate: Se dispara cuando cambia el tiempo actual durante la reproducción.
- volumechange: Se dispara cuando el volumen cambia.
- seeking: Se dispara cuando el usuario comienza a buscar en el archivo multimedia.
- seeked: Se dispara cuando el usuario ha terminado de buscar en el archivo multimedia.
- progress: Se dispara periódicamente mientras se descargan los datos.

## 10. Eventos Generales (aplicables a múltiples elementos)

- focus: Se dispara cuando un elemento gana el foco.
- blur: Se dispara cuando un elemento pierde el foco.

- keydown: Se dispara cuando una tecla es presionada.

- keyup: Se dispara cuando una tecla es soltada.

- keypress: Se dispara cuando se presiona una tecla y se mantiene.

- mouseover: Se dispara cuando el ratón pasa sobre el elemento.

- mouseout: Se dispara cuando el ratón sale del elemento.

- click: Se dispara cuando se hace clic en el elemento.

- dblclick: Se dispara cuando se hace doble clic en el elemento.

- contextmenu: Se dispara cuando se abre el menú contextual (clic derecho).

- drag: Se dispara cuando se arrastra un elemento.

- drop: Se dispara cuando un elemento es soltado en una zona de arrastre.

- scroll: Se dispara cuando se desplaza la página o un elemento con scroll.

//addEventListener

element.onclick = function () {

// ... function logic ...

};

event.preventDefault();

En el controlador de eventos del formulario evita ese comportamiento. En lugar de enviar los datos y recargar la página, el formulario no se envía, permitiéndote manejar los datos en el código JavaScript sin recargar la página.

Esto es especialmente útil cuando:

- **Validas datos del formulario** antes de enviarlo.
- **Usas AJAX** para enviar datos sin recargar la página.
- **Guardas datos en localStorage o sessionStorage**.

EJEMPLO:

javascript Copiar código

```javascript
loginForm.addEventListener("submit", function(event) {
    event.preventDefault(); // Previene el envío del formulario

    // Aquí realizas la validación y otros pasos sin recargar la página
    if (username === "usuarioEjemplo" && password === "contraseñaEjemplo") {
        localStorage.setItem("loggedIn", "true");
        window.location.href = "perfil.html";
    } else {
        alert("Nombre de usuario o contraseña incorrectos.");
    }
});
```

En este caso, event.preventDefault(); permite mostrar una alerta de error si los datos son incorrectos sin recargar la página. Solo si los datos son correctos, se redirige al perfil del usuario mediante window.location.href, en lugar de enviarse automáticamente.

MÉTODOS COMUNES POR TIPO DE DATOS EN JAVASCRIPT.

## 1. *String*

Los objetos de tipo String tienen los siguientes métodos:

- charAt(index): Devuelve el carácter en el índice especificado.
- charCodeAt(index): Devuelve el código Unicode del carácter en el índice especificado.
- concat(string2, string3, ...): Combina dos o más cadenas y devuelve una nueva.
- includes(searchString): Verifica si la cadena contiene la subcadena especificada.
- endsWith(searchString): Verifica si la cadena termina con la subcadena especificada.
- indexOf(searchValue): Devuelve el índice de la primera aparición de la subcadena.
- lastIndexOf(searchValue): Devuelve el índice de la última aparición de la subcadena.
- match(regex): Busca coincidencias basadas en una expresión regular.
- replace(searchValue, newValue): Reemplaza las coincidencias de la subcadena con otra cadena.
- search(regex): Busca el índice basado en una expresión regular.
- slice(start, end): Extrae una parte de la cadena.
- split(separator): Divide la cadena en un array utilizando un separador.
- startsWith(searchString): Verifica si la cadena comienza con la subcadena especificada.
- substring(start, end): Extrae caracteres entre índices especificados.
- toLowerCase(): Convierte la cadena a minúsculas.
- toUpperCase(): Convierte la cadena a mayúsculas.
- trim(): Elimina los espacios en blanco de ambos lados de la cadena.
- padStart(targetLength, padString): Rellena el inicio de la cadena con otra cadena.
- padEnd(targetLength, padString): Rellena el final de la cadena con otra cadena.

## 2. *Number*

Los objetos de tipo Number tienen los siguientes métodos:

- toFixed(digits): Formatea el número con un número fijo de decimales.
- toPrecision(digits): Formatea el número con una precisión específica.

- toExponential(fractionDigits): Convierte el número en notación exponencial.

- toString([radix]): Convierte el número en una cadena, opcionalmente en la base especificada.

- valueOf(): Devuelve el valor primitivo del objeto Number.

### Métodos estáticos de Number:

- Number.isFinite(): Verifica si el valor es finito.

