Practica 04-03b El objeto Map

Nombre		Curso	
Apellidos		Fecha	

Map es, al igual que Objet, una colección de datos identificados por claves. La principal diferencia es que Map permite claves de cualquier tipo.

Los métodos y propiedades son:

- new Map() – crea el mapa.
- map.set(clave, valor)) – almacena el valor asociado a 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 en map, false si no existe.
- map.delete(clave) – elimina el elemento con esa clave.
- map.clear() – elimina todo de map.
- map.size – tamaño, devuelve la cantidad actual de elementos.

ejemplo1

```
let map = new Map();

map.set('1', 'str1');   // un string como clave
map.set(1, 'num1');     // un número como clave
map.set(true, 'bool1'); // un booleano como clave

// ¿recuerda el objeto regular? convertiría las claves a string.
// Map mantiene el tipo de dato en las claves, por lo que estas dos son diferentes:
alert( map.get(1)   ); // 'num1'
alert( map.get('1') ); // 'str1'

alert( map.size ); // 3
```

ejemplo2

```
let recipeMap = new Map([
  ['pepino', 500],
  ['tomates', 350],
  ['cebollas',   50]
]);

alert( map.get('pepino')   ); // 500
alert( map.get('tomates')   ); // 350
```

Podemos ver que, a diferencia de los objetos, las claves no se convierten en strings. Cualquier tipo de clave es posible en un Map.

map[clave] no es la forma correcta de usar Map

Aunque map[clave] también funciona (por ejemplo podemos establecer map[clave] = 2), esto es tratar a map como un objeto JavaScript simple, lo que implica tener todas las limitaciones correspondientes (que solo se permita string/symbol como clave, etc.).

Por lo tanto, debemos usar los métodos de Map: set, get y demás.

Uso de objetos como clave

podemos usar objetos como claves.

ejemplo3

```
let john = { name: "John" };

// para cada usuario, almacenemos el recuento de visitas
let visitsCountMap = new Map();

// john es la clave para el Map
visitsCountMap.set(john, 123);

alert( visitsCountMap.get(john) ); // 123
```

El uso de objetos como claves es una de las características de Map más notables e importantes. Esto no se aplica a los objetos: una clave de tipo string está bien en un Object, pero no podemos usar otro Object como clave.

ejemplo4

```
let john = { name: "John" };
let ben = { name: "Ben" };

let visitsCountObj = {}; // intenta usar un objeto

visitsCountObj[ben] = 234; // intenta usar el objeto ben como clave
visitsCountObj[john] = 123; // intenta usar el objeto john como clave, el objeto ben es reemplazado

// Esto es lo que se escribió!
alert( visitsCountObj["[object Object]"] ); // 123
```

Como visitsCountObj es un objeto, convierte todas los objetos como john y ben en el mismo string "[objeto Objeto]". Definitivamente no es lo que queremos.

Cómo Map compara las claves

Para probar la equivalencia de claves, Map utiliza el algoritmo SameValueZero. Es aproximadamente lo mismo que la igualdad estricta ===, pero la diferencia es que NaN se considera igual a NaN. Por lo tanto, NaN también se puede usar como clave.

Este algoritmo no se puede cambiar ni personalizar.

Encadenamiento

Cada llamada a map.set devuelve map en sí, así que podamos “encadenar” las llamadas:

ejemplo5

```
let mapconjunto = map.set('1', 'str1')
  .set(1, 'num1')
  .set(true, 'bool1');
```

Iteración sobre Map

Para recorrer un map, hay 3 métodos:

- map.keys() –- devuelve un iterable con las claves.
- map.values() -— devuelve un iterable con los valores.
- map.entries() -— devuelve un iterable para las entradas [clave, valor]. Es el que usa por defecto en for..of.

Por ejemplo:

ejemplo6

```
let recipeMap = new Map([
  ['pepino', 500],
  ['tomates', 350],
  ['cebollas',   50]
]);

// iterando sobre las claves (verduras)
for (let vegetable of recipeMap.keys()) {
  alert(vegetable); // pepino, tomates, cebollas
}

// iterando sobre los valores (precios)
for (let amount of recipeMap.values()) {
  alert(amount); // 500, 350, 50
}

// iterando sobre las entradas [clave, valor]
for (let entry of recipeMap) { // lo mismo que recipeMap.entries()
  alert(entry); // pepino,500 (etc)
}
```

Se utiliza el orden de inserción.

