

Introducción a JavaScript



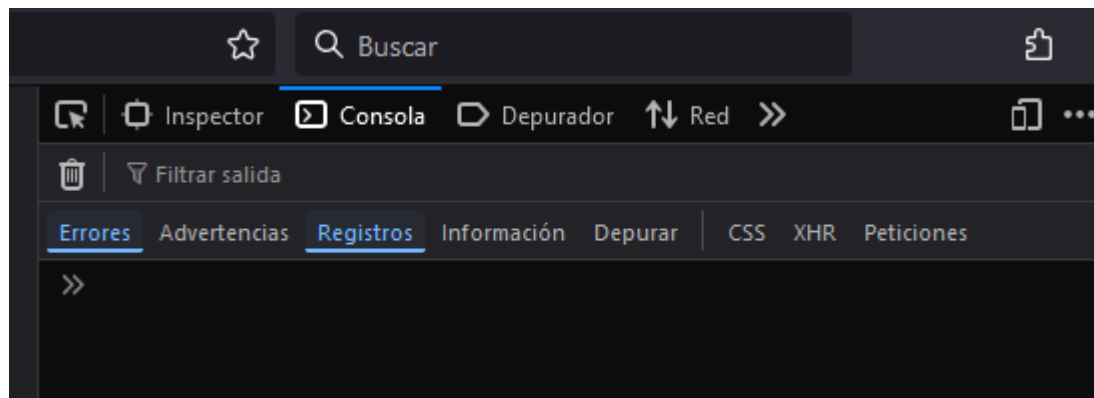
JavaScript es un lenguaje de scripting multiplataforma y orientado a objetos, se utiliza principalmente para crear páginas web dinámicas.

El código JavaScript (JS) en el documento HTML se encierra entre las etiquetas **<script></script>**, u otra forma es por medio de un archivo externo enlazado en el HTML en formato .js, habitualmente al final de la etiqueta **<body>**

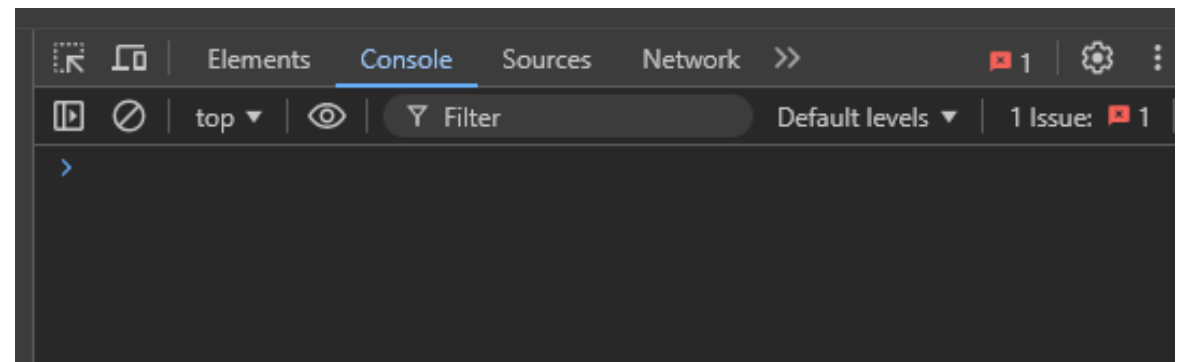
Para trabajar con JavaScript y poder debuggear nuestro código utilizaremos la consola que nos proveen los navegadores web.

Para acceder a ella, desde un navegador web presionamos la tecla F12, la cual nos abrirá una ventana con pestañas, en las pestañas seleccionamos la opción “consola”.

