Using Subjects and Multicasted Observables

Brice Wilson

@brice_wilson www.BriceWilson.net

What Are Subjects?

Observables

Observers

Produce values

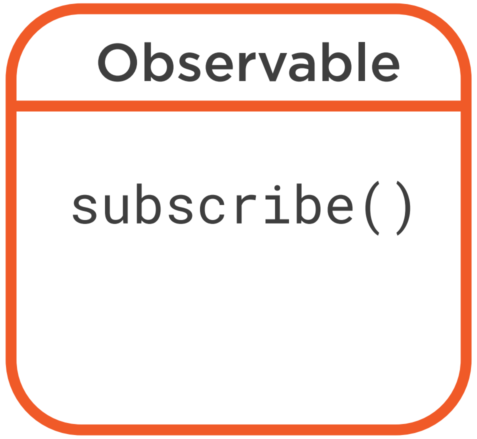
Proxy values

Have state and maintain a list of observers

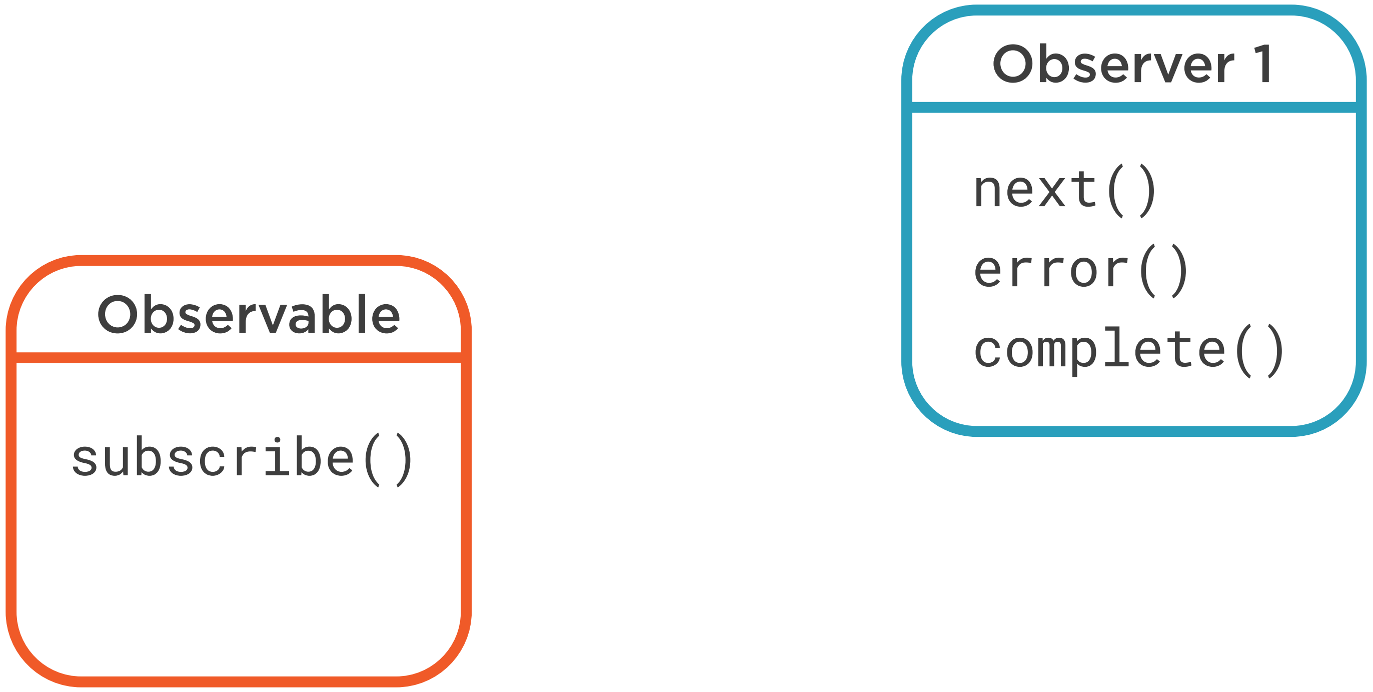
Multicast instead of unicast

What Are Subjects?

