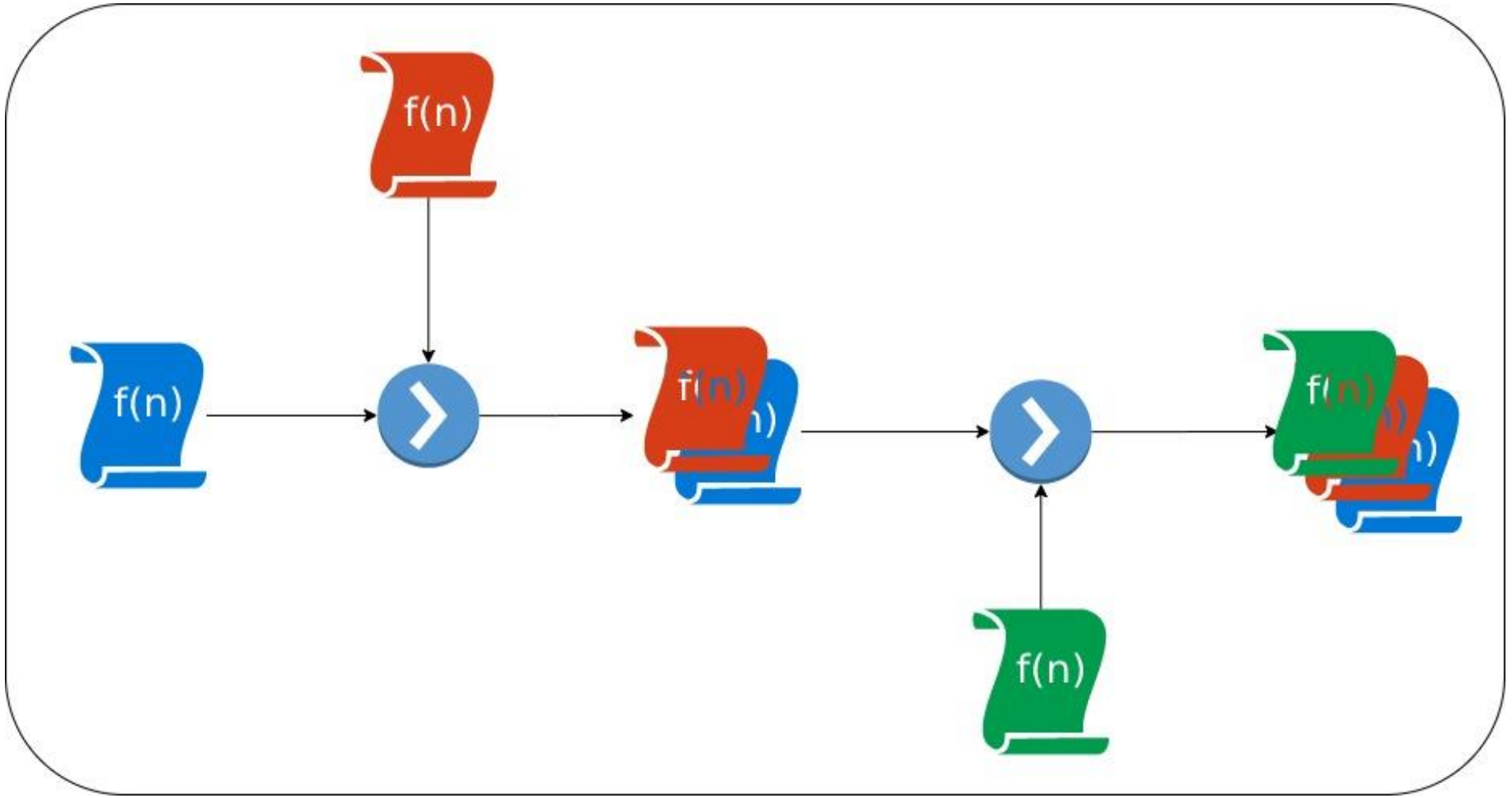
S2 Z4

Design Patterns (2)

Design Patterns (2)

- Wrapper
- Proxy
- Compose
- Observer
- Mediator
- Curry


f(n)
f(n)
f(n)
f(n)
f(n)

