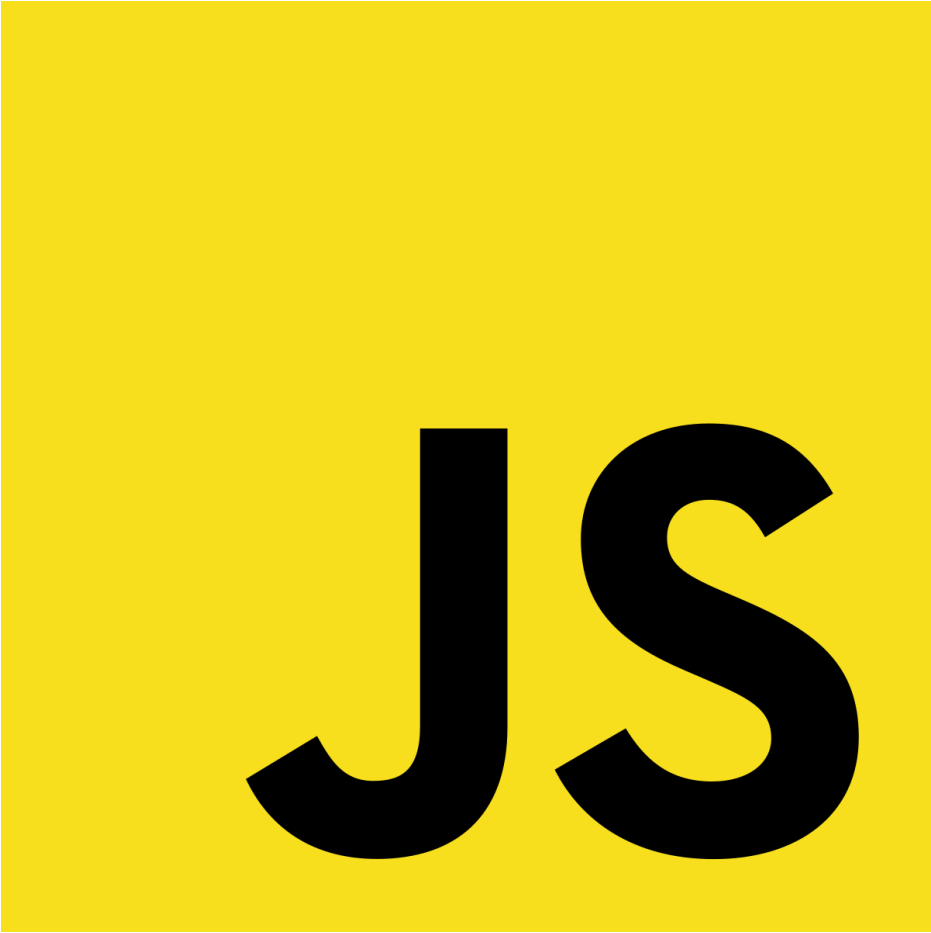


Ejercicio

Teórico

TEMA 3



JS

INDICE

1. Búsqueda de información de los conjuntos en JS.....	3
2. Búsqueda de información de los mapas en JS.	4

1. Búsqueda de información de los conjuntos en JS.

a. ¿Qué son?

Los conjuntos son una colección de elementos únicos que no almacenan valores repetidos. Los conjuntos en JS almacenan los elementos en el orden de inserción. Almacenan tipos de datos u objetos primitivos.

b. Declaración

Para declarar un conjunto se hace de la siguiente manera:

```
const ejemploconjunto = new Set(["Alberto", "Fernandez", "Ramirez"]);  
console.log(ejemploconjunto);
```

c. Propiedades

Los conjuntos tienen diferentes propiedades que nos ayudan a trabajar con ellos para resolver diferentes problemas. Son tan simples de usar que basta con crearlos.

i. size

La propiedad size devuelve el número de elementos presentes en el conjunto.

```
console.log(`Size: ${ejemploconjunto.size}`);
```

d. Métodos

Los conjuntos tienen diferentes métodos que nos ayudan a trabajar con ellos para resolver diferentes problemas. Podemos obtener fácilmente la funcionalidad de los métodos con sus propios nombres.

i. add

El método add se utiliza para agregar un nuevo elemento al conjunto. Si el nuevo elemento ya está presente en el conjunto, no lo agregará.

```
ejemploconjunto.add("Otro");  
ejemploconjunto.add("Fernandez");//El repetido no se introduce  
ejemploconjunto.add("Mas");  
ejemploconjunto.add("En");  
ejemploconjunto.add("Lista");  
console.log(ejemploconjunto);
```

ii. delete

El método delete toma un argumento y lo elimina del conjunto. No arroja ningún error incluso si el argumento dado no está presente en el conjunto.

```
ejemploconjunto.delete("Otro");  
console.log(ejemploconjunto);
```

iii. has

El método has toma un argumento y regresa true si el elemento está presente en el conjunto de lo contrario, regresa false.

```
console.log(ejemploconjunto.has("Fernandez"));  
console.log(ejemploconjunto.has("noestoy"));
```

e. Recorrer conjuntos

Podemos recorrer un conjunto con `forEach` y obtener todos los elementos uno por uno.

```
ejemploconjunto.forEach((element) => {  
    console.log(element);  
});
```

2. Búsqueda de información de los mapas en JS.

a. ¿Qué son?

Los mapas son un diccionario clave-valor donde cualquier tipo puede ser usado como clave. Esta es la principal característica de los mapas y su mayor diferencia con los objetos, donde las claves solo pueden ser cadenas de texto. En los mapas cualquier elemento, incluidos los objetos como matrices o funciones, pueden ser claves para localizar valores.

b. Declaración

Para declarar un mapa se hace de la siguiente manera:

```
var ejemplomapa = new Map();  
console.log(ejemplomapa);
```

c. Propiedades

i. size

Nos permite obtener el numero de valores en el mapa.

```
console.log(ejemplomapa.size);
```

d. Métodos

i. set

Nos permite añadir una nueva pareja clave-valor.

```
ejemplomapa.set( 'uno', 1 );  
ejemplomapa.set( 'dos', 2 );  
ejemplomapa.set( 'tres', 3 );  
console.log(ejemplomapa);
```

ii. get

Nos permite obtener el valor asociado a una clave.

```
console.log(ejemplomapa.get('uno'));
```

iii. has

Nos permite comprobar si hay una determinada clave en el mapa. Devuelve true si lo encuentra o false de lo contrario.

```
console.log(ejemplomapa.has(["dos"]));
```

iv. delete

Nos permite borrar una pareja clave-valor por medio de la clave.

```
ejemplomapa.delete("dos");  
console.log(ejemplomapa);
```

e. Recorrer mapas

Podemos recorrer un mapa con `forEach` y obtener todas las claves-valor uno por uno.

```
ejemplomapa.forEach((values,keys)=>{  
    console.log(values,keys);  
})
```