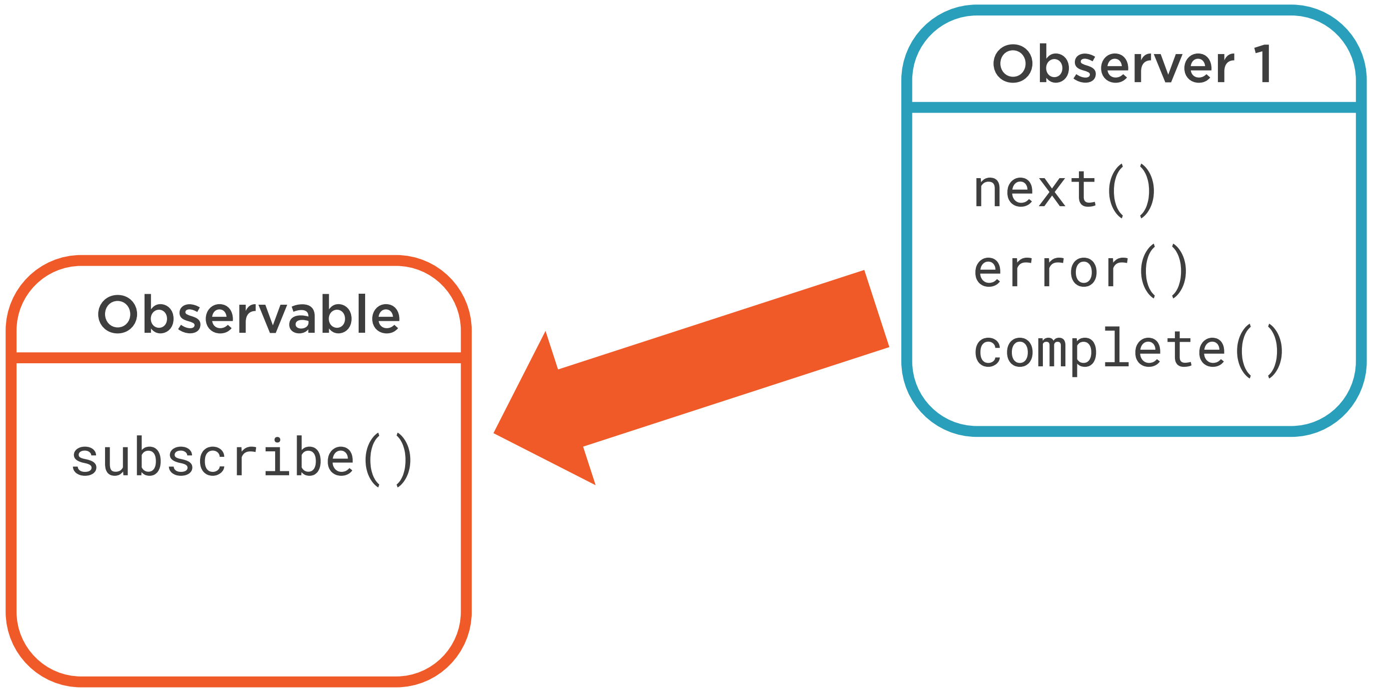
What Are Subjects?



