Learning how references work in JavaScript

Naveen Karippai

Nov 13, 2016

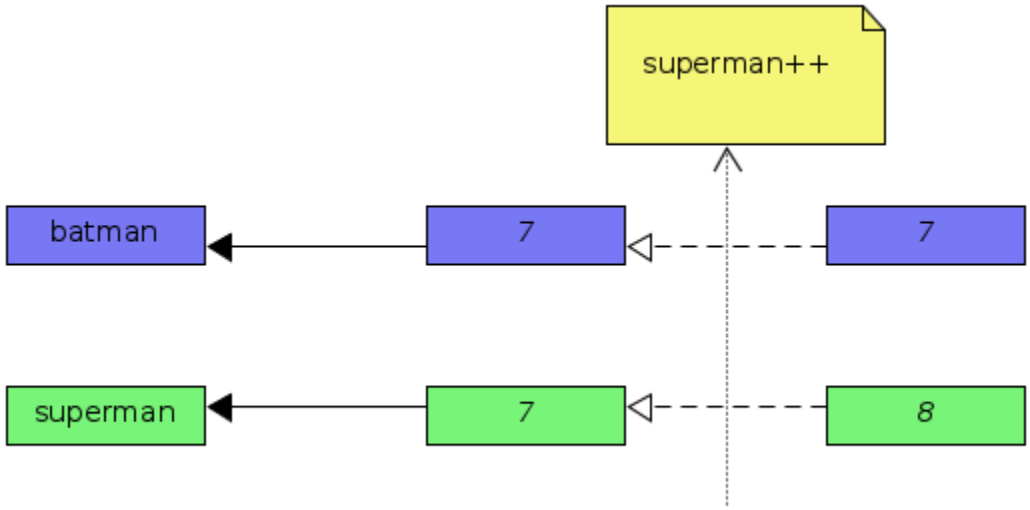
TL;DR**:* There are NO pointers in JavaScript and references work differently from what we would normally see in most other popular programming languages. In JavaScript, it's just NOT possible to have a reference from one variable to another variable. And, only compound values (Object, Array) can be assigned by ***reference.

Bottom line:

1. The *typeof* value assigned to a variable decides whether the value is stored with *assign-by-value* or *assign-by-reference*
2. On variable assignment, the scalar primitive values (Number, String, Boolean, undefined, null, Symbol) are *assigned-by-value* and compound values (Object, Array) are *assigned-by-reference*
3. The references in JavaScript only point at contained values and NOT at other variables or references
4. In JavaScript, scalar primitive values (Number, String, Boolean, undefined, null, Symbol) are *immutable* and compound values (Object, Array) are *mutable*

A quick example on assign-by-value:

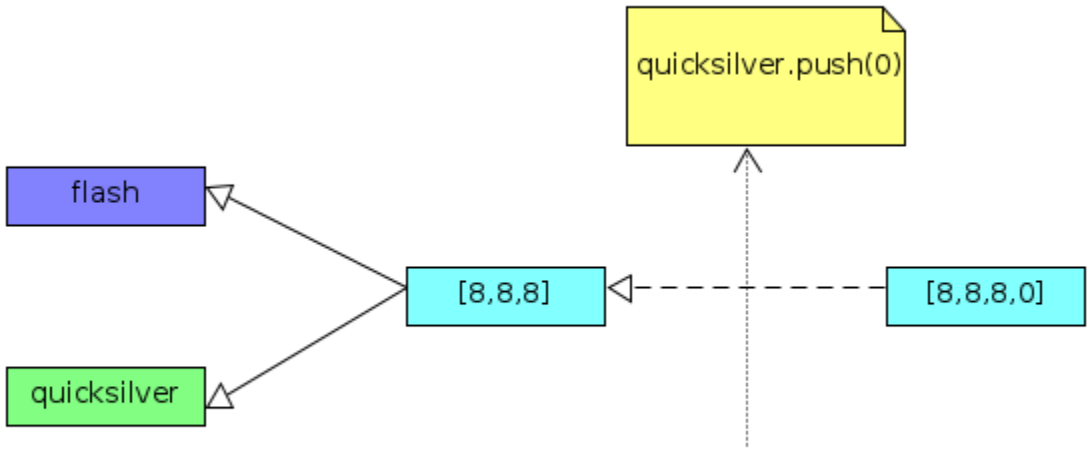
In the code snippet below, we are assigning a scalar primitive value (number) to a variable and thus *assign-by-value* applies here. Firstly, the variable `batman` is initialized and when the variable `superman` is assigned with the value stored in the variable `batman`, it creates a new copy of the value and stores it. When the variable `superman` is modified, variable `batman` is left unaffected as they point to distinct values.



