Relaciones de objetos y métodos

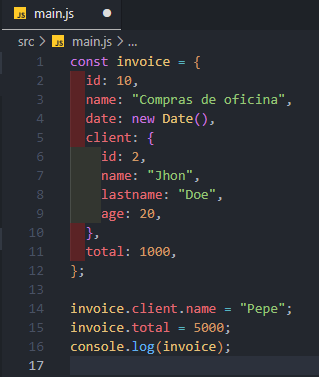
En esta sección, se explicará las relaciones entre objetos y métodos de un objeto en JavaScript, con la finalidad de modificar el valor de un objeto anidado y definir una función dentro de un objeto para crear un método.

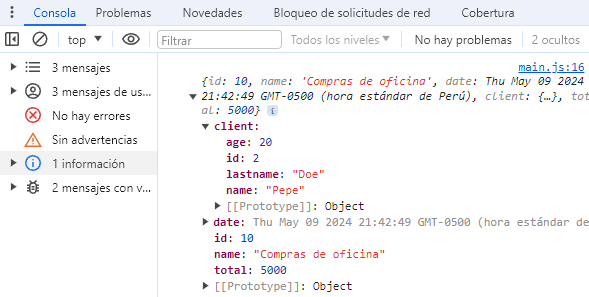
Relaciones entre objetos

En JavaScript, un objeto puede contener otro objeto como uno de sus atributos, lo que se conoce como anidación de objetos. Esto permite construir una estructura jerárquica, facilitando la modularización de datos y la organización de la información.

Para modificar el valor de un atributo dentro de un objeto anidado, se debe especificar primero el objeto contenedor, luego el objeto anidado y el atributo que se desea modificar.

En el siguiente ejemplo, el objeto **invoice** está anidado con un objeto **client**, el cual contiene los siguientes atributos: **id**, **name**, **lastname** y **age**.





Métodos de un objeto

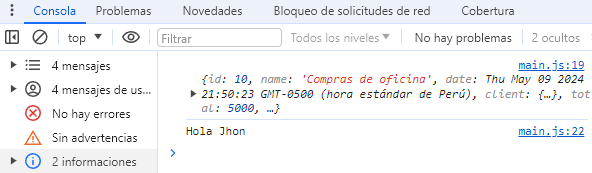
Dentro de un objeto en JavaScript, se puede definir funciones como atributos. Estas funciones se conocen como métodos del objeto. Los métodos permiten que un objeto realice operaciones y manipule sus propios datos de manera organizada.

Para definir un método, se puede utilizar la palabra clave **function**. El término **this** dentro de un método hace referencia al objeto en el que está definido el método. Es importante tener en cuenta que los métodos se invocan a través de un objeto específico.

A continuación, se presenta un ejemplo de cómo definir un método dentro de un objeto y cómo invocarlo:







Función de flecha dentro de un objeto

También es posible definir una función de flecha como método dentro de un objeto. Sin embargo, al usar una función de flecha, el uso de **this** cambia. En lugar de **this**, se debe utilizar el nombre del objeto para hacer referencia a sus propiedades.



const invoice = {
  id: 10,
  name: "Compras de oficina",
  date: new Date(),
  client: {
    id: 2,
    name: "Jhon",
    lastname: "Doe",
    age: 20,
  },
  total: 1000,
  greeting: () => {
    return `Hola ${invoice.client.name}`;
  },
};

// invoice.client.name = "Pepe";
invoice.total = 5000;
console.log(invoice);

const greeting = invoice.greeting();
console.log(greeting);

