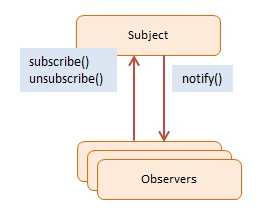
Observer Pattern

Observer pattern offers a subscription model where objects subscribe to events and get notified when event occurs.

Another name for observer pattern is pub/sub



We basically maintain a list of functions to be called if something happens. We can add/remove functions from list depending on if we have to subscribe or unsubscribe.

function Click() {

    this.handlers = [];  // observers

}

Click.prototype = {

    subscribe: function (fn) {

        this.handlers.push(fn);

    },

    unsubscribe: function (fn) {

        this.handlers = this.handlers.filter(

            function (item) {

                if (item !== fn) {

                    return item;

                }

            }

        );

    },

    fire: function () {

        this.handlers.forEach(function (item) {

            item.call();

        });

    }

}

function run() {

    var clickHandler = function (item) {

        console.log("fired: " + item);

    };

    var click = new Click();

    click.subscribe(clickHandler);

    click.fire('event #1');

    click.unsubscribe(clickHandler);

    click.fire('event #2');

    click.subscribe(clickHandler);

    click.fire('event #3');

}

Output of above will be following. ‘event 2’ won’t be fired because ‘clickHandler’

fired: event #1  
fired: event #3