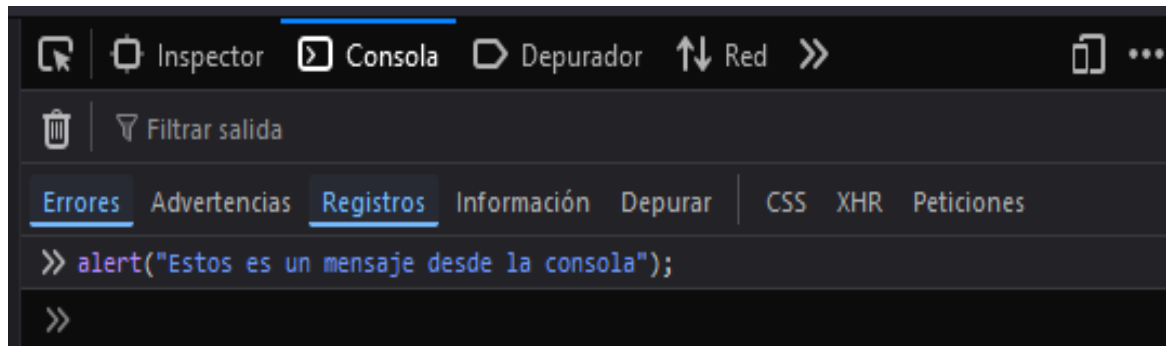
Firefox



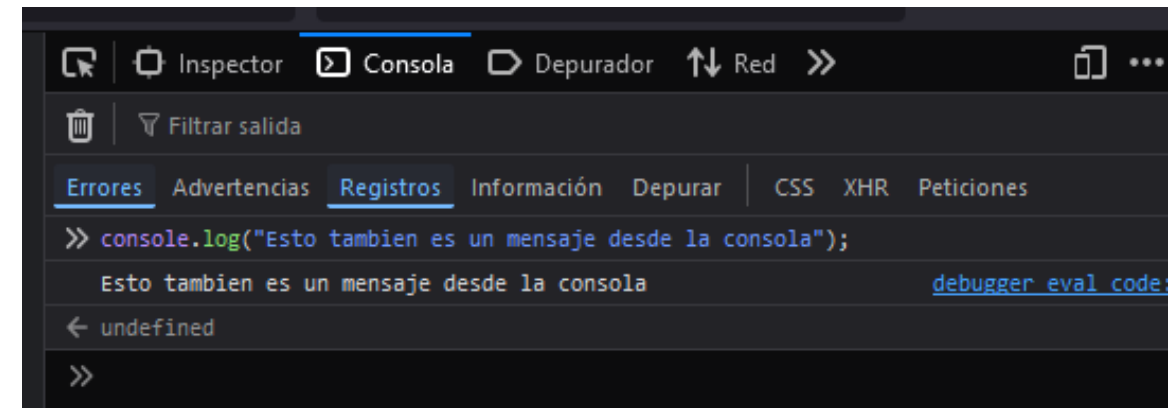
Chrome



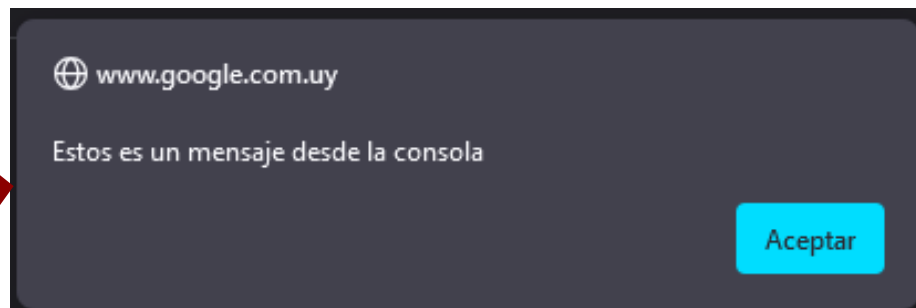
Probemos la consola con simples mensajes de texto



A screenshot of the Chrome DevTools Console. The 'Consola' tab is selected. The console shows a single log entry: `>> alert("Estos es un mensaje desde la consola");`. The console is currently empty of messages.



A screenshot of the Chrome DevTools Console. The 'Consola' tab is selected. The console shows a single log entry: `>> console.log("Esto tambien es un mensaje desde la consola");`. Below the code, the message "Esto tambien es un mensaje desde la consola" is displayed, followed by a link to "debugger eval code".



alert("Esto es un mensaje en una ventana emergente");

console.log("Esto es un mensaje que se ve en la consola");

Estas son dos formas distintas de mostrar un mensaje desde JavaScript, los “**alert()**” son ventanas emergentes que se presentan al usuario.

“**console.log()**” son mensajes que se muestran solo en la consola de los navegadores.

Habitualmente usaremos `console.log()` para ir viendo en la consola los resultados de nuestro código mientras programamos.