Tool: UMLet (GNU Linux client)

A quick example on assign-by-reference:

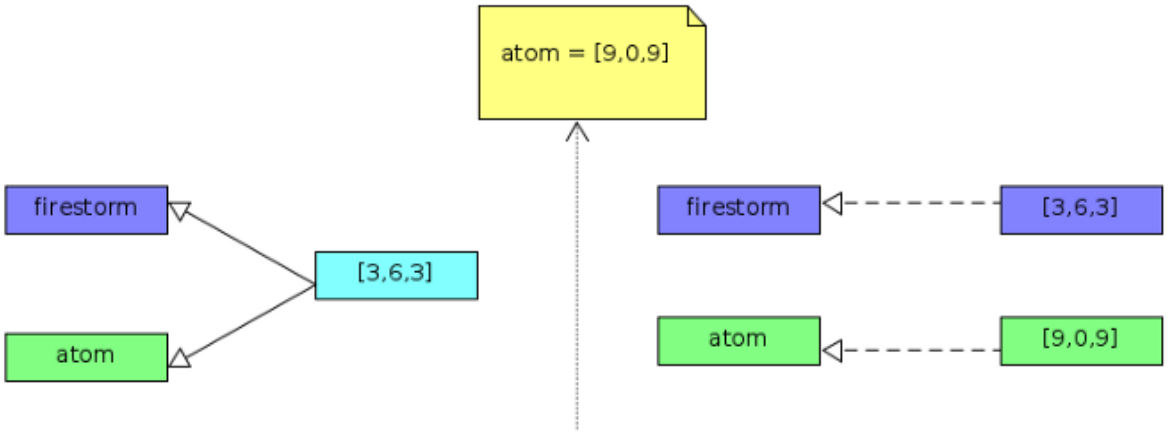
In the code snippet below, we are assigning a compound value (array) in a variable and thus *assign-by-reference* applies here. The variables `flash` and `quicksilver` are references to the same value (aka shared value). The references will point to the updated value when the shared value is modified.



Tool: UMLet (GNU Linux client)

How to create a new reference?

When the compound value in a variable is reassigned, a new reference is created. In JavaScript, unlike in most other popular programming languages, the references are pointers to value stored in variables and NOT pointers to other variables, or references.



Tool: UMLet (GNU Linux client)

How references work when values are passed as function parameters?

In the code snippet below, the variable `magneto` is a compound value (Array object), thus it is assigned in variable (function argument) `x` with assign-by-reference. The `Array.prototype.push` method invoked inside IIFE mutates the value in variable `magneto` through JavaScript reference. But, the reassignment of variable `x` creates a new reference, and further modifications on it do NOT affect the reference to the variable `magneto`.

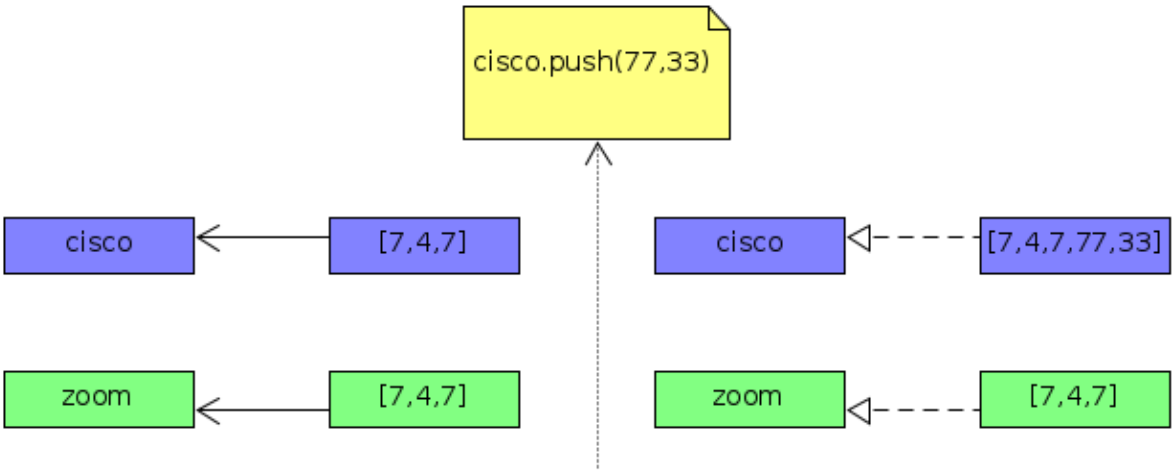
How to change the original value in a compound variable passed as a function argument through JavaScript reference?

The solution here would be to modify the existing compound value where the reference is pointing to. In the code snippet below, the variable `wolverine` is a compound value (Array object) and on IIFE invocation, variable (function argument) `x` is assigned by reference. The `Array.prototype.length` property can be used to create an empty array by setting its value to `0`. Thus, the variable `wolverine` is changed to the new value set in a variable `x` through JavaScript reference.

How to store a compound value through assign-by-value?

The solution here would be to make a manual copy of the compound value and then assign the copied value to a variable. Therefore, the reference of the assigned value does NOT point back to the original

value. The recommended approach to creating a (shallow) copy of the compound value (Array object) is to invoke `Array.prototype.slice` method on it with no arguments passed.



Tool: UMLet (GNU Linux client)

How to store a scalar primitive value through assign-by-reference?

The solution here would be to wrap the scalar primitive value in a compound value (Object, Array) as its property value. Thus, it can be assigned-by-reference. In the code snippet below, the scalar primitive value in the variable `speed` is set as a property on the object `flash`. Therefore, it is assigned-by-reference on IIFE invocation to the variable (function argument) `x`.

An aside:

In JavaScript, the scalar primitive values are *immutable* while compound values are *mutable*.

Summary: A good understanding of references in JavaScript can help developers to avoid many common mistakes and write better code.

Happy coding!!