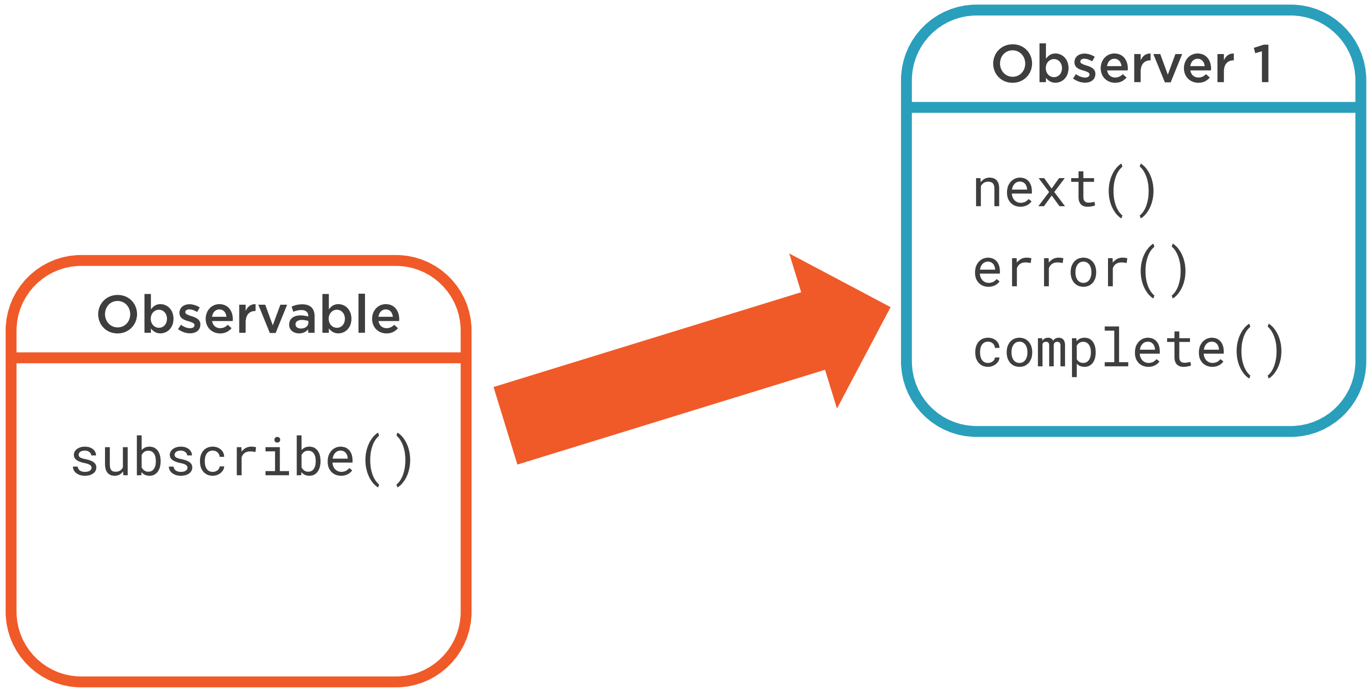
What Are Subjects?



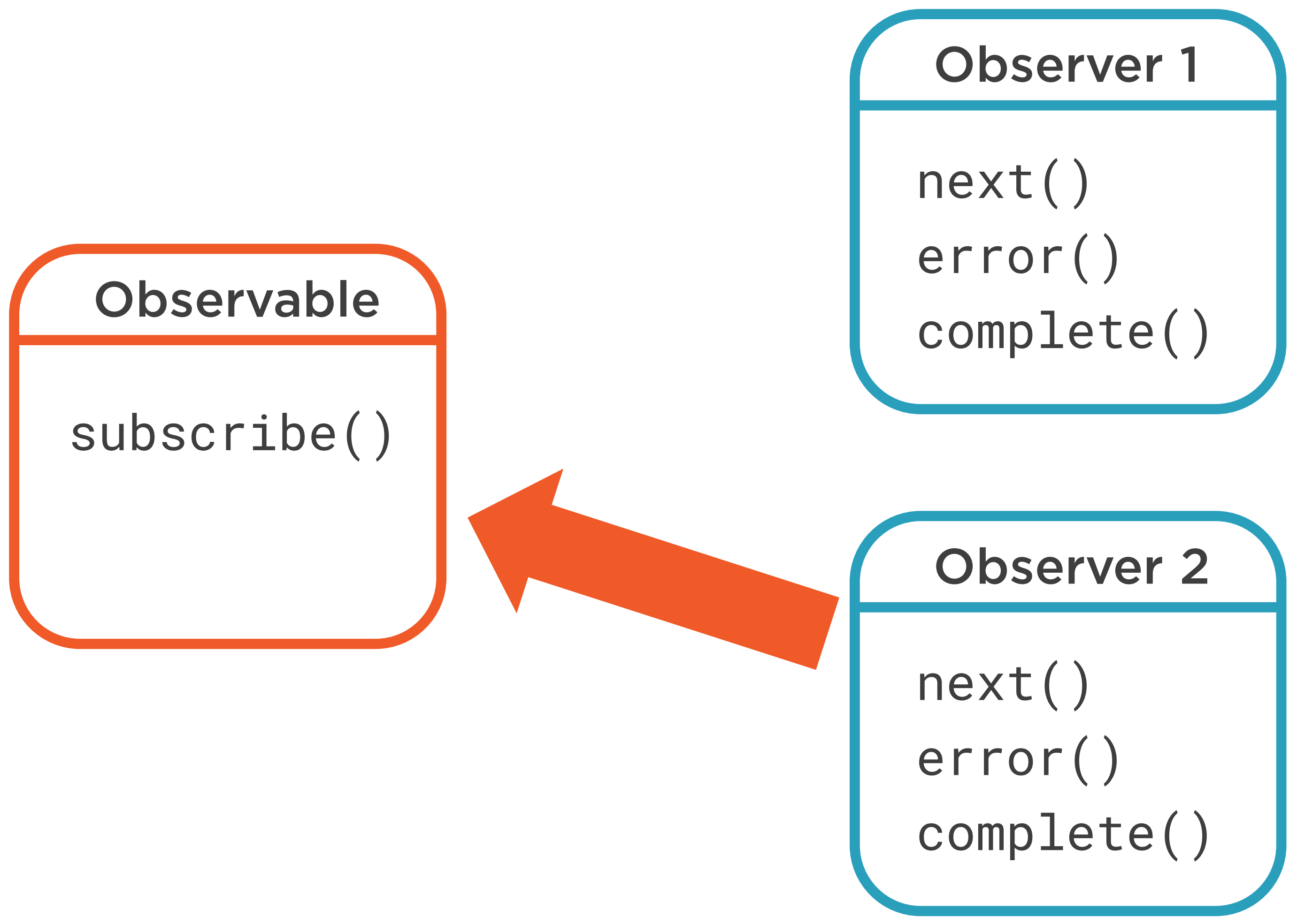
What Are Subjects?



