

DESARROLLO WEB EN ENTORNO CLIENTE

TEMA3: **COLECCIONES CON CLAVE**

Índice

2

- Colecciones con clave
- Mapas – objeto map
- Comparar object y map
- SET – objeto set
- Conversión entre arreglo y Set
- Comparar array y set

1. Colecciones con clave

- Este capítulo presenta colecciones de datos que están indexadas por una clave.
- Los objetos `Map` y `Set` contienen elementos que son iterables en el orden de inserción.

1. Mapas – Objeto map

4

- ECMAScript 2015 introduce una nueva estructura de datos para asociar claves con valores. Un objeto [Map](#) (en-US) es un mapa de clave/valor simple y puedes iterar sobre sus elementos en el orden en que fueron insertados.
- El siguiente código muestra algunas operaciones básicas con un Map. Consulta también la página de referencia de Map (en-US) para obtener más ejemplos y la API completa.
- Puedes usar un bucle [for...of](#) para devolver un arreglo de [key, value] para cada iteración.

1. Mapas – Objeto map

5

JS

```
let sayings = new Map();
sayings.set("dog", "woof");
sayings.set("cat", "meow");
sayings.set("elephant", "toot");
sayings.size; // 3
sayings.get("dog"); // woof
sayings.get("fox"); // undefined
sayings.has("bird"); // false
sayings.delete("dog");
sayings.has("dog"); // false

for (let [key, value] of sayings) {
  console.log(key + " goes " + value);
}
// "cat goes meow"
// "elephant goes toot"

sayings.clear();
sayings.size; // 0
```

1. Mapas – Objeto map

Comparar Object y map

6

- Comparar Object y map
- Tradicionalmente, los objetos se han utilizado para asignar cadenas a valores. Los objetos te permiten establecer claves a valores, recuperar esos valores, eliminar claves y detectar si algo está almacenado en una clave. Los objetos Map, sin embargo, tienen algunas ventajas más que los hacen mejores.

1. Mapas – Objeto map

Comparar Object y map

7

- Las **claves de un Object** son Cadenas o Símbolos, donde *pueden tener cualquier valor para un Map.*
- Puedes obtener el **size** de un Map fácilmente, mientras que tienes que realizar un seguimiento manual del tamaño de un Object.
- La **iteración** de mapas está en el orden de inserción de los elementos.
- Un Object tiene un **prototipo**, por lo que hay claves predeterminadas en el mapa. (Esto se puede omitir usando `map = Object.create(null)`).

1. Mapas – Objeto map

Comparar Object y map

8

- Estos tres consejos te pueden ayudar a decidir si usar un **Map** o un **Object**:
- Usa **mapas sobre objetos** cuando las claves sean desconocidas hasta el momento de la ejecución, y cuando todas las claves sean del mismo tipo y todos los valores sean del mismo tipo.
- Utiliza **mapas** si es necesario almacenar valores primitivos como claves porque el objeto trata cada clave como una cadena, ya sea un valor numérico, un valor booleano o cualquier otro valor primitivo.
- Usa **objetos** cuando haya lógica que opere en elementos individuales.

1. Mapas – Objeto map

Comparar Object y map

9

Map – es una colección de valores con clave.

Métodos y propiedades:

- `new Map([iterable])` – crea el mapa, con un iterable (p.ej. array) de pares [clave,valor] para su inicialización.
- `map.set(clave, valor)` – almacena el valor para la clave.
- `map.get(clave)` – devuelve el valor de la clave: será undefined si la clave no existe en Map.
- `map.has(clave)` – devuelve true si la clave existe, y false si no existe.
- `map.delete(clave)` – elimina del map el elemento con esa clave.
- `map.clear()` – vacía el Map.
- `map.size` – devuelve la cantidad de elementos del Map.

La diferencia con un Objeto regular:

Cualquier clave. Los objetos también pueden ser claves.

Métodos adicionales convenientes, y la propiedad size.