Variables y Constantes

Para definir variables en JS utilizamos la palabra reservada “**let**” seguido del nombre de la variable, en caso de querer definir constantes en JS utilizamos la palabra reservada “**const**” seguido del nombre de la constante, por ejemplo:

```
let numero = 198;  
const descuento = 15;
```

En JS cuando definimos variables no se les indica el tipo de dato, alcanza con la palabra `let` y especificar un nombre y un valor, JS asignará un tipo de dato automáticamente, el cual podremos ver por ejemplo utilizando la función

```
typeof(nombre_variable);
```

```
let numero = 199;  
let texto = "Hola mundo";  
let continuar = false;  
let perro = { 'nombre': "firulais", 'edad': 3 }  
  
console.log(typeof(numero)); // number  
console.log(typeof(texto)); // string  
console.log(typeof(continuar)); // boolean  
console.log(typeof(perro)); // object
```

Solicitar datos al usuario

Para solicitar datos al usuario tenemos la instrucción ***prompt()***, cuya sintaxis es la siguiente:

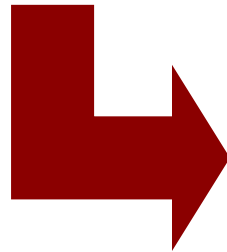
prompt(mensaje, default);

El parámetro “mensaje” es el texto que vera el usuario en pantalla, y el parámetro default es opcional, es un valor por defecto que damos la posibilidad al usuario que lo acepte o no.

El valor que introduzca el usuario será un dato de tipo string y debemos guardarlo en una variable para no perderlo.

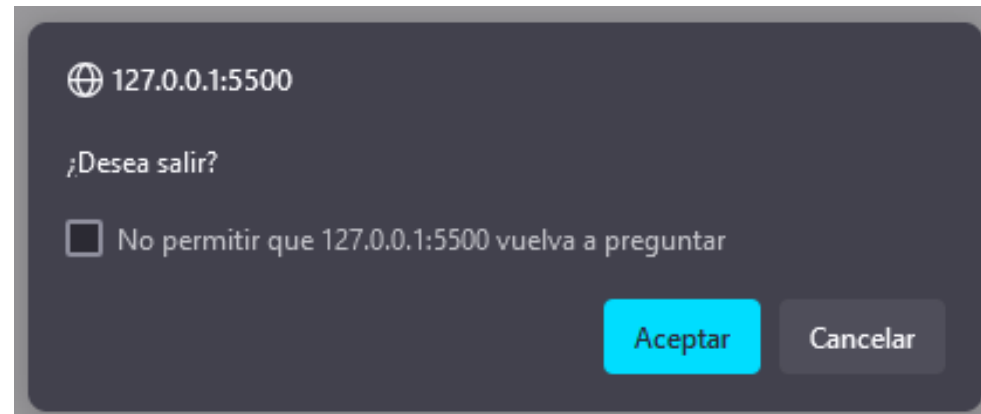
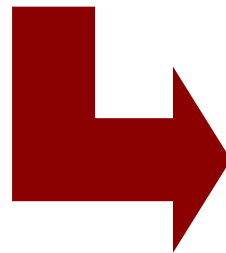
let dato = prompt(“Ingrese su nombre:”);


```
let dato = prompt("Ingrese su nombre", "MATIAS");
```

A browser prompt dialog box with a dark gray background. At the top, it shows a globe icon followed by the address "127.0.0.1:5500". Below this, the text "Ingrese su nombre" is displayed. A text input field contains the text "MATIAS" with a cursor at the end. At the bottom right, there are two buttons: "Aceptar" (highlighted in blue) and "Cancelar" (in gray).

Otra opción que tenemos es utilizando la instrucción ***confirm()***, la cual muestra una ventana de diálogo con un mensaje cuyo resultado será un valor `true` o `false`.

```
let dato = confirm("¿Desea salir?");
```



