JavaScript Maps and Sets Review

Sets in JavaScript

- A Set is a built-in option for managing data collection.
- Sets ensure that each value in it appears only once, making it useful for eliminating duplicates from an array or handling collections of distinct values.
- You can create a Set using the Set() constructor:

Example Code

```
const set = new Set([1, 2, 3, 4, 5]);

console.log(set); // Set { 1, 2, 3, 4, 5 }
```

- Sets can be manipulated using these methods:
    - add(): Adds a new element to the Set.
    - delete(): Removes an element from the Set.
    - has(): Checks if an element exists in the Set.
    - clear(): Removes all elements from the Set.
    - keys() and values(): Both returns a SetIterator that contains the values of the Set. They are the same because keys() is an alias for values().
    - forEach(): for iterating over the values of the Set.

Weaksets in JavaScript

- WeakSet is a collection of objects that allows you to store weakly held objects.

Sets vs WeakSets

- Unlike Sets, a WeakSet does not support primitives like numbers or strings.
- A WeakSet only stores objects, and the references to those objects are "weak," meaning that if the object is not being used anywhere else in your code, it is removed automatically to free up memory.

Maps in JavaScript

- A Map is a built-in object that holds key-value pairs just like an object.

- Maps differ from the standard JavaScript objects with their ability to allow keys of any type, including objects, and functions.
- A Map provides better performance over the standard object when it comes to frequent addition and removals of key-value pairs.
- You can create a Map using the Map() constructor:

Example Code

```
const map = new Map([
 ['flower', 'rose'],
 ['fruit', 'apple'],
 ['vegetable', 'carrot']
]);
console.log(map); // Map(3) { 'flower' => 'rose', 'fruit' => 'apple', 'vegetable' => 'carrot' }
```

- Maps can be manipulated using these methods:
    - set(): Adds a new key-value pair to the Map.
    - get(): Retrieves the value of a key from the Map.
    - delete(): Removes a key-value pair from the Map.
    - has(): Checks if a key exists in the Map.
    - clear(): Removes all key-value pairs from the Map.

Note that both Maps and Sets have the size property that returns the number of unique elements in them.

WeakMaps in JavaScript

- A WeakMap is a collection of key-value pairs just like Map, but with weak references to the keys. The keys must be an object and the values can be anything you like.

Maps vs WeakMaps

- WeakMaps are similar to WeakSets in that they only store objects and the references to those objects are "weak".