**Set**

Un Set en JavaScript es una colección de valores únicos. A diferencia de los arrays, un Set no permite duplicados, lo que significa que cada valor solo puede aparecer una vez en la colección. Puedes almacenar tipos de datos diferentes.

**Características**

* **Valores Únicos:** Un Set automáticamente elimina los valores duplicados. Si intentas agregar un valor que ya está en el Set, no se añadirá de nuevo.
* **Orden de Inserción:** Los valores en un Set están ordenados en el orden en que se insertaron. Aunque no se puede acceder a los elementos por índice como en un array, se mantienen en el orden de inserción.
* **Iterabilidad:** Un Set es iterativo, lo que significa que puedes usar bucles for...of y otros métodos de iteración como forEach para recorrer los elementos.
* **No Permite Duplicados:** No permite valores duplicados. Si intentas añadir un valor que ya existe, el Set no cambiará.
* **Mutabilidad:** Un Set es mutable. Puedes agregar, eliminar y modificar sus elementos después de la creación.

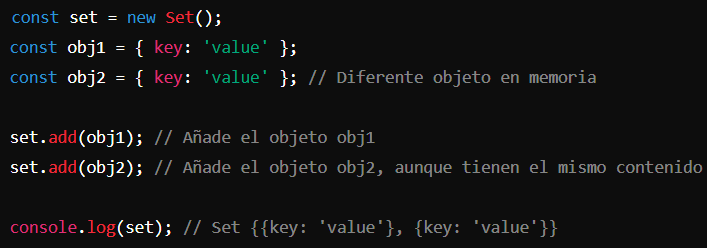
**Cómo Funciona la Adición de Elementos en un Set**

* **Para Tipos Primitivos (como números, cadenas, booleanos):** Un Set compara los valores utilizando la comparación estricta (===). Esto significa que los valores son únicos basándose en su valor real. Dos valores primitivos idénticos son considerados el mismo valor.

**Texto

Descripción generada automáticamente**

* **Para Objetos, Arrays y Funciones:** Un Set compara las referencias de estos elementos. Es decir, dos objetos o arrays distintos, aunque tengan el mismo contenido, son considerados únicos porque tienen referencias diferentes en memoria.

****

**Métodos Comunes del Set**

* **add(value):** Agrega un nuevo valor al Set. Si el valor ya está presente, no se agrega (depende del tipo de dato). **Retorno**: El propio Set, lo que permite encadenar llamadas a métodos.

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

* **delete(value):** Elimina un valor del Set. **Retorno**: true si el valor fue eliminado, false si el valor no estaba presente.

****

* **has(value):** Verifica si un valor está presente en el Set. **Retorno**: true si el valor está presente, false si no lo está.

****

* **clear():** Elimina todos los valores del Set, no toma argumentos. **Retorno:** undefined.
* **size():** Propiedad que devuelve el número de elementos en el Set, no toma argumentos. **Retorno**: Un número entero que representa la cantidad de elementos en el Set.
* **forEach(callbackFn[, thisArg]):** Ejecuta una función callbackFn para cada valor del Set en el orden de inserción. Puedes proporcionar un segundo argumento thisArg para definir el contexto de this en la función callback. **Retorno**: undefined.

**Argumentos**

**callbackFn** - La función a ejecutar para cada valor. Recibe tres argumentos: el valor actual, el valor mismo, y el Set sobre el cual se está iterando.

**thisArg (opcional)** - Valor que se usará como this cuando se ejecute el callbackFn.

**Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media**

* **values():** Devuelve un nuevo objeto iterador que contiene los valores del Set en el orden de inserción, no tiene argumentos. **Retorno**: Un objeto iterador de tipo Set Iterator que puede ser recorrido con bucles for...of.

Texto

Descripción generada automáticamente

* **keys():** Devuelve un nuevo objeto iterador que contiene las claves del Set. En un Set, las claves y los valores son equivalentes, por lo que se comporta de manera similar a values(), no tiene argumentos. **Retorno**: Un objeto iterador de tipo Set Iterator.