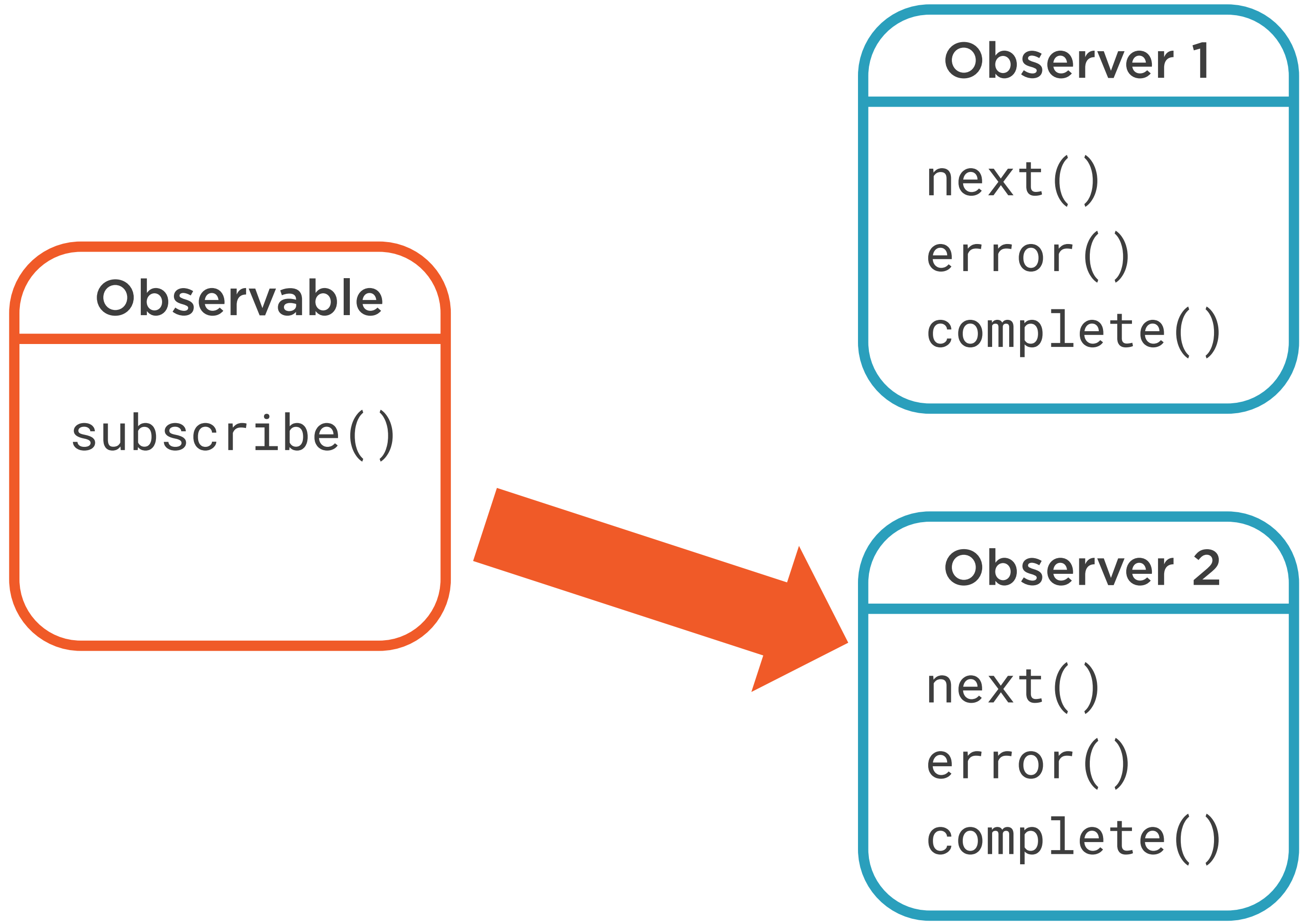
What Are Subjects?



