


Import / Export

`import` y `export` son dos palabras clave utilizadas en JavaScript para trabajar con módulos, que son piezas de código independientes y reutilizables. `export` se utiliza para declarar qué partes del código de un archivo deben ser accesibles desde otros archivos, mientras que `import` se utiliza para traer (importar) esas partes exportadas de otros archivos para su uso en el archivo actual. Esto permite una estructura de código modular y modularidad en JavaScript, lo que facilita la organización, mantenimiento y reutilización del código.

Primero asegúrate que en el `HTML` enlaces o llames al archivo principal `JS` con el parámetro `module` definido de la siguiente manera:


index.html

 Copiar código

```
<script src="operacion13.js" type="module"></script>
```

Ahora en el archivo secundario `exporta` las funciones:


funciones13.js

 Copiar código

```
function sumar (a,b){  
  return a+b;  
}  
  
function restar (a,b){  
  return a-b;  
}  
  
export{sumar, restar};
```

Finalmente en el archivo `js` principal `importa` las funciones

operaciones13.js

 Copiar código

```
import { sumar, restar } from './funciones13.js'

console.log("La suma es " + sumar(17,3));
```

POO

La **Programación Orientada a Objetos** (POO) en JavaScript implica trabajar con **"objetos"** que combinan **datos y funciones**. Puedes definir **clases** para crear objetos y organizar tu código de manera más estructurada y reutilizable. Esto facilita el mantenimiento y la escalabilidad del código al tiempo que permite características como la herencia y el polimorfismo para crear relaciones jerárquicas entre objetos.

Clases y Objetos: Las clases definen la estructura y el comportamiento de los objetos, que son instancias de esas clases.

Encapsulamiento: Oculta la implementación interna de un objeto y solo muestra las operaciones públicas. Esto se logra mediante el uso de métodos y propiedades privadas y públicas.

Abstracción: Permite modelar objetos del mundo real como objetos de software, enfocándose en las características esenciales y omitiendo los detalles irrelevantes.

Herencia: Permite que una clase herede atributos y métodos de otra clase. Esto fomenta la reutilización de código y facilita la creación de jerarquías de clases.

Polimorfismo: Permite que objetos de diferentes clases respondan al mismo mensaje de diferentes maneras. Esto se logra mediante la capacidad de una clase para sobrescribir métodos

Métodos y Propiedades: Los métodos son funciones asociadas a un objeto y las propiedades son valores asociados a un objeto.


Constructor: Un método especial utilizado para inicializar un objeto cuando se crea una instancia de una clase.

Instanciación: El proceso de crear un objeto a partir de una clase se conoce como instanciación.

Mensaje: La forma en que un objeto solicita que se realice una operación específica.

Interfaz: Define un conjunto de métodos que una clase debe implementar. Ayuda a establecer un contrato entre clases.


js

 Copiar código

```
// Construyendo un objeto en Js
const perro = {
  nombre: 'Paco', // Atributos
  edad: 4,        // Atributos
  color: 'negro', // Atributos
  raza: 'Pincher' // Atributos
};

console.log(perro.color);
console.log(perro.raza);
console.log(perro.nombre);
console.log(perro.edad);
```

js

 Copiar código

```
// Construyendo una clase
class animal{
  constructor(nombre, edad, color){
    this.nombre = nombre;
    this.edad = edad;
    this.color = color;
  }
}

// Podemos instanciar objetos usando la clase anterior :
const gato = new animal('Bruno', 3, 'Gris');
console.log(gato);

const loro = new animal('Orus', 1, 'Verde');
```

```
console.log(loro);
```

22.03.2024 REPASO

```
const conejo = new animal('Boss', 2, 'Blanco');  
console.log(conejo);
```

[Refuerza este tema aquí](#)

Bryan Hernández | Telento Tech DWFSV2-42 | 2024

—