Texto

Descripción generada automáticamente

* **entries():** Devuelve un nuevo objeto iterador que contiene una matriz de pares [valor, valor] para cada elemento del Set. Es útil para trabajar con estructuras de datos similares a los arrays, aunque en un Set cada elemento es su propio valor, no tiene argumentos. **Retorno**: Un objeto iterador de tipo Set Iterator.

**Texto

Descripción generada automáticamente**

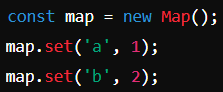
**Map**

Un Map en JavaScript es una colección de pares clave-valor en la que cada clave es única y está asociada a un valor. A diferencia de los objetos ({}), los Map tienen algunas características adicionales que pueden ser útiles en diversos escenarios.

* **Claves Únicas:** Cada clave en un Map es única. Si intentas establecer un valor para una clave que ya existe, el valor anterior se sobrescribe.
* **Cualquier Tipo de Clave:** A diferencia de los objetos, que solo pueden usar cadenas de texto o símbolos como claves, los Map pueden usar cualquier tipo de dato como clave, incluidos objetos, funciones y valores primitivos.
* **Orden de Inserción:** Los elementos en un Map están ordenados en el orden en que fueron insertados. Esto significa que al iterar sobre un Map, los pares clave-valor se devolverán en el orden en que se agregaron.
* **Iterabilidad:** Los Map son iterables, lo que significa que puedes usar bucles for...of y otros métodos de iteración para recorrer sus elementos.
* **Tamaño:** La propiedad .size de un Map indica el número de pares clave-valor que contiene.
* **Mutabilidad:** Los Map son mutables; puedes agregar, eliminar y modificar pares clave-valor después de la creación.

**Métodos Comunes del Set**

* **set(key, value):** Agrega un nuevo par clave-valor al Map. Si la clave ya existe, actualiza el valor asociado a esa clave. **Retorno**: El propio Map, lo que permite encadenar llamadas a métodos.



* **get(key):** Obtiene el valor asociado a una clave específica. **Retorno:** El valor asociado a la clave. Devuelve undefined si la clave no está presente en el Map.



* **has(key):** Verifica si una clave está presente en el Map. **Retorno**: true si la clave está presente, false si no lo está.



* **delete(key):**  Elimina el par clave-valor asociado a una clave específica. **Retorno**: true si el par clave-valor fue eliminado, false si la clave no estaba presente.



* **clear():** Elimina todos los pares clave-valor del Map. **Retorno**: undefined
* **size():** Propiedad que devuelve el número de pares clave-valor en el Map. **Retorno**: Un número entero que representa la cantidad de pares clave-valor en el Map.
* **forEach():** Ejecuta una función callbackFn para cada par clave-valor en el Map en el orden de inserción. Puedes proporcionar un segundo argumento thisArg para definir el contexto de this en la función callback. **Retorno**: undefined.

**Argumentos**

**callbackFn** - La función a ejecutar para cada par clave-valor. Recibe cuatro argumentos: el valor, la clave, el Map sobre el cual se está iterando, y el thisArg opcional.

**thisArg** (opcional) - Valor que se usará como this cuando se ejecute el callbackFn.

Texto

Descripción generada automáticamente

* **keys():** Devuelve un nuevo objeto iterador que contiene las claves del Map en el orden de inserción. **Retorno**: Un objeto iterador de tipo Map Iterator que puede ser recorrido con bucles for...of

Texto

Descripción generada automáticamente

* **values ():** Devuelve un nuevo objeto iterador que contiene los valores del Map en el orden de inserción. **Retorno**: Un objeto iterador de tipo Map Iterator que puede ser recorrido con bucles for...of.

Texto

Descripción generada automáticamente

* **entries():** Devuelve un nuevo objeto iterador que contiene una matriz de pares [clave, valor] para cada elemento del Map. **Retorno**: Un objeto iterador de tipo Map Iterator que puede ser recorrido con bucles for...of.

Texto

Descripción generada automáticamente