Wrapper ½

```
function Add(x, y) {
    for (let i = 0; i <= 100000; i++) {
        for (let j = 0; j <= 100000; j++) {}
    }
    return x + y;
}

function ShowAggumentsWrapper(func) {
    return function(x, y) {
        console.log("Args: " + x + ", " + y);
        return func(x, y);
    };
}

function ShowExecutionTimeWrapper(func) {
    return function(x, y) {
        let timeStaptStart = Date.now();
        let result = func(x, y);
        console.log("Execution time: " + (Date.now() - timeStaptStart));
        return result;
    };
}
```

Wrapper 2/2

```
let addWrapperInShowArguments = ShowAggumentsWrapper(Add);
console.log(addWrapperInShowArguments(3, 7));
```

```
Args: 3, 7
10
```

```
let addWrapperInShowExecutionTime = ShowExecutionTimeWrapper(Add);
console.log(addWrapperInShowExecutionTime(5, 7));
```

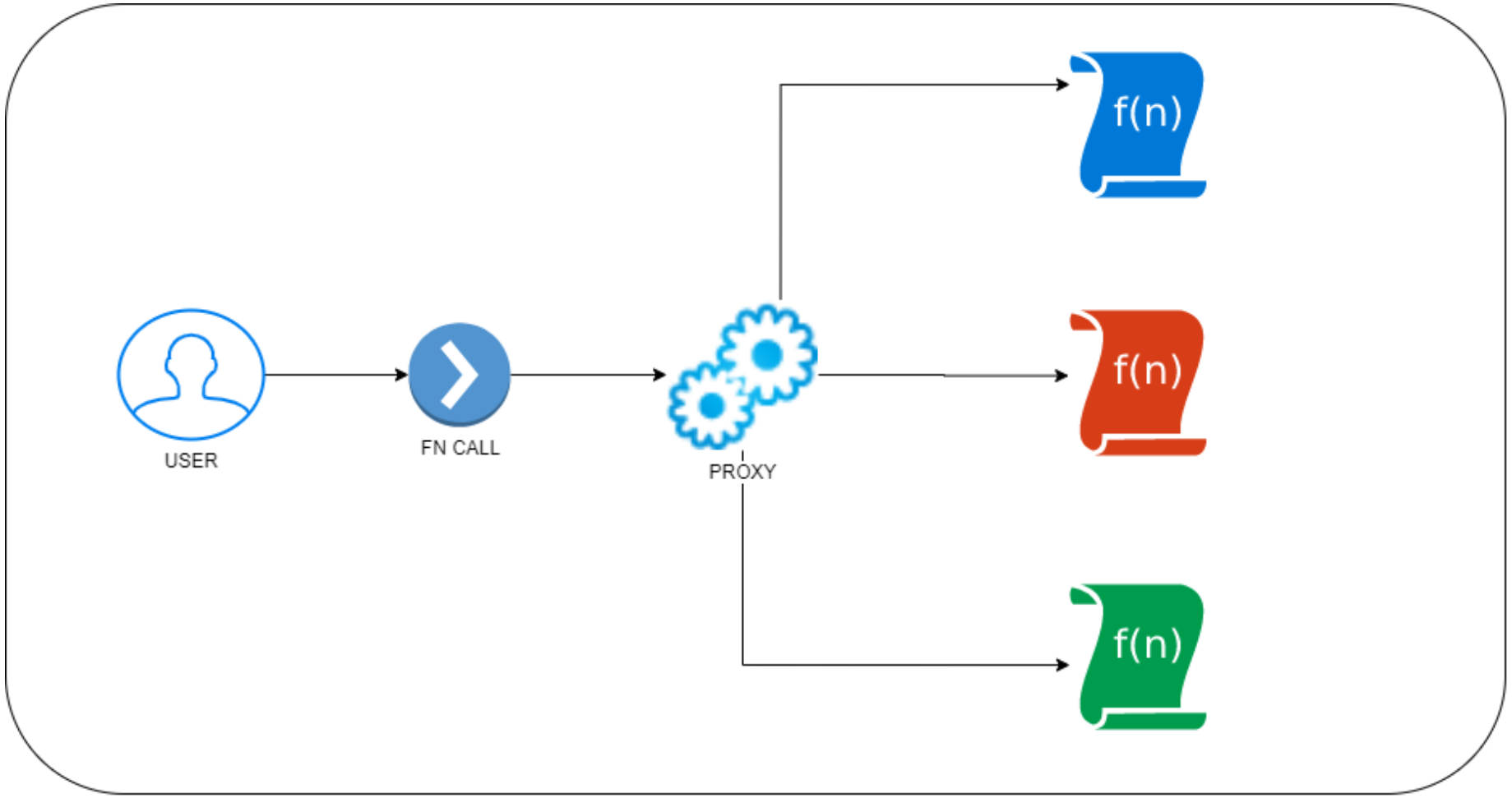
```
Execution time: 7591
12
```

```
let addWrappedInAll = ShowExecutionTimeWrapper(ShowAggumentsWrapper(Add));
console.log(addWrappedInAll(3, 5));
```

```
Args: 3, 5
Execution time: 8033
8
```

```
Add = ShowExecutionTimeWrapper(ShowAggumentsWrapper(Add));
console.log(Add(1, 2));
```

```
Args: 1, 2
Execution time: 7577
3
```

