

Bootcamp de Full Stack

Bienvenidos a la clase N°25

- Objetos literales
- Instrucción for...in
- Instrucción for...of
- Conversión de object=array
- Destructuring
- Operador spread
- Operador rest

Javascript

Objetos Literales

Un **objeto literal** es una forma sencilla de agrupar datos y/o funciones dentro de una sola estructura. Se escribe entre llaves { } y contiene **pares de clave y valor**, donde cada clave representa una propiedad o acción, y cada valor puede ser un dato (como un número o texto) o una función. Esto se lo suele utilizar para:

- Representar entidades con características (como un usuario, un producto, etc.).
- Agrupar datos relacionados en una sola variable.

Los objetos permiten agregar, modificar o eliminar propiedades libremente. Para evitar este comportamiento y proteger el contenido del objeto, se puede usar **Object.freeze()**, que lo vuelve inmutable y evita cualquier cambio en sus propiedades.

Object {
 key: value,
 key: value,
 key: function() { },

Javascript

Instrucción for...in

La instrucción **for...in** permite recorrer todas las propiedades enumerables de un objeto, accediendo a cada clave (nombre de propiedad) para leer o manipular su valor. Es útil cuando querés obtener las claves de un objeto y trabajar con ellas, pero no se recomienda usarlo en arrays porque puede generar un comportamiento no deseado al incluir propiedades heredadas y no garantizar el orden de los elementos.

```
const object = {  
  a: 10,  
  b: true,  
};  
  
for (const key in object) {  
  if (object.hasOwnProperty(key)) {  
    const keyName = key;  
    const keyValue = object[key];  
    console.log(keyName);      // Imprime el nombre de la propiedad  
    console.log(keyValue);     // Imprime el valor de la propiedad  
  }  
}
```

Javascript

Instrucción for...of

La instrucción **for...of** permite recorrer los valores de cualquier **objeto iterable**, como arrays, strings, Maps o Sets. Es ideal para acceder directamente a cada elemento en secuencia, garantizando un orden correcto y sin incluir propiedades adicionales. Por eso, es la opción recomendada para iterar arrays y otras estructuras iterables.

```
let string = "Hola";

for (const value of string) {
  console.log(value);    // Imprime el valor de la iteración. Es decir, el caracter.
}
```

Javascript

Conversión De Object=Array

```
const array = Object.keys(object);
```

Devuelve un array con los nombres (claves) de todas las propiedades propias y enumerables del objeto. Es ideal cuando solo necesitas trabajar con las claves.

```
const array = Object.values(object);
```

Devuelve un array con los valores de todas las propiedades propias y enumerables del objeto. Útil cuando te interesan solo los valores.

```
const array = Object.entries(object);
```

Devuelve un array de arrays, donde cada subarray contiene un par [clave, valor]. Es perfecto para iterar tanto claves como valores al mismo tiempo.

BREAK

Descansemos 10 minutos



Javascript

Destructuring

El **destructuring** es una sintaxis que permite extraer valores de arrays o propiedades de objetos y asignarlos a variables de forma concisa y legible. Facilita el acceso a datos complejos sin tener que referenciar manualmente cada elemento o propiedad.

```
const object = {  
  a: "hola",  
  b: true,  
};  
const { a, b } = object;  
console.log(a); // Imprime el valor de la variable "a" → "hola"
```

```
const colors = ["rojo", "verde", "azul"];  
const [ first, second ] = colors;  
console.log(first); // Imprime el valor de la variable "first" → "rojo"
```

Javascript

Operador Spread

El operador spread permite expandir los elementos de un array u objeto. Se utiliza para copiar, combinar o pasar datos de forma individual, especialmente útil en arrays, objetos y llamadas a funciones.

Fusionar arrays: consiste en combinar sus elementos en uno solo.

```
const array1 = [1, 2, 3];  
const array2 = [4, 5, 6];  
const newArray = [...array1, ...array2];  
console.log(newArray);
```

Fusionar objetos: implica combinar sus propiedades en un nuevo objeto.

```
const object1 = { a: 1, b: 2 };  
const object2 = { b: 99, c: 3 };  
const newObject = { ...object1, ...object2 };  
console.log(newObject);
```


Javascript

Operador Rest

El **operador rest** permite agrupar múltiples elementos o propiedades en una sola variable. Se utiliza para capturar el "resto" de los valores en arrays o objetos, es útil en parámetros de funciones y destructuring.

- Rest en destructuring de objetos: agrupa las propiedades restantes

```
const object = { a: "hola", b: true, c: 50 };
const { a, ...rest } = object;
console.log(a, rest);      // Imprime el valor de la variable "a" → "hola" y "rest" → {b: true, c: 50}
```

- Rest en destructuring de arrays: agrupa los elementos restantes

```
const colors = ["rojo", "verde", "azul"];
const [ a, ...rest ] = colors;
console.log(a, rest);      // Imprime el valor de la variable "a" → "rojo" y "rest" → ["verde", "azul"]
```

- Rest en parámetros de funciones: permite recibir un número indefinido de argumentos como un array

```
const add = (...numbers) => numbers.reduce((accumulator, number) => accumulator + number, 0);
console.log(add(10, 50, 15));      // Imprime el retorno de la función "add" → 75
console.log(add(5, 15, 20, 7, 3)); // Imprime el retorno de la función "add" → 50
```

CIERRE DE CLASE

Continuaremos en la próxima clase

