Deep Copy

An introduction to deep copy

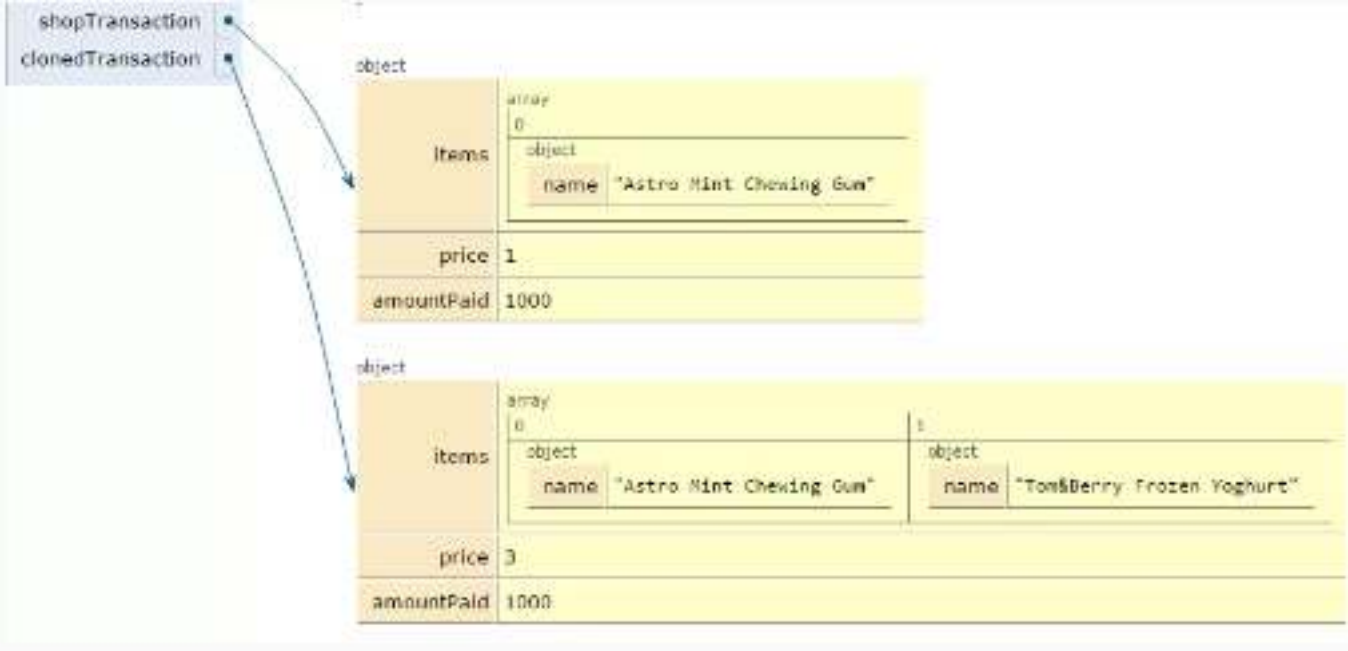
When deeply cloning an object, all references are dereferenced. Only the structure of the object, key names and the atomic values are kept. A deep copy requires traversal of the whole object and building the cloned object from scratch.

```
var shopTransaction = {
    items: [ { name: 'Astro Mint Chewing Gum' } ],
    price: 1,
    amountPaid: 1000
}

var clonedTransaction = deepCopy( shopTransaction );

clonedTransaction.price = 3;
clonedTransaction.items.push( { name: 'Tom&Berry Frozen Yoghurt' } );

console.log( 'clonedTransaction.price = ', clonedTransaction.price );
console.log( 'shopTransaction.price = ', shopTransaction.price );

console.log( 'clonedTransaction.items.length = ', clonedTransaction.items.length );
console.log( 'shopTransaction.items.length = ', shopTransaction.items.length );
```

The deep copy made it possible to modify the `items` member of both transaction objects individually.

```
clonedTransaction.price =  3
shopTransaction.price =  1
clonedTransaction.items.length =  2
shopTransaction.items.length =  1
```

The image below shows that `shopTransaction` and `clonedTransaction` are fully distinct: no values are reachable from both objects.



Another necessary condition for a sound deep clone function is cycle detection. Cloning functions should terminate even if an object references itself. It is possible to clone these objects in case a lookup table of the references are constructed.

Many cloning methods based on an intermediate representation fail though as the intermediate representation has to be finite.