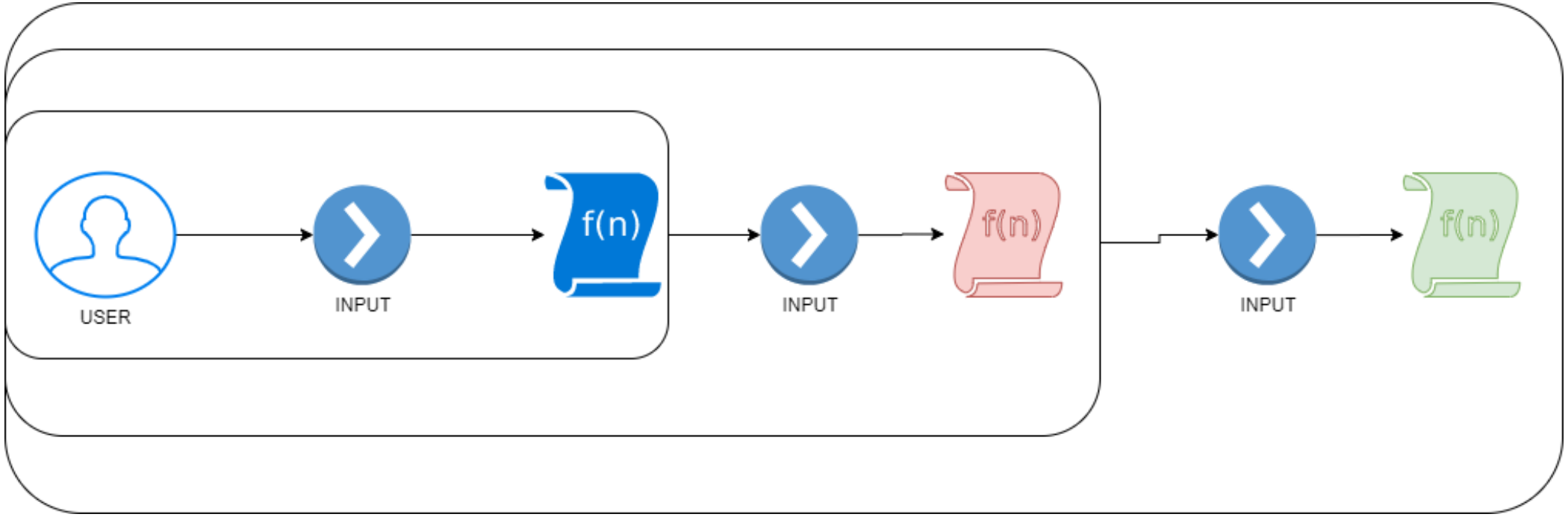

f(n)
f(n)
f(n)
USER
FN CALL
PROXY

Proxy ½

```
let GetValuePlus1 = arg1 => arg1 + 1;
let GetValueMinus1 = arg1 => arg1 - 1;

class Proxy {
    constructor(func1, func2) {
        this.cache = {};
        this.func1 = func1;
        this.func2 = func2;
    }

    getValue(arg1) {
        if (!this.cache[arg1]) {
            if (arg1 >= 0) {
                this.cache[arg1] = this.func1(arg1);
            } else if (arg1 < 0) {
                this.cache[arg1] = this.func2(arg1);
            }
        }
        return this.cache[arg1];
    }
}
```

