1.Tanto los objetos javaScript como los Maps permiten almacenar pares clave/valor. Indica la diferencia entre ambos.

- Las claves de un Object son Strings y Symbols, mientras que para un Map pueden ser de cualquier tipo, incluyendo funciones, objetos y cualquier otro tipo primitivo.
- Puedes saber facilmente el tamaño de un Map usando la propiedad size, mientras que le número de propiedades en un Object tiene que ser deternminado manualmente.
- Un Map es un iterable lo que permite iterar directamente sobre el, mientras que si queremos iterar sobre un Object necesitamos obtener primero las claves de alguna forma para después iterar sobre el.
- Un Object tiene prototipo, por lo que hay claves por defecto en tu mapa que pueden colisionar con tus claves si no eres cuidadoso. En el estandard ES5 esto se puede evitar usando mapa = Object.create(null), pero esto raramente se hace.

2.Tanto los arrays javaScript como los Sets permiten almacenar elementos. Indica la diferencia entre ambos.

Una de las mayores diferencias es que los elementos en un Array pueden duplicarse, en un Set estos son únicos. Además, Array se considera como el tipo de estructura de datos de "colección indexada", mientras que Set se considera como "colección con clave".

3.Responde con respecto a Map:

Un conjunto de elementos de tipo:
clave/valor, pueden ser objetos o valores primitivos.

Constructor admite como parámetros:
Iterable, cualquier otro objeto iterable cuyos elementos son pares clave-valor

Métodos para añadir:
set(key, value) .

Métodos para eliminar:
delete(key)

clear()

Métodos para buscar:
get(key)
entries()
keys()
has()
values()

Número de elementos:
size

Se recorren mediante:
forEach(callbackFn[, thisArg])

5.Responde con respecto a Set:
Un conjunto de elementos de tipo:
Permite almacenar valores únicos de cualquier tipo, incluso valores primitivos u objetos de referencia.

Constructor admite como parámetros:
iterable, Si un objeto iterable es pasado, todos sus elementos serán añadidos al nuevo Set. Si no se especifica este parámetro, o si su valor es null, el nuevo Set estará vacío.

Métodos para añadir:
add(value)

Métodos para eliminar:
delete(value)
clear()

Métodos para buscar:
entries()
keys()
has(value)
values()

Número de elementos:
size

Se recorren mediante:
forEach(callbackFn[, thisArg])

8.Responde con respecto a WeakSet:
Un conjunto de elementos de tipo:
Object

Constructor admite como parámetros:
iterable, si un objeto iterable es pasado, todos sus elementos se agregaran al nuevo WeakSet. null es tratado como undefined.

Métodos para añadir:
add(value)

Métodos para eliminar:
delete(value)

Métodos para buscar:
has(value)

Número de elementos:

Si dos elementos son iguales:

Se recorren mediante:

11.Indica la desventaja de realizar esta modificación sobre el objeto
wS = new WeakSet();
wS.add({numero: 1});

Las referencias a objetos en la colección se mantienen débilmente.. Si ya no hay otra referencia a un objeto almacenado en el WeakSet, ellos pueden ser removidos por el recolector de basura. Esto también significa que no hay ninguna lista de objetos almacenados en la colección.

12.Responde verdadero falso:
Tanto WeakSet como Set almacenan valores únicos.
VERDADERO

WeakSet sólo almacena Objetos.
VERDADERO

La estructura Set autoelimina los objetos que no tienen referencia.
FALSO