# **Práctica** Etiqueta script. Añadiendo comportamiento al html.

## Objetivos

Contestar las siguientes cuestiones sobre collections en javascript:

# Mi equipo:

La práctica ha sido realizada bajo un equipo con sistema operativo MacOs, todo el software usado a sido actualizado a la última versión disponible el día de la práctica.

**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 fácilmente el tamaño de un Map usando la propiedad size, mientras que el número de propiedades en un Object tiene que ser determinado manualmente.

Un Map es un iterable lo que permite iterar directamente sobre él, 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.**

Los objetos Set son colecciones de valores. Se puede iterar sus elementos en el orden de su inserción. Un valor en un Set sólo puede estar una vez; éste es único en la colección Set, esto en un array no pasaría.

**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, Iterable es un array o cualquier otro objeto iterable cuyos elementos son pares clave-valor (arrays de 2 elementos). Cada par clave-valor será agregado al nuevo Map.
**Métodos para añadir:**
set(key, value), Establece un valor para la llave en el objeto Map. Devuelve el objeto Map.
**Métodos para eliminar:**
delete(key), Elimina cualquier valor asociado a la llave y devuelve el valor que Map.prototype.has(key) tenía previamente. Después Map.prototype.has(key) devolverá false.
**Métodos para buscar:**
get(key), Devuelve el valor asociado a la llave, o undefined si no tiene ninguno.
**Número de elementos:**
.size, devuelve el número de conjuntos de llave/valor en el objeto Map.
**Si dos elementos son iguales:**

**Se recorren mediante:**
**.forEach(callbackFn[, thisArg])**, llama a la callbackFn una vez por cada conjunto llave/valor presentes en cada objeto Map, en orden de inserción. Si se le proporciona un parámetro thisArg a forEach, se usará como valor "this" para cada callback.

**4.Entrega un código (debidamente comentado) donde demuestres los apartados del ejercicio anterior sobre la collection Map. Para ello utiliza como elementos los nombres y apellidos de cinco compañeros de clase.**

**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), agrega un nuevo elemento con el valor dado al objeto Set. Devuelve el objeto Set.
**Métodos para eliminar:**
delete(value)**,** elimina cualquier valor asociado a la llave y devuelve el valor que Set.prototype.has(key) tenía previamente. Después Set.prototype.has(key) devolverá false.
**Métodos para buscar:**
has(value), devuelve un booleano que indica si un valor se ha asociado a la llave en el objeto Map o no se ha asociado.
**Número de elementos:**
.size, devuelve el número de valores del objeto Set.
**Si dos elementos son iguales:**

**Se recorren mediante:**
forEach(callbackFn[, thisArg])**,** llama a callbackFn una vez por cada valor presente en el objeto Set, en orden de inserción. Si se proporciona un parámetro thisArg para forEach, se utilizará como este valor para cada devolución de llamada.

**6.Entrega un código (debidamente comentado) donde demuestres los apartados del ejercicio anterior sobre la collection Set. Para ello utiliza como elementos los nombres y apellidos de cinco compañeros de clase.**

**7.Analiza el siguiente enlace Javascript Set vs. Array performance y responde a las siguientes preguntas con respecto al rendimiento:**
**Al añadir elementos. Métodos utilizados.**
**Al modificar elementos. Métodos utilizados.**
**Al eliminar elementos. Métodos utilizados.**

**8.Responde con respecto a WeakSet:**
**Un conjunto de elementos de tipo:**
Te permite almacenar y mantener objetos débilmente en una colección.
**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), agrega un nuevo objeto con el valor dado al objeto WeakSet.
**Métodos para eliminar:**
delete(value), elimina el elemento asociado al valor. WeakSet.prototype.has (valor) devolverá falso después.
**Métodos para buscar:**
has(value), devuelve un valor booleano que indica si un elemento está presente con el valor dado en el objeto WeakSet o no.
**Número de elementos:**

**Si dos elementos son iguales:**

**Se recorren mediante:**

**9.Entrega un código (debidamente comentado) donde demuestres los apartados del ejercicio anterior sobre la collection WeakSet . Para ello utiliza como elementos los nombres y apellidos de cinco compañeros de clase**

**10.Crea mediante iterables estos dos objetos collections. Indica el número de elementos de cada uno. Justifica el comportamiento:**
**b = new WeakSet([{}, {}]);**
**o = {};**
**a = new WeakSet([o, o]);**

**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**