Observer Pattern

Behavioral Design Patterns

Is the Observer Pattern the same as Publish–subscribe pattern?

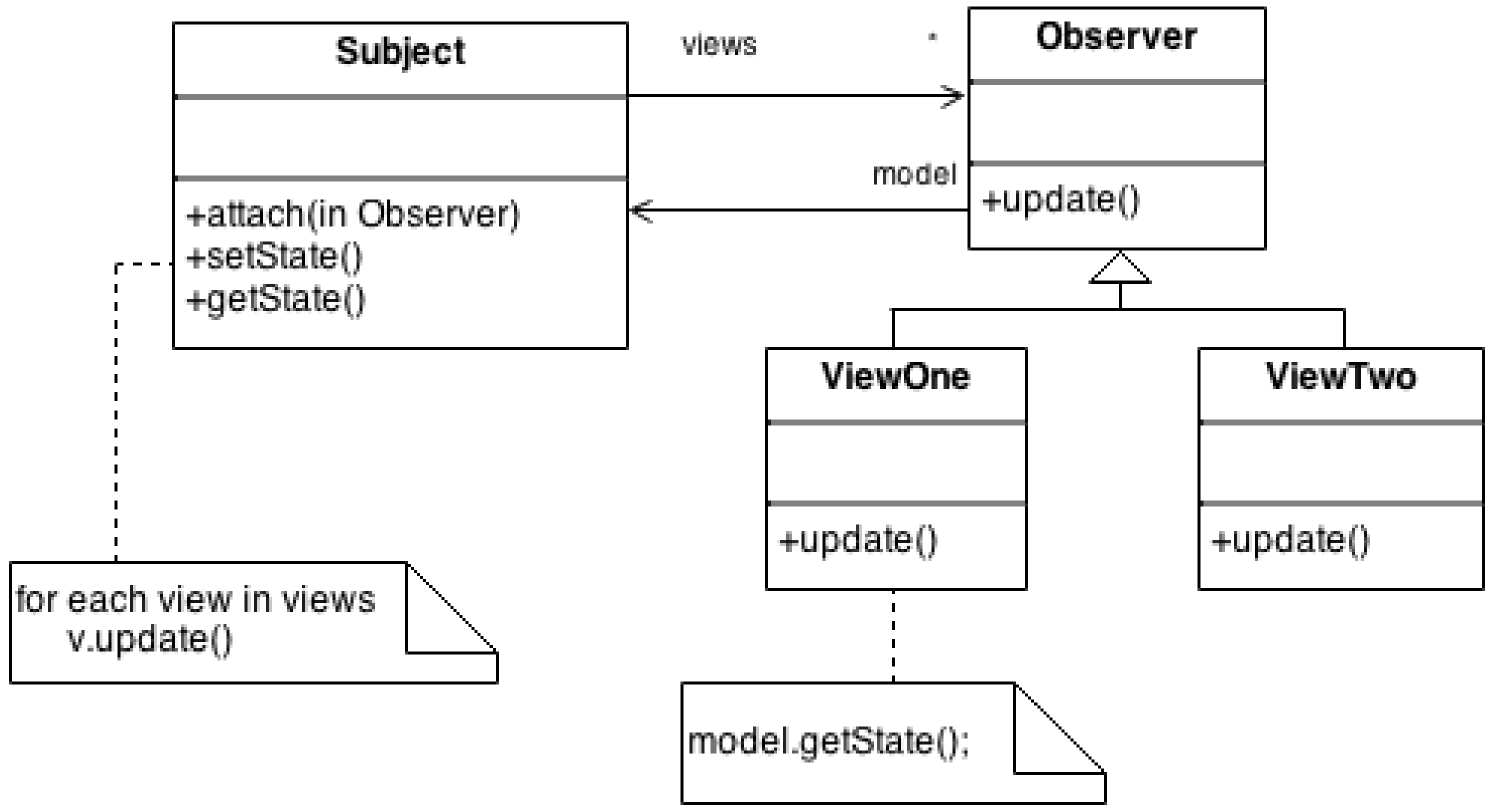
Definition

- The observer pattern is a software design pattern in which an object, called the **subject**, maintains a list of its dependents, called **observers**, and notifies them automatically of any state changes, usually by calling one of their methods. (*)

- Defines a dependency between objects so that whenever an object changes its state, all its dependents are notified.