Cadenas de texto (string)

```
let texto = "Hola mundo";  
//Indica el largo de la cadena de texto  
console.log(texto.length);  
  
/*  
indexOf()  
Nos retorna la posición de la primera ocurrencia de un término de búsqueda,  
si no lo encuentra retorna -1  
*/  
console.log(texto.indexOf("mun"));  
  
/*  
includes()  
Determina si existe una cadena dentro de otra cadena de texto,  
retornando true o false. (también aplica en arreglos para buscar elementos)  
*/  
console.log(texto.includes("chau")); // false
```

```
/**
charAt()
Permite acceder a un caracter en concreto de una cadena de texto através
de su índice.
*/
console.log(texto.charAt(3)) // a

/*
lastIndexOf()
Retorna la última posición de la ocurrencia del caracter pasado como parámetro,
si no lo encuentra retorna -1
*/
console.log(texto.lastIndexOf("o")) // 9
console.log(texto.lastIndexOf("w")) // -1
```

```
/*  
toLowerCase() -> convierte una cadena de texto en minúscula  
toUpperCase() -> convierte una cadena de texto en mayúscula  
*/  
  
console.log(texto.toLowerCase()); //hola mundo  
console.log(texto.toUpperCase()); //HOLA MUNDO
```

```
/*  
startsWith(palabra)  
Indica si una cadena de texto comienza con la palabra o texto pasado como parametro,  
retorna true o false  
  
endsWith(palabra)  
Similar al anterior pero indicando si finaliza con la palabra pasada como parametro  
*/  
console.log(texto.startsWith("Hol")); // true  
console.log(texto.endsWith("undo")); // true
```

Operador ===

Operador de igualdad estricta, revisa si dos operandos son estrictamente iguales, lo cual significa que son del mismo tipo de dato y que tienen el mismo valor.

Operador ==

Operador de igualdad simple, solo verifica que los valores sean iguales, no verifica el tipo de dato.

```
console.log("13" == 13); // true
```

```
console.log("13" === 13); // false
```

Estructura de control IF

```
if (condicion){  
    //La condicion es verdadera  
}else{  
    //La condicion es falsa  
}
```

```
if (condicion){  
    //Sentencias  
}else if(condicion_2){  
    //Sentencias  
}else if(condicion_3){  
    //Sentencias  
}else{  
    //No se cumple ninguna condición  
}
```


Estructuras iterativas, while y for

```
while(condicion){  
    //Mientras la condicion sea verdadera  
    //se ejecutan las instrucciones  
    //de aqui adentro  
}
```

```
for(let i = 0; i < 10; i++){  
    //Mientras la 'i' sea menor que 10  
    //Se ejecutan las sentencias de  
    //aquí dentro  
}
```

```
// FOR para recorrer un arreglo  
  
let arreglo = [1,2,3];  
  
for(let i = 0; i < arreglo.length; i++){  
    console.log("Posición: " + i + ", valor: " + arreglo[i]);  
}
```

```
/* FOR OF
Ejecuta un bloque de código para cada elemento
de una lista (arreglo)
*/

let lista = [1,2,3];

for(let elemento of lista){
    console.log(elemento * elemento);
}
```

Arreglos (array)

Son un conjunto de datos ordenados por posiciones, asociados a una sola variable. Los datos que contienen pueden ser de cualquier tipo, se declaran utilizando `[]` o la palabra “new”.

```
let arreglo = ["HOLA", 12, true];
```

```
let numeros = [1, 2, 3, 4, 5]
```

```
let nombres = new Array("Juan", "Andrea", "Marcos");
```

```
let numeros = [1, 2, 3]

// Agregar elementos al final de arreglo
numeros.push(19);

// Eliminar el último elemento
numeros.pop();

// Agregar un elemento al inicio
numeros.unshift(8);

// Eliminar el primer elemento
numeros.shift();
```

Array asociativo

Es un array cuyos índices no son numéricos sino cadenas de texto (claves), están organizados por claves en lugar de por índices.

```
let auto = {  
  'marca': 'Audi',  
  'modelo': 'A3',  
  'color': 'Negro',  
  'Anio': 2024  
}  
  
// Podemos agregar elementos agregando la pareja clave:valor  
auto['usado'] = false;
```

```
let auto = {  
  'marca': 'Audi',  
  'modelo': 'A3',  
  'color': 'Negro',  
  'Anio': 2024  
}  
  
// Podemos agregar elementos agregando la pareja clave:valor  
auto['usado'] = false;  
  
console.log(auto);
```



```
▼ Object { marca: "Audi", modelo: "A3", color: "Negro", Anio: 2024, usado: false }  
  Anio: 2024  
  color: "Negro"  
  marca: "Audi"  
  modelo: "A3"  
  usado: false
```

```
// Para acceder a su valores podemos utilizar sus claves  
  
console.log(auto['marca']);  
  
// Ó tambien la notación de punto  
  
console.log(auto.marca);
```


