Data Structures 101

introduction to sets and maps in Javascript

This lesson introduces some data structures that we often use during coding: sets and maps. Using these data structures prevents us from reinventing the wheel.

A **set** is an unordered collection of distinct elements. An element is said to be a member of the set, if the set contains the element. Adding an element to the set that's already a member does not change that set.

While a mathematical set is unordered, the ES6 `Set` data structure is an ordered list of distinct elements. Don't confuse yourself with the mathematical definition. We can enumerate the elements of an ES6 `Set` in a given order.

There are some operations defined on sets. These are

- union: `e` becomes a member of `union( A, B )`, if `e` is a member of `A`, **or** `e` is a member of `B`.
- intersection: `e` becomes a member of `intersection( A, B )`, if `e` is a member of `A`, **and** `e` is a member of `B`.
- difference: `e` becomes a member of `difference( A, B )`, if `e` is a member of `A`, and `e` is **not** a member of `B`.

A **map** is a collection of key-value pairs. If you provide a key, you can retrieve the corresponding value. This sounds very much like a simple object. However, there are significant differences between objects and maps. Objects can only work as maps under special circumstances.

In the next few lessons, we'll cover both these data structures in more detail.