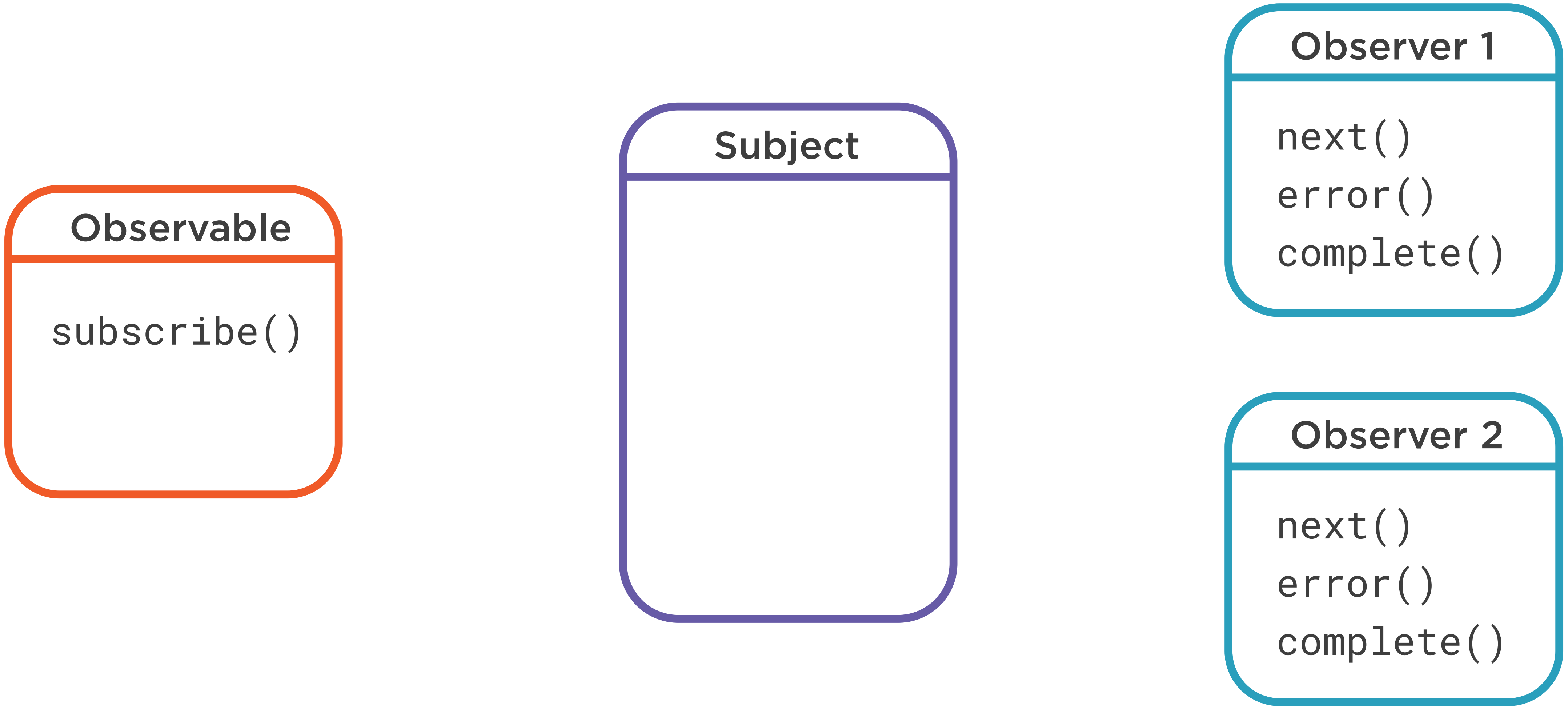
What Are Subjects?