Clases

```
// Clases
class Persona{
    constructor(nombre, apellido, edad){
        this.nombre = nombre;
        this.apellido = apellido;
        this.edad = edad;
    }
}

// Creación de objetos de tipo Persona:
let persona_1 = new Persona("Juan", "Perez", 30);
let persona_2 = new Persona("Andrea", "Sosa", 31);

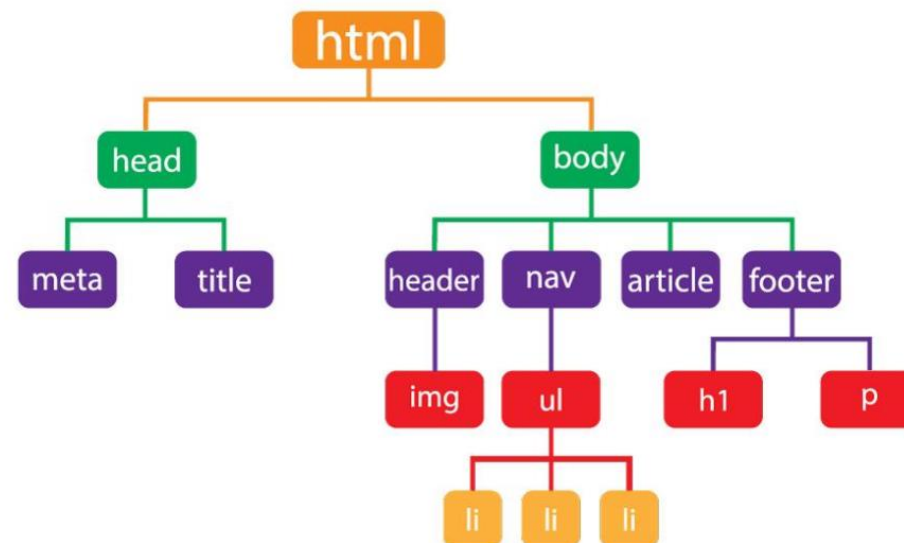
console.log(persona_1.nombre);
console.log(persona_1.apellido);

persona_1.nombre = "Sebastian";

if(persona_1.edad > 17){
    console.log(persona_1.nombre + " es mayor de edad");
}
```

DOM – Modelo de Objetos del Documento

DOM es la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina **árbol DOM**.



Seleccionar elementos del DOM con JavaScript

```
document.querySelector(selector)
```

```
/*
```

```
Nos retorna uno o ningún elemento que concuerde con el selector que le pasamos  
como parámetro, utiliza los mismos selectores que CSS:
```

- id
- clase
- etiqueta

```
Por ejemplo:
```

```
document.querySelector("#identificador")  
document.querySelector(".clase")  
document.querySelector("etiqueta_html")
```

```
*/
```

```
document.querySelectorAll()
```

```
/*
```

Nos retorna de 0 a todos los elementos HTML que concuerden con el selector CSS que le pasamos como parámetro.

Por ejemplo:

```
<a href=""> GOOGLE </a>
```

```
<a href=""> BING </a>
```

```
<a href=""> BUSCADOR_X </a>
```

```
let enlaces = document.querySelectorAll("a");
```

Selecciona todos los elementos <a> del documento HTML

Lo que obtengo es una lista con todos ellos

```
*/
```

Podemos agregar o eliminar clases a un elemento seleccionado desde JavaScript:

```
// Agrega las clases al elemento seleccionado
document.querySelector("#identificador").classList.add("clase_1", "clase_2");

//Elimina la clase del elemento seleccionado
document.querySelector("#identificador").classList.remove("clase_1");
```

Podemos agregar o eliminar atributos a un elemento seleccionado desde JavaScript:

```
// Agregar un atributo y su valor al elemento seleccionado  
let elemento = document.querySelector("#identificador");  
elemento.setAttribute("nombre_atributo", "valor");
```

```
//Eliminar un atributo del elemento seleccionado  
let elemento_2 = document.querySelector("#identificador");  
elemento_2.removeAttribute("nombre_atributo")
```

