Cheat Sheet – Maps, Sets & Subclassing

Map

A Map is a key-value collection introduced in ES6. It kind of fills the gap between arrays (no key-value pairs) and objects (key-value pairs but much more complex than a simple collection).
You can create a Map like this:

```
let cardAce = {
   name: 'Ace of Spades'
};
let cardKing = {
   name: 'King of Clubs'
};
let deck = new Map();
deck.set('as', cardAce);
deck.set('kc', cardKing);
```

You can get values from a map by using the get(key) method. Of course you also have methods to clear a map or delete single items.
More information can be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map

WeakMap

A WeakMap basically also is a Map but it misses some features. It is not enumerable (you can't loop through it) and it has no size property.
Why would you use a WeakMap if you can use a "real" Map? As the name implies, WeakMaps hold weak references to the stored values. That means, if some values aren't used anymore, they can get garbage-collected and will be released from the map. That's also the reason why a WeakMap has no size property.
More information can be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakMap

Set

A Set is a collection which only holds values. Sounds like an Array? Almost, but a Set will only old unique values. That means, no value can appear more often than once in a Set.
You can loop through a set to retrieve the values (or use an Iterator). You can also clear() a set or delete individual values by using delete(). Remember, each value is unique, therefore you don't need a key or index to delete a value!
You can create a Set like this:

```
let cardAce = {
   name: 'Ace of Spades'
};
let cardKing = {
   name: 'King of Clubs'
};
let deck = new Set();
deck.add(cardAce);
deck.add(cardKing);
deck.add(cardKing); // Won't be added, only added once!
```

More information can be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Set

WeakSet

Like a WeakMap, a WeakSet is comparable to a "normal" Set but it only holds weak references.
You may find more information here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/WeakSet

Subclassing

Subclassing basically means, that you can now extend certain JavaScript base-objects. This allows you to add your own functionalities to those objects.
For example, you may extend the Array object like this

```
class ConvertableArray extends Array {
   convert() {
      let returnArray = [];
      this.forEach(value => returnArray.push('' + value));
      return returnArray;
   }
}
```