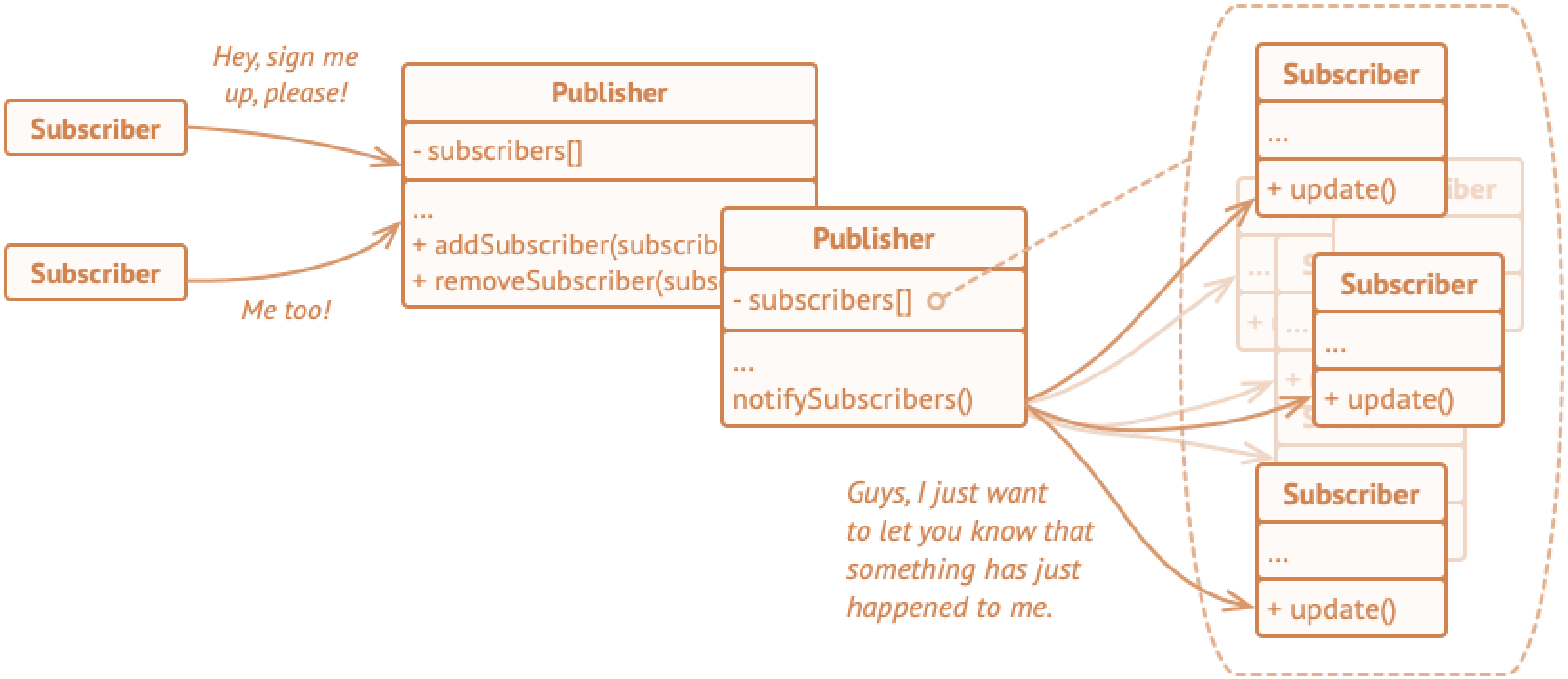
* https://en.wikipedia.org/wiki/Observer_pattern

UML Diagram



https://sourcemaking.com/design_patterns/observer

Simple Diagram



https://refactoring.guru/design-patterns/observer

Demo

When to use?

- Use the Observer pattern when changes to the state of one object may require changing other objects, and the actual set of objects is unknown beforehand or changes dynamically.
- Use the pattern when some objects in your app must observe others, but only for a limited time or in specific cases.

Advantages:

- Subject only knows that observer implement Observer interface.
- There is no need to modify Subject to add or remove observers.
- We can reuse subject and observer classes independently of each other.