Crear un elemento HTML desde JavaScript:

```
// document.createElement("etiqueta_html");  
// Por ejemplo:  
  
// Creo un elemento de tipo <p>  
let parrafo = document.createElement("P");  
  
// Agrego el texto del parrafo  
parrafo.textContent = "Contenido del párrafo";  
  
// Agrego clases CSS al parrafo  
parrafo.classList.add("clase_1", "clase_2");  
  
// Agrego el parrafo al documento HTML, por ejemplo,  
// dentro de un <section> seleccionado:  
  
let ubicacion = document.querySelector("section");  
  
// Agrego el parrafo creado en el section seleccionado, como un hijo más  
ubicacion.appendChild(parrafo);
```


EVENTOS

Los eventos son cosas que suceden en el sistema, por ejemplo, en un documento HTML, los cuales de ser controlados (escuchados) nos permiten tomar acciones a partir de ellos.

Existen eventos del mouse, eventos cuando se carga una página web, eventos de click sobre botones o etiquetas, eventos de teclado, eventos de formularios, como, por ejemplo, el “submit” para enviar los datos a un servidor, etc.

Para controlar los eventos desde JavaScript tenemos que:

- 1) Seleccionar el elemento que es capaz de lanzar el evento que queremos escuchar
- 2) Escuchar el evento deseado con la función “**addEventListener()**”
- 3) Asociar al evento una función de JS para que cuando suceda el evento, se ejecute la función
- 4) Definir la función que asociamos en el paso anterior


```
// Escuchar el evento click de un boton

// 1. Seleccionar el boton que queremos escuchar el evento click
let boton = document.querySelector("#id_boton");

// 2. Escuchar el evento click y asociar una funcion para cuando suceda
boton.addEventListener("click", funcion_a_ejecutar);

// 3. Definir la función asociada anteriormente
function funcion_a_ejecutar(evento){
    console.log("Usted hizo click en el boton");
}
```

Ejemplo para el evento cuando carga el HTML de una página web:

```
// Escuchar el evento de carga del HTML de una página web

document.addEventListener("DOMContentLoaded", funcion_iniciar);

function funcion_iniciar(evento){
    console.log("Ha cargado la página web!!");
}
```

Escuchar el evento SUBMIT de un botón en un formulario HTML:

```
<form id="formulario" >  
  <input type="text" id="nombre">  
  <input type="text" id="apellido">  
  
  <input type="submit" value="Enviar">  
</form>
```

Formulario

```
// Seleccionar el formulario por su identificador
let formulario = document.querySelector("#formulario");

// Escuchar el evento submit del formulario y asociarlo a una funcion
formulario.addEventListener("submit", enviar_datos);

// Definir la función "enviar_datos"
function enviar_datos(evento){
    // Paramos el envio de datos al servidor para analizarlos
    evento.preventDefault();

    //Obtener los campos del formulario
    let nombre = document.querySelector("#nombre").value;
    let apellido = document.querySelector("#apellido").value;

    // Mostramos lo ingresado en el formulario en la consola
    console.log("Usted ingreso: " + nombre);
    console.log("Usted ingreso: " + apellido);
}
```

localStorage

Es una propiedad para acceder al objeto Storage del navegador, tiene la función de almacenar datos de manera local.

Los datos aquí almacenados no tienen fecha de expiración.

Se utilizan ***clave:valor*** que son siempre cadenas de texto, dado que localStorage solo almacena string.

Para guardar datos en localStorage:

```
localStorage.setItem("clave", "valor");
```

Para obtener datos desde localStorage:

```
localStorage.getItem("clave");
```

Si deseo guardar un objeto en localStorage, tengo que convertirlo en un string, utilizando la función **JSON.stringify()**, y cuando obtengo desde localStorage tengo que hacer la operación inversa, convertir de string a objeto, por ejemplo:

```
const producto = {  
  nombre : "Monitor ViewSonic",  
  precio : 350  
}  
  
// Convierto el objeto producto para guardarlo en localStorage  
const productoString = JSON.stringify(producto);  
  
// Guardo el producto convertido a string en localStorage  
localStorage.setItem("producto", productoString);
```

```
// Para obtener el producto desde localStorage
// tengo que hacer la operación inversa, o sea
// convertir el producto de string a objeto

// Obtengo el producto desde localStorage
const prod = localStorage.getItem("producto");

// Convierto de string a object
const objeto = JSON.parse(prod);
```


FIN.