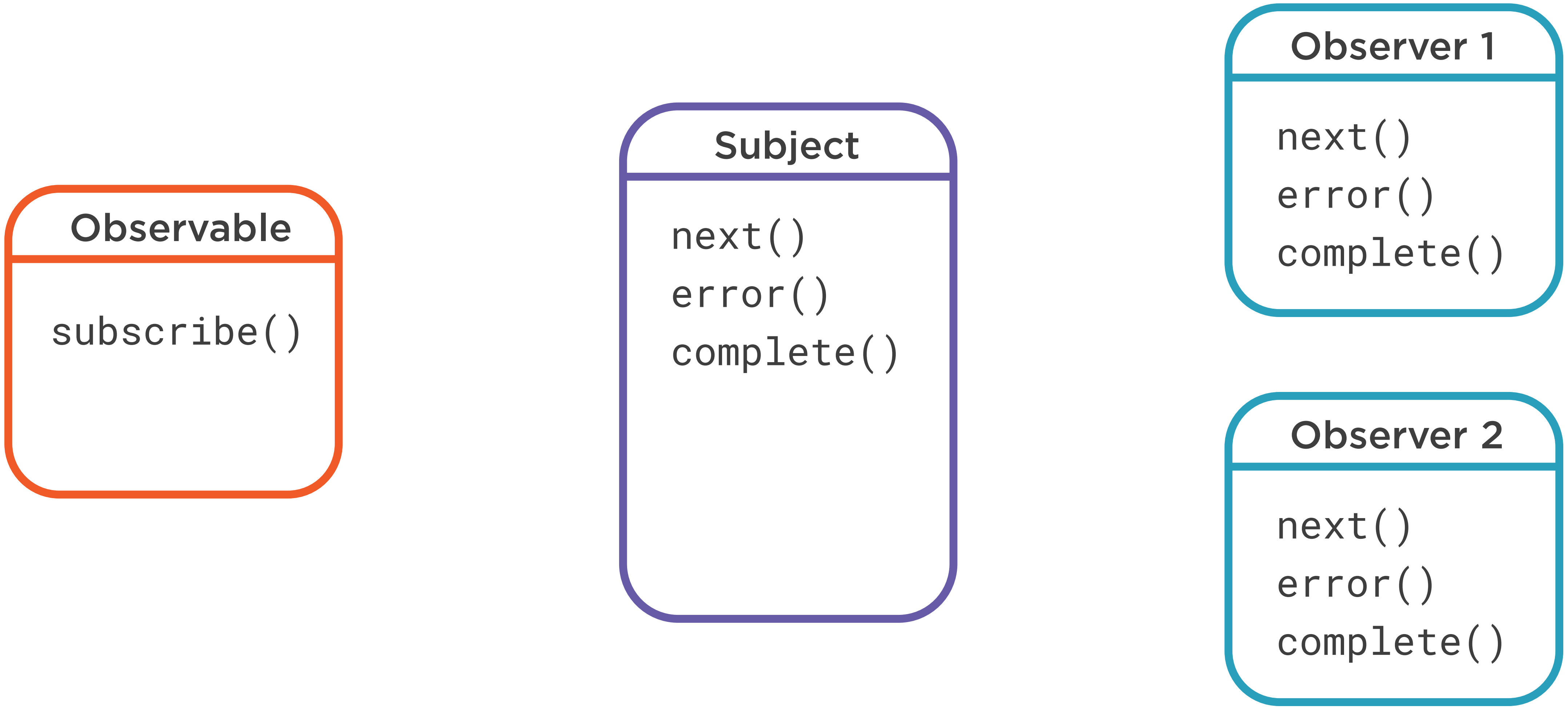
What Are Subjects?