2. SET –Objeto set

10



- ❑ Los objetos **Set** son colecciones de valores. Puedes iterar sus elementos en el orden en que se insertaron. *Un valor en un Set solo puede aparecer una vez; es **único** en la colección del Set.*
- ❑ El siguiente código muestra algunas operaciones básicas con un Set. Además, consulta la página de referencia de Set para obtener más ejemplos y la API completa.

2. SET –Objeto set

11



JS




```
let mySet = new Set();
mySet.add(1);
mySet.add("algún texto");
mySet.add("foo");

mySet.has(1); // true
mySet.delete("foo");
mySet.size; // 2

for (let item of mySet) console.log(item);
// 1
// "algún texto"
```

2. SET –Objeto set

12



Un **Set** es una colección de tipo especial: “conjunto de valores” (sin claves), donde cada valor puede aparecer solo una vez.

Sus principales métodos son:

- `new Set([iterable])` – crea el set. El argumento opcional es un objeto iterable (generalmente un array) con los valores para inicializarlo.
- `set.add(valor)` – agrega un valor, y devuelve el set en sí.
- `set.delete(valor)` – elimina el valor, y devuelve true si el valor existía al momento de la llamada; si no, devuelve false.
- `set.has(valor)` – devuelve true si el valor existe en el set, si no, devuelve false.
- `set.clear()` – elimina todo el contenido del set.
- `set.size` – es la cantidad de elementos.

2. SET –Objeto set

13



También tiene los mismos métodos que Map:

- `set.keys()` – devuelve un iterable para las claves.
- `set.values()` – lo mismo que `set.keys()`, por su compatibilidad con Map.
- `set.entries()` – devuelve un iterable para las entradas [clave, valor], por su compatibilidad con Map.

```
1 let set = new Set(["oranges", "apples", "bananas"]);
2
3 for (let value of set) alert(value);
4
5 // lo mismo que forEach:
6 set.forEach((value, valueAgain, set) => {
7     alert(value);
8 });
```

2. Conversión entre arreglo y Set

14



- Puedes crear un Array a partir de un Set usando `Array.from` o el operador de propagación (en-US). Además, el constructor Set acepta un Array para convertirlo en la otra dirección.
- Nota: Recuerda que los objetos Set almacenan valores únicos, por lo que cualquier elemento duplicado de un arreglo se elimina al realizar la conversión.

JS

```
Array.from(mySet);  
[...mySet2];  
  
mySet2 = new Set([1, 2, 3, 4]);
```

2. Conversión entre arreglo y Set

15

- ✓ Eliminar elementos Array por valor

(`arr.splice(arr.indexOf(val), 1)`) es muy lento.

Los objetos Set te permiten eliminar elementos por su valor.

- ✓ Con un arreglo, tendrías que empalmar (con `splice`) en función del índice de un elemento.
- ✓ El valor NaN no se puede encontrar con `indexOf` en un arreglo.
- ✓ *Los objetos Set almacenan valores únicos. No es necesario que realices un seguimiento manual de los duplicados.*

1. Map y Set

16



A.1.1. Filtrar miembros únicos del array

Cree una función `unique(arr)` que debería devolver un array con elementos únicos de `arr`. Parte de un array con valores repetidos, para comprobarlo.

A.1.2. Filtrar anagramas

Anagramas son palabras que tienen el mismo número de letras, pero en diferente orden. Escriba una función `aclean(arr)` que devuelva un array limpio de anagramas. Parte del array:

```
let arr = ["nap", "teachers", "cheaters", "PAN", "ear", "era", "hectares"];
```

A.1.3. Claves iterables

Nos gustaría obtener un array de `map.keys()` en una variable y luego aplicarle métodos específicos de array, ej. `.push`. Partiendo de:

```
let map = new Map(); map.set("name", "John");
```

Extrae las claves y aplícales `push`, haciendo una conversión previa a `Array`.