Map – a JavaScript data structure for mapping values to values

Lets you use arbitrary values as keys and is highly welcome

WeakMaps work mostly like Maps, with the following differences:

WeakMap keys are objects (values can be arbitrary values)

WeakMap keys are weakly held

You can’t get an overview of the contents of a WeakMap

You can’t clear a WeakMap

What are the main benefits?

In a direct comparison, Sets have several advantages over arrays, especially when it comes to a faster run-time:

Search for an Item: Using indexOf() or includes() to check whether an item exists in an array is slow.

Deleting an Item: In a Set, you can delete an item by its value. In an array, the equivalent is using splice() based on an element’s index. As in the previous point, depending on indices is slow.

Insert an Item: It is faster to add an item to a Set than to add an item to an array using push() , unshift() or an equivalent method.

Storing NaN: You cannot use indexOf() or includes() to find the value NaN , while a Set is able to store this value.

Removing Duplicates: Set objects only store unique values. If you want to avoid storing duplicates, this is a significant advantage over arrays, where additional code would be required to deal with duplicates.

ECMAScript 5 doesn’t have a Set data structure, either. There are two possible work-arounds:

• Use the keys of an object to store the elements of a set of strings. • Store (arbitrary) set elements in an Array: Check whether it contains an element via indexOf(), remove elements via filter(), etc. This is not a very fast solution, but it’s easy to implement. One issue to be aware of is that indexOf() can’t find the value NaN.

19.6.1 Why do Maps and Sets have the property size and not length?

Arrays have the property length to count the number of entries. Maps and Sets have a different property, size. The reason for this difference is that length is for sequences, data structures that are indexable – like Arrays. size is for collections that are primarily unordered – like Maps and Sets.

19.6.4 When should I use a Map, when an object?

If you map anything other than strings to any kind of data, you have no choice: you must use a Map. If, however, you are mapping strings to arbitrary data, you must decide whether or not to use an object. A rough general guideline is:

• Is there a fixed set of keys (known at development time)? Then use an object and access the values via fixed keys: obj.key • Can the set of keys change at runtime? Then use a Map and access the values via keys stored in variables: map.get(theKey)