- Number.isInteger(): Verifica si el valor es un número entero.

- Number.isNaN(): Verifica si el valor es NaN.

- Number.isSafeInteger(): Verifica si el valor es un entero seguro.

- Number.parseFloat(): Analiza una cadena y la convierte en un número de coma flotante.

- Number.parseInt(): Analiza una cadena y la convierte en un entero.

## 3. *Array*

Los objetos de tipo Array tienen los siguientes métodos:

- push(element1, ..., elementN): Añade elementos al final del array.

- pop(): Elimina el último elemento del array y lo devuelve.

- shift(): Elimina el primer elemento del array y lo devuelve.

- unshift(element1, ..., elementN): Añade elementos al principio del array.

- concat(array2, array3, ...): Combina arrays y devuelve uno nuevo.

- slice(start, end): Extrae una porción del array.

- splice(start, deleteCount, item1, item2, ...): Añade, elimina o reemplaza elementos en el array.

- indexOf(element): Devuelve el índice de la primera aparición del elemento.

- lastIndexOf(element): Devuelve el índice de la última aparición del elemento.

- forEach(callback): Ejecuta una función por cada elemento del array.

- map(callback): Crea un nuevo array con el resultado de ejecutar una función en cada elemento.

- filter(callback): Crea un nuevo array con los elementos que cumplan una condición.

- reduce(callback, initialValue): Aplica una función a un acumulador y a cada elemento, reduciendo el array a un único valor.

- some(callback): Verifica si al menos un elemento cumple una condición.

- every(callback): Verifica si todos los elementos cumplen una condición.

- find(callback): Devuelve el primer elemento que cumple una condición.

- findIndex(callback): Devuelve el índice del primer elemento que cumple una condición.

- join(separator): Combina todos los elementos en una cadena utilizando un separador.

- reverse(): Invierte el orden de los elementos en el array.

- sort(compareFunction): Ordena los elementos del array.

- includes(element): Verifica si el array contiene un elemento.

- flat(depth): Aplana un array de arrays en un solo nivel o en varios niveles.

- flatMap(callback): Aplica una función a cada elemento, luego aplana el resultado.

## 4. *Object*

Los objetos de tipo Object tienen los siguientes métodos:

- hasOwnProperty(property): Verifica si el objeto tiene la propiedad especificada.

- isPrototypeOf(object): Verifica si el objeto es prototipo de otro.

- toString(): Devuelve una cadena representando el objeto.

- valueOf(): Devuelve el valor primitivo del objeto.

### Métodos estáticos de Object:

- Object.assign(target, ...sources): Copia las propiedades de uno o más objetos fuente a un objeto destino.

- Object.create(proto, [propertiesObject]): Crea un nuevo objeto con el prototipo especificado.

- Object.keys(object): Devuelve un array con las claves de un objeto.

- Object.values(object): Devuelve un array con los valores de un objeto.

- Object.entries(object): Devuelve un array con las claves y valores de un objeto.

- Object.freeze(object): Congela un objeto para que no se puedan añadir o modificar propiedades.

- Object.seal(object): Sella un objeto, permitiendo modificar propiedades existentes pero no añadir nuevas.

- Object.getOwnPropertyNames(object): Devuelve un array con todas las propiedades del objeto, incluidas las no enumerables.

- Object.getOwnPropertyDescriptor(object, property): Devuelve el descriptor de una propiedad específica.

## 5. *Boolean*

Los objetos de tipo Boolean tienen los siguientes métodos:

- toString(): Convierte el valor booleano a cadena ("true" o "false").

- valueOf(): Devuelve el valor primitivo del objeto Boolean.

## 6. *Function*

Los objetos de tipo Function tienen los siguientes métodos:

- apply(thisArg, [argsArray]): Llama a una función con un valor de this y un array de argumentos.

- call(thisArg, arg1, arg2, ...): Llama a una función con un valor de this y argumentos individuales.

- bind(thisArg, arg1, arg2, ...): Crea una nueva función con un valor de this predefinido.

- toString(): Devuelve una cadena que representa el código fuente de la función.

## 7. *Date*

Los objetos de tipo Date tienen los siguientes métodos:

- getDate(): Devuelve el día del mes (1-31).

- getDay(): Devuelve el día de la semana (0-6).

- getFullYear(): Devuelve el año completo (4 dígitos).

- getHours(): Devuelve la hora (0-23).

- getMilliseconds(): Devuelve los milisegundos (0-999).

- getMinutes(): Devuelve los minutos (0-59).

- getMonth(): Devuelve el mes (0-11).

- getSeconds(): Devuelve los segundos (0-59).

- getTime(): Devuelve el tiempo en milisegundos desde el 1 de enero de 1970.

- setDate(day):