Proxy 2/2

```
var proxy = new Proxy(GetValuePlus1, GetValueMinus1);

console.log(proxy.getValue(1));    2

console.log(proxy.getValue(2));    3

console.log(proxy.getValue(-1));  -2

console.log(proxy.getValue(1));    2

console.log(proxy.getValue("a")); undefined
```


USER
INPUT
f(n)
INPUT
f(n)
INPUT
f(n)

Compose

```
let sum = (x, y) => x + y;
let mul = (x, y) => x * y;

let x = sum(mul(2, 3), 5);

console.log(x);
```

11

Compose

```
let sum = (x, y) => x + y;
let mul = (x, y) => x * y;
let mulAndSum = (x, y, z) => sum(mul(x, y), z);
let x = mulAndSum(2, 3, 5);

console.log(x);
```

11

Compose

```
function sum(x, y) {
    return x + y;
}
function mul(x, y) {
    return x * y;
}

function compose(f1, f2) {
    return function fc() {
        var args = [].slice.call(arguments);
        return f2(f1(args.shift(), args.shift()), args.shift());
    };
}

let mulAndSum = compose(
    mul,
    sum
);

let x = mulAndSum(2, 3, 5);
console.log(x);
```

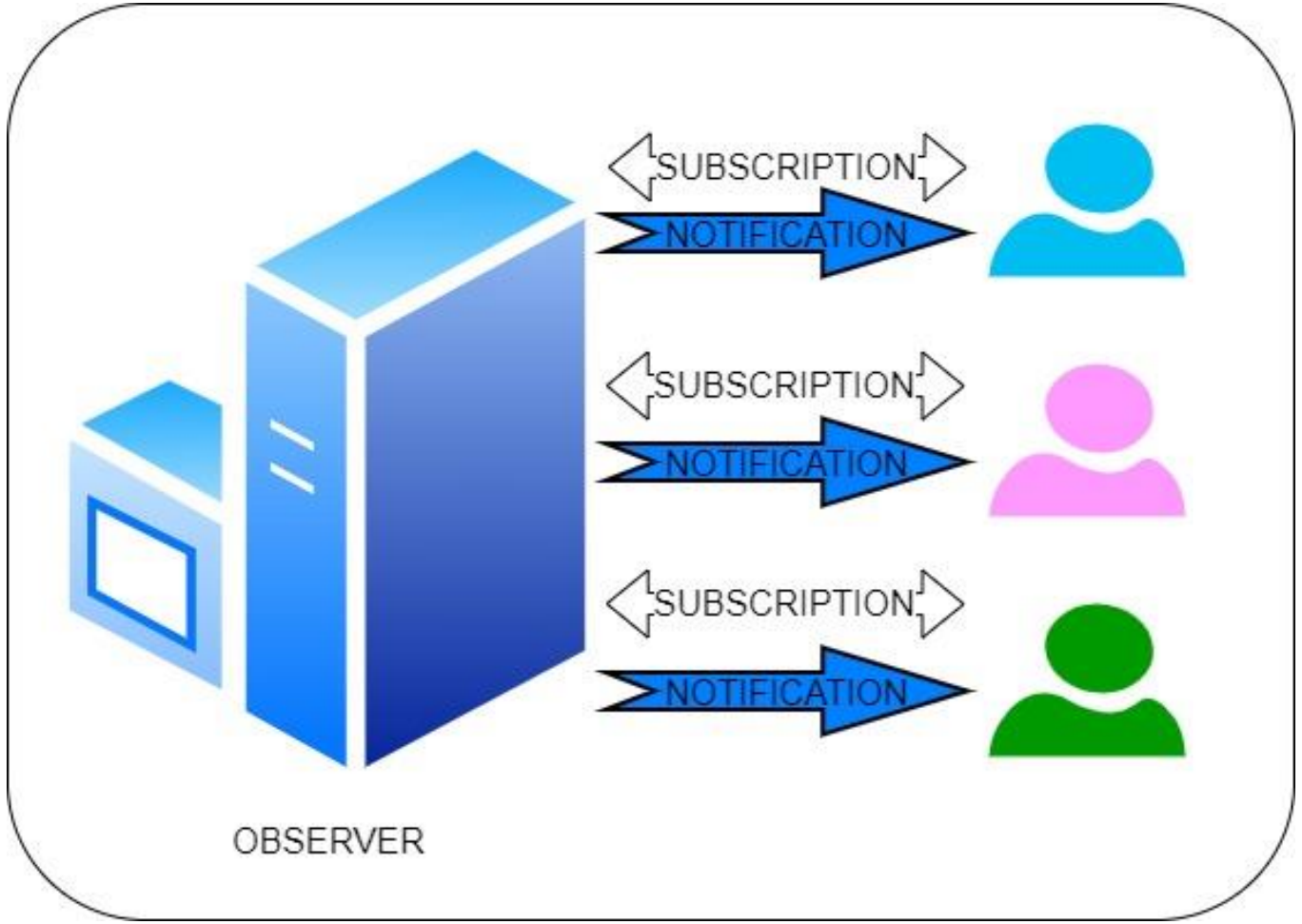
Compose

```
let sum = (x, y) => x + y;
let mul = (x, y) => x * y;

function compose(f1, f2) {
    return function fc() {
        var args = [].slice.call(arguments);
        return f2(f1(args.shift(), args.shift()), args.shift());
    };
}

let x = compose(
    mul,
    sum
)(2, 3, 5);

console.log(x);
```

