```
function Observer(data) {
	this.data = data;
	this.walk(data);
}
Observer.prototype = {
	walk: function(data) {
		var self = this;
		Object.keys(data).forEach(function(key) {
			self.defineReactive(data, key, data[key]);
		});
	},
	defineReactive: function(data, key, val) {
		var dep = new Dep();
		var childObj = observe(val);
		Object.defineProperty(data, key, {
			enumerable: true,
			configurable: true,
			get: function() {
				if (Dep.target) {
					dep.addSub(Dep.target);
				}
				return val;
			},
			set: function(newVal) {
				if (newVal === val) {
					return;
				}
				val = newVal;
				dep.notify();
			}
		});
	}
};

function observe(value, vm) {
	if (!value || typeof value !== 'object') {
		return;
	}
	return new Observer(value);
};

function Dep () {
	this.subs = [];
}
Dep.prototype = {
	addSub: function(sub) {
		this.subs.push(sub);
	},
	notify: function() {
		this.subs.forEach(function(sub) {
			sub.update();
		});
	}
};
Dep.target = null;
```