Disadvantages:

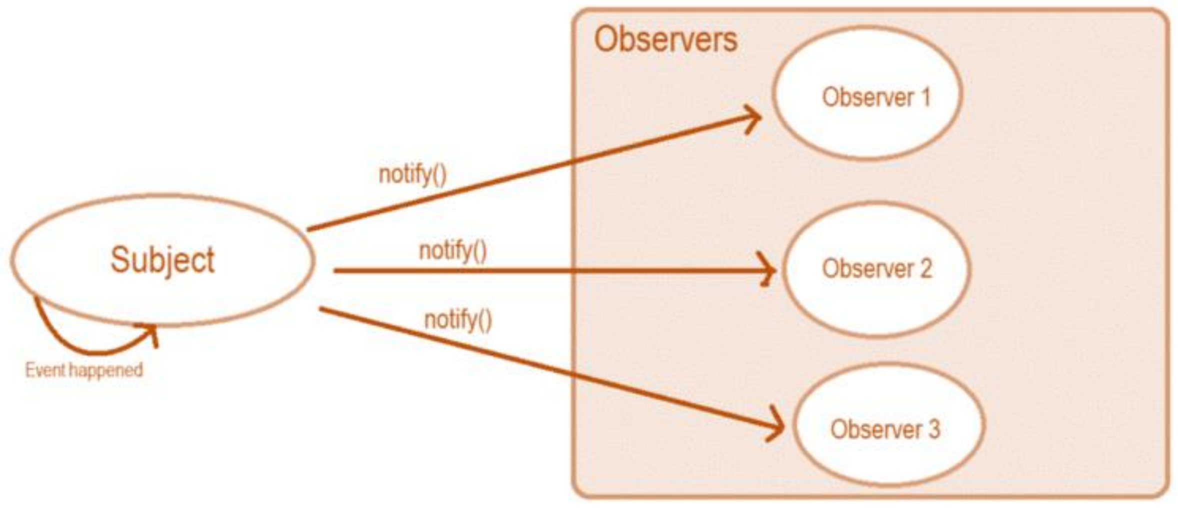
- Memory leaks (because of explicit register and unregistering of observers)

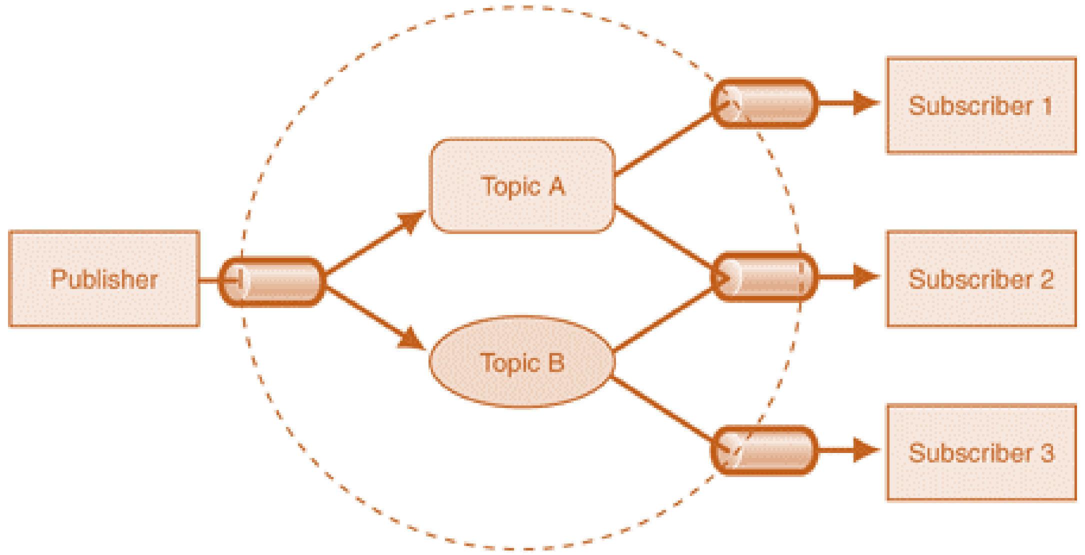
Example (Node.js)

```
const http = require('http');



const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Your own server here');
});

server.on('error', err => {
    console.log("Error:: ", err)
})

server.listen(3000, '127.0.0.1', () => {
  console.log('Server up and running');
});
```

```
class Observable {

    constructor() {
     this.observers = {}
    }

    on(input, observer) {
     if(!this.observers[input]) this.observers[input] = []
     this.observers[input].push(observer)
    }

    triggerInput(input, params) {
     this.observers[input].forEach( o => {
         o.apply(null, params)
     })
    }
}
```