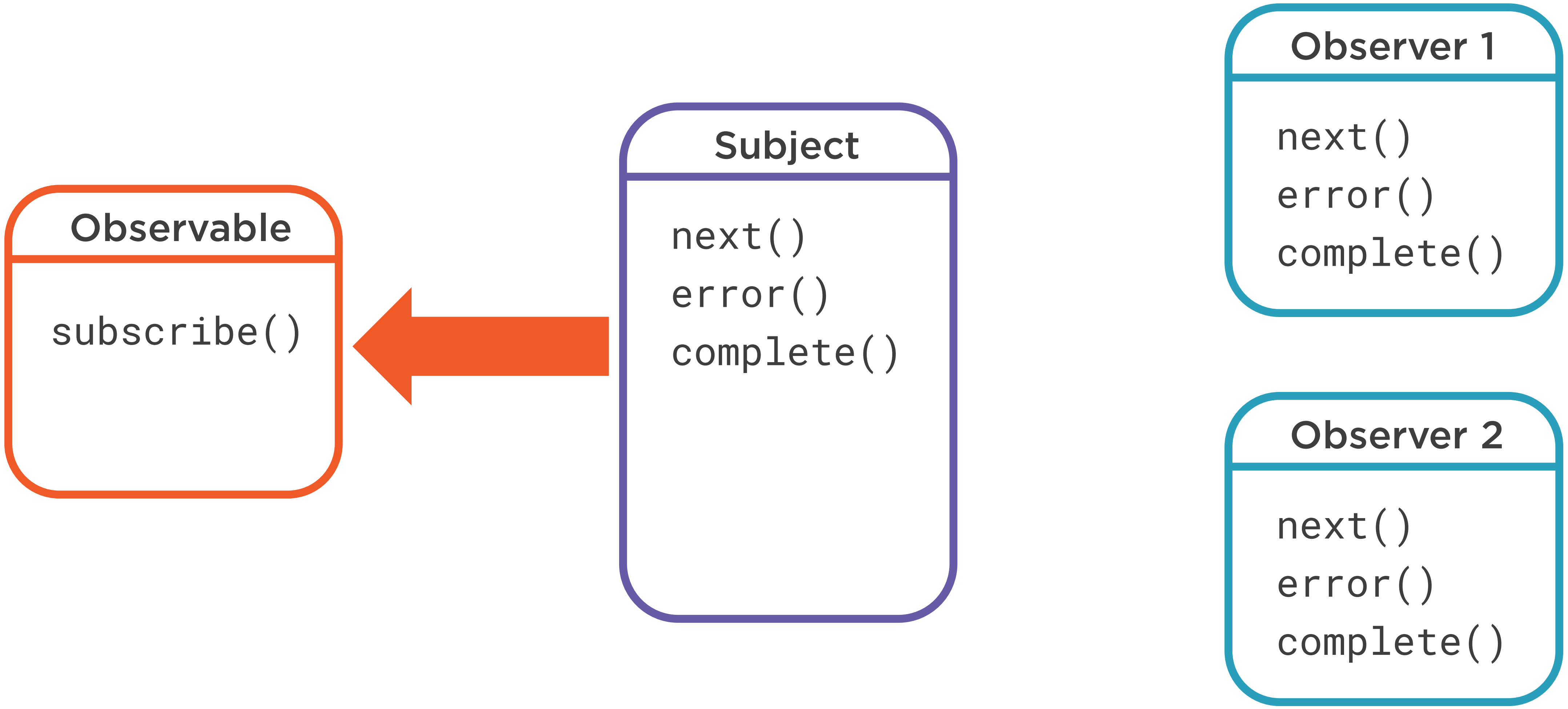
What Are Subjects?