11

Compose

```
function Mul1(x) {
    return x * 1;
}
function Mul2(x) {
    return x * 2;
}
function Mul3(x) {
    return x * 3;
}
function composeRecurent(...funcs) {
    return function(x) {
        if (funcs.length) {
            return funcs.pop()(composeRecurent(...funcs)(x));
        }
        return x;
    };
}

let compFunc = composeRecurent(Mul1, Mul2, Mul3);
console.log(compFunc(2));
```


SUBSCRIPTION
NOTIFICATION
SUBSCRIPTION
NOTIFICATION
SUBSCRIPTION
NOTIFICATION
OBSERVER

Observer 1/2

```
let Show1 = _ => console.log("Show 1");

let Show2 = x => console.log("Show 2. Arg " + x);

let Show3 = (x, y, z) => console.log("Show 3. Arg " + x + ", " + y + ", " + z);

class Subscriber {
    constructor() {
        this.observers = [];
    }

    subscribe(func) {
        this.observers.push(func);
    }
    unsubscribe(func) {
        this.observers = this.observers.filter(fn => fn !== func);
    }
    fire(...arg) {
        this.observers.forEach(fn => fn(...arg));
    }
}
```

Observer 2/2

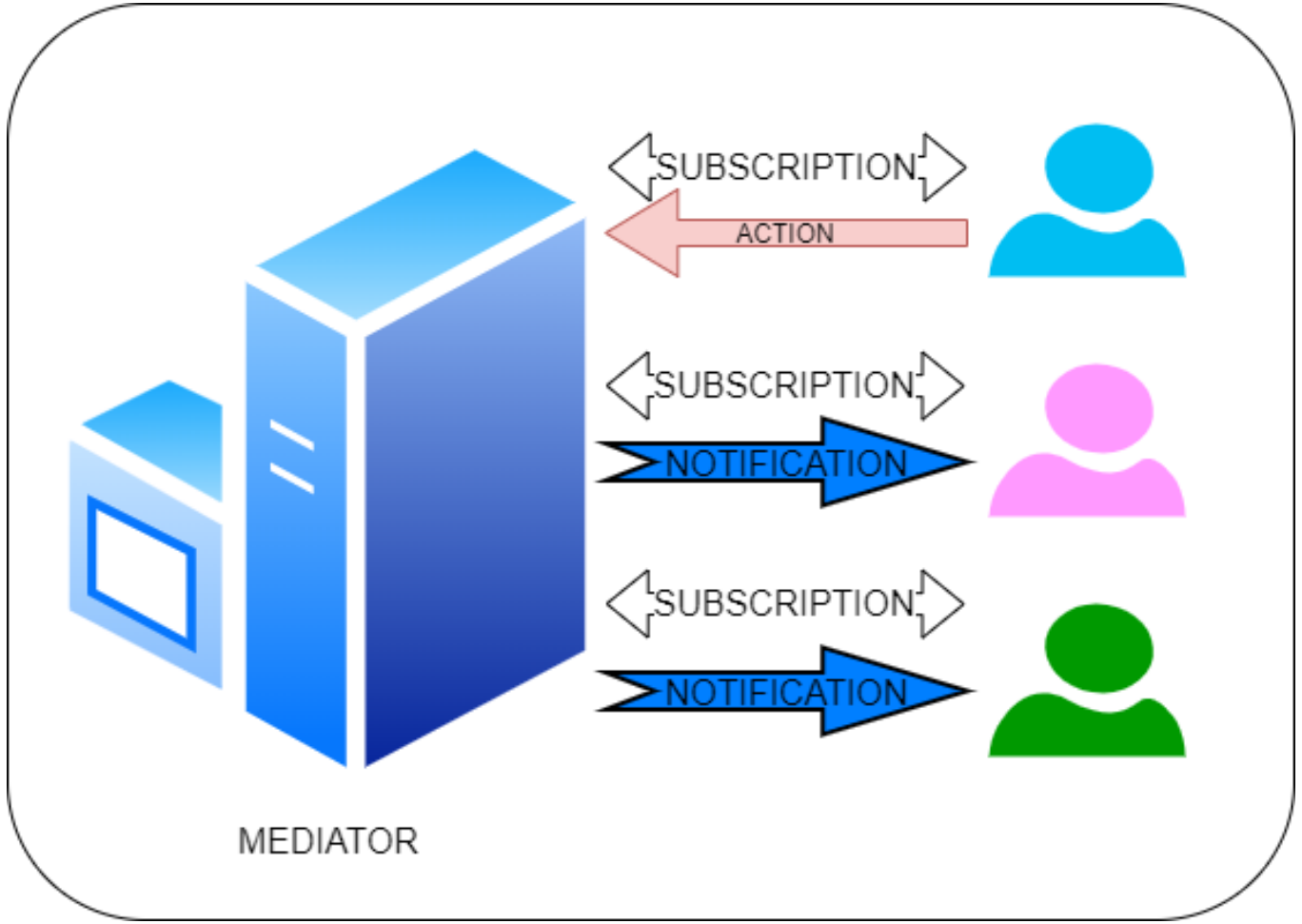
```
let subscriber = new Subscriber();
subscriber.subscribe(Show1);
subscriber.subscribe(Show2);
subscriber.subscribe(Show2);
subscriber.subscribe(Show3);
```

```
subscriber.fire("a", "b", "c");
```

Show 1
Show 2. Arg a
Show 2. Arg a
Show 3. Arg a, b, c

```
subscriber.unsubscribe(Show2);
subscriber.fire("a", "b", "c");
```

Show 1
Show 3. Arg a, b, c


SUBSCRIPTION
ACTION
SUBSCRIPTION
NOTIFICATION
SUBSCRIPTION
NOTIFICATION
MEDIATOR

Mediator 1/3

```
class User {
    constructor(userName) {
        this.userName = userName;
        this.charRoom = null;
    }

    receiveMessage(message) {
        console.log(this.userName + " <== " + message);
    }

    sendMessage(message) {
        console.log(this.userName + " ==> " + message);
        this.charRoom.sendMessage(message, this);
    }
}
```