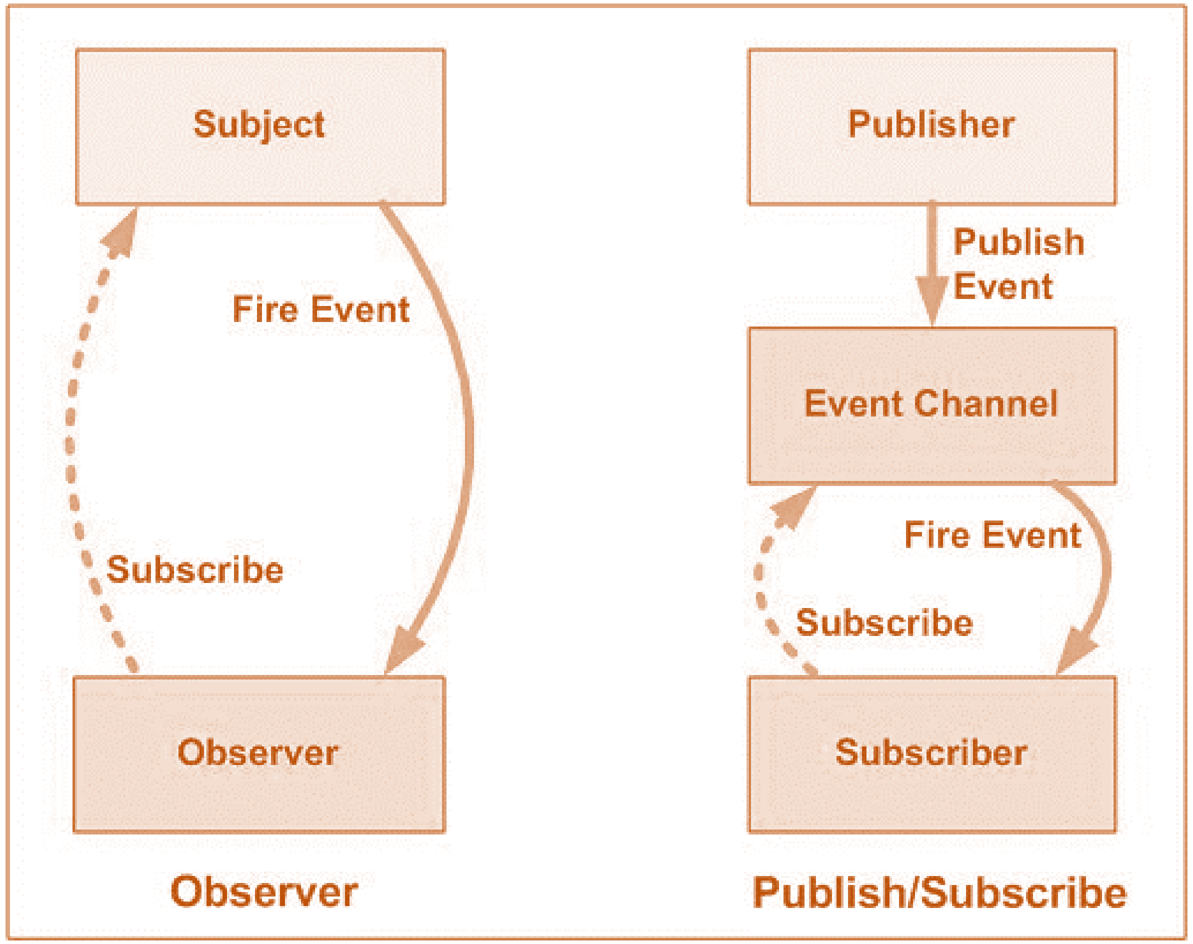
Is the Observer Pattern the same as Publish–subscribe pattern?





[* https://betterprogramming.pub/observer-vs-pub-sub-pattern-50d3b27f838c#:~:text=In%20the%20observer%20pattern%2C%20the,message%20queues%20or%20a%20broker](https://betterprogramming.pub/observer-vs-pub-sub-pattern-50d3b27f838c#:~:text=In%20the%20observer%20pattern%2C%20the,message%20queues%20or%20a%20broker).

Is the Observer Pattern the same as Publish–subscribe pattern?



[* https://betterprogramming.pub/observer-vs-pub-sub-pattern-50d3b27f838c#:~:text=In%20the%20observer%20pattern%2C%20the,message%20queues%20or%20a%20broker](https://betterprogramming.pub/observer-vs-pub-sub-pattern-50d3b27f838c#:~:text=In%20the%20observer%20pattern%2C%20the,message%20queues%20or%20a%20broker).

Q&A

Thank you!

https://github.com/furoTmark/ObserverDesignPattern