EJERCICIO TEÓRICO TEMA 3

*1.Búsqueda de información de los conjuntos en JS.*

1. ¿Qué son?

Los conjuntos en JavaScript almacenarán los elementos en el orden de inserción. están ordenados en JavaScript. Y almacenará tipos de datos u objetos primitivos y son valores únicos no se pueden repetir en el conjunto.

1. Declaración:

var nombres = new Set (["Outmane", "Harry", "John", "Jack", "Bo"]);

console.log(nombres);

1. Propiedades: Size

La propiedad Size devuelve el número de elementos que están presentados en el conjunto.

Console.log(nombres.size);

1. Métodos:
2. Add

Para agregar un nuevo elemento al conjunto utilizamos el método add. Si el nuevo elemento ya está en el conjunto, no lo agregará.

// set esta vacia

var nombres = new Set ();

//Añadimos los nombres con el método add.

nombres.add("outmane");

nombres. add("bo");

console.log(nombres);

1. Delete

El método delete saca un argumento y lo elimina desde el conjunto. Si el argumento no esta en el conjunto no da ningún error.

var nombres = new Set (["Outmane", "Harry", "John", "Jack", "Bo"]);

nombres. delete("outmane");

console.log(nombres);

1. Has

El método has saca un argumento y devuelve true si existe en el conjunto y false si no existe.

var nombres = new Set (["Outmane", "Harry", "John", "Jack", "Bo"]);

console.log(nombres.has(“outmane”));

1. Recorrer conjuntos:

var nombres = new Set (["Outmane", "Harry", "John", "Jack", "Bo"]);

nombres. forEach((element) => {

console.log(element);

});

Or

for (let x of nombres) {

console.log(x);

}

*2. Búsqueda de información de los mapas en JS.*

1. ¿Qué son?

El mapa es un tipo de estructura de datos que almacena pares clave-valor. Es una de las formas más eficientes de buscar y recuperar datos. Puede almacenar todo tipo de datos en mapas JavaScript.

1. Declaración: var nombres = new Map ();
2. Propiedades: Size

La propiedad Size devuelve el número de elementos que están presentados en el mapa que hemos declarado. Console.log(nombres.size);

1. Métodos:
2. set

se utiliza para agregar un nuevo elemento al mapa. Y también se utiliza para cambiar los valores existentes del mapa.

// map está vacía

var nombres = new Map ();

//Añadimos los nombres con el método add.

nombres.set ("outmane",” bouhou”);

nombres. set ("bo",” Jack”);

console.log(nombres);

1. get

El método get saca un valor de calve del mapa dado.

Map {

' outmane ' => ' bouhou ',

' bo ' => ' Jack ',

}

console.log(nombres.get(“outmane”)); //devuelve bouhou

1. Has

El método has saca un calve y devuelve true si existe en el mapa y false si no existe.

Map {

' outmane ' => ' bouhou ',

' bo ' => ' Jack ',

}

console.log(nombres.has(“bh”)); //devuelve false porque no existe

console.log(nombres.has(“outmane”)); //devuelve true porque existe este clave en el mapa.

1. Delete

El método delete saca un valor de calve del mapa y borrarlo.

Map {

' outmane ' => ' bouhou ',

' bo ' => ' Jack ',

}

nombres.delete(“bo”);

Map {

' outmane ' => ' bouhou ',

}

1. Recorrer mapas:

var text = "";  
nombres. forEach (function(value, key) {  
  text += key + ' => ' + value;  
})