La iteración va en el mismo orden en que se insertaron los valores. Map conserva este orden, a diferencia de un Objeto normal.

Además, Map tiene un método forEach incorporado, similar al de Array:

ejemplo7

```
// recorre la función para cada par (clave, valor)
```


```
recipeMap.forEach( (value, key, map) => {
  alert(`${key}: ${value}`); // pepino: 500 etc
});
```

Object.entries: Map desde Objeto

Al crear un Map, podemos pasarle un array (u otro iterable) con pares clave/valor para la inicialización:

ejemplo8

```
// array de [clave, valor]
let map = new Map([
  ['1',  'str1'],
  [1,    'num1'],
  [true, 'bool1']
]);

alert( map.get('1') ); // str1
```

Si tenemos un objeto plano, y queremos crear un Map a partir de él, podemos usar el método incorporado Object.entries(obj) que devuelve un array de pares clave/valor para un objeto en ese preciso formato.

Entonces podemos inicializar un map desde un objeto:

ejemplo9

```
let obj = {
  name: "John",
  age: 30
};

let map = new Map(Object.entries(obj));

alert( map.get('name') ); // John
```

Aquí, Object.entries devuelve el array de pares clave/valor: [["name","John"], ["age", 30]]. Es lo que necesita Map.

Object.fromEntries: Objeto desde Map

Acabamos de ver cómo crear un Map a partir de un objeto simple con Object.entries (obj). Existe el método Object.fromEntries que hace lo contrario: dado un array de pares [clave, valor], crea un objeto a partir de ellos:

ejemplo 10

```
let prices = Object.fromEntries([    ['banana', 1],   ['orange', 2],   ['meat', 4]    ]);
// ahora prices = { banana: 1, orange: 2, meat: 4 }
alert(prices.orange); // 2
```

Podemos usar Object.fromEntries para obtener un objeto desde Map.

Ejemplo: almacenamos los datos en un Map, pero necesitamos pasarlos a un código de terceros que espera un objeto simple.

ejemplo 11

```
let map = new Map();
map.set('banana', 1);
map.set('orange', 2);
map.set('meat', 4);

let obj = Object.fromEntries(map.entries()); // hace un objeto simple (*)

// Hecho!
// obj = { banana: 1, orange: 2, meat: 4 }

alert(obj.orange); // 2
```

Una llamada a map.entries() devuelve un array de pares clave/valor, exactamente en el formato correcto para Object.fromEntries.

es posible acortar la línea omitiendo **entries**

let obj = Object.fromEntries(map);

Es lo mismo, porque Object.fromEntries espera un objeto iterable como argumento. No necesariamente un array. Y la iteración estándar para el Map devuelve los mismos pares clave/valor que map.entries(). Entonces obtenemos un objeto simple con las mismas claves/valores que Map.

Ejercicios

Ejercicio 0: Ejecuta cada uno de los ejemplos pero modificando la salida para que lo muestre en un div.

Ejercicio 1: Crear un Map de Personajes Crea un objeto Map llamado personajes donde las claves sean los nombres de algunos personajes de Los Simpsons y los valores sean las ocupaciones de esos personajes.

Ejercicio 2: Imprimir Ocupaciones Itera sobre el mapa de personajes e imprime en la consola el nombre del personaje seguido de su ocupación.

Ejercicio 3: Encontrar Personajes por Ocupación Escribe una función que tome el mapa de personajes y una ocupación como argumentos, y devuelva un array con los nombres de los personajes que tienen esa ocupación.

Ejercicio 4: Modificar Ocupación Escribe una función que tome el mapa de personajes, el nombre de un personaje y una nueva ocupación, y actualice la ocupación del personaje en el mapa.

Ejercicio 5: Contar Ocupaciones Únicas Escribe una función que tome el mapa de personajes y devuelva el número de ocupaciones únicas presentes en el mapa.

Ejercicio 6: Eliminar Personaje Escribe una función que tome el mapa de personajes y el nombre de un personaje, y elimine ese personaje del mapa.

Ejercicio 7: Combinar dos Maps de Personajes Crea dos mapas de personajes y luego escribe una función que combine ambos mapas en uno solo. En caso de conflictos (mismas claves), suma las ocupaciones.

Ejercicio 8: Elimina los personajes repetidos Crea un mapas de personajes nuevo eliminando los personajes que esten repetidos (mismas claves y ocupaciones.)