Weak Sets

introduction to weak sets, their usage and comparison with regular sets

Sets and maps hold a reference of their values. This means that the garbage collector won't be able to collect the values in sets, and key-value pairs in maps to free some memory.

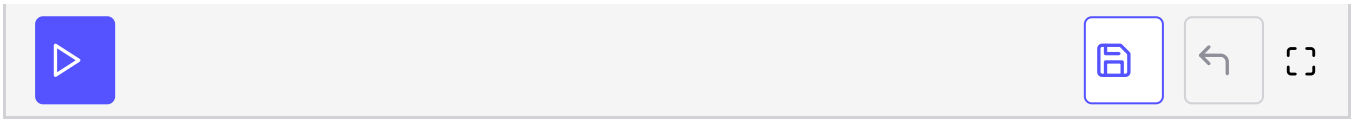
This is where weak sets and maps come into play. They only hold *weak references* to their values, allowing garbage collection of the values while they are members of a set or map.

Weak sets are similar to regular sets. The main difference is that their elements may disappear once they are garbage collected. Consider the following code:

```
let firstElement = { order: 1 }, secondElement = { order: 2 };
let ws = new WeakSet( [ firstElement, secondElement ] );

console.log('has firstElement: '+ws.has( firstElement ));
//> true

delete firstElement;
// firstElement is removed from the weak set
```



Other characteristics of weak sets include:

- You don't have access to the size of the set.
- Weak sets may only store objects.
- Weak sets are not iterable.

The use cases for weak sets are limited. We can use them whenever we deal with objects that can be garbage collected from other sources, and we only want to check the existence of objects in the weak set.

For instance, whenever we traverse a graph, and we would like to detect cycles, we can store the nodes in weak sets during traversal.

Now, let's talk about weak maps.