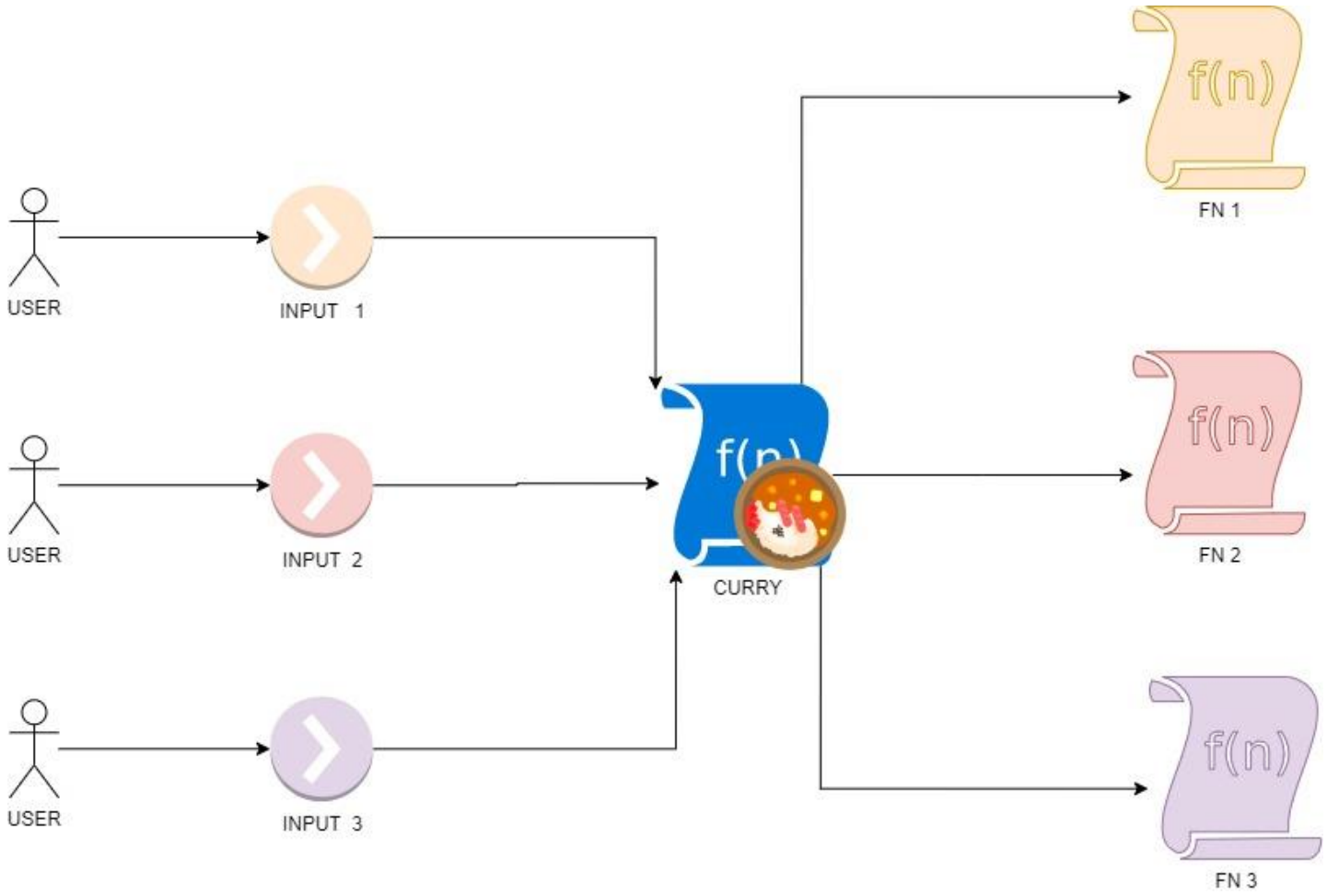
Mediator 2/3

```
class ChatRoom {
    constructor(chatRoomName) {
        this.chatRoomName = chatRoomName;
        this.users = [];
    }

    sendMessage(message, user) {
        this.users.filter(u => u !== user).forEach(u => u.receiveMessage(message));
    }

    regisgerUser(user) {
        if (this.users.indexOf(user) === -1) {
            this.users.push(user);
            user.charRoom = this;
        }
    }
}
```

Mediator 3/3

```
let charRoom = new ChatRoom("Mediator Chatroom");
let userKarol = new User("Karol");
let userAdam = new User("Adam");
let userAnna = new User("Anna");


charRoom.regisgerUser(userKarol);
charRoom.regisgerUser(userAdam);
charRoom.regisgerUser(userAnna);

userAnna.sendMessage("Hi");
```

```
Anna ==> Hi
Karol <== Hi
Adam <== Hi
```


USER
INPUT 1
USER
INPUT 2
USER
INPUT 3
f(n)
CURRY
f(n)
FN 1
f(n)
FN 2
f(n)
FN 3

Curry

```
function f() {
    let count = 0;

    return function() {
        return ++count;
    };
}

let x = f();
let y = f();

console.log("x_" + x());    x_1
console.log("x_" + x());    x_2

console.log("y_" + y());    y_1
console.log("y_" + y());    y_2
console.log("y_" + y());    y_3
```

Curry

```
function s(x) {
    return function(y) {
        return x + y;
    };
}

let sumWith5 = s(5);
console.log(sumWith5(1)); 6
console.log(sumWith5(2)); 7

let sumWith1 = s(1);
console.log(sumWith1(1)); 2
console.log(sumWith1(2)); 3
```

Curry

```
function s(x, y) {
    return function() {
        return x + y;
    };
}

let sum5and1 = s(5, 1);
console.log(sum5and1());  6
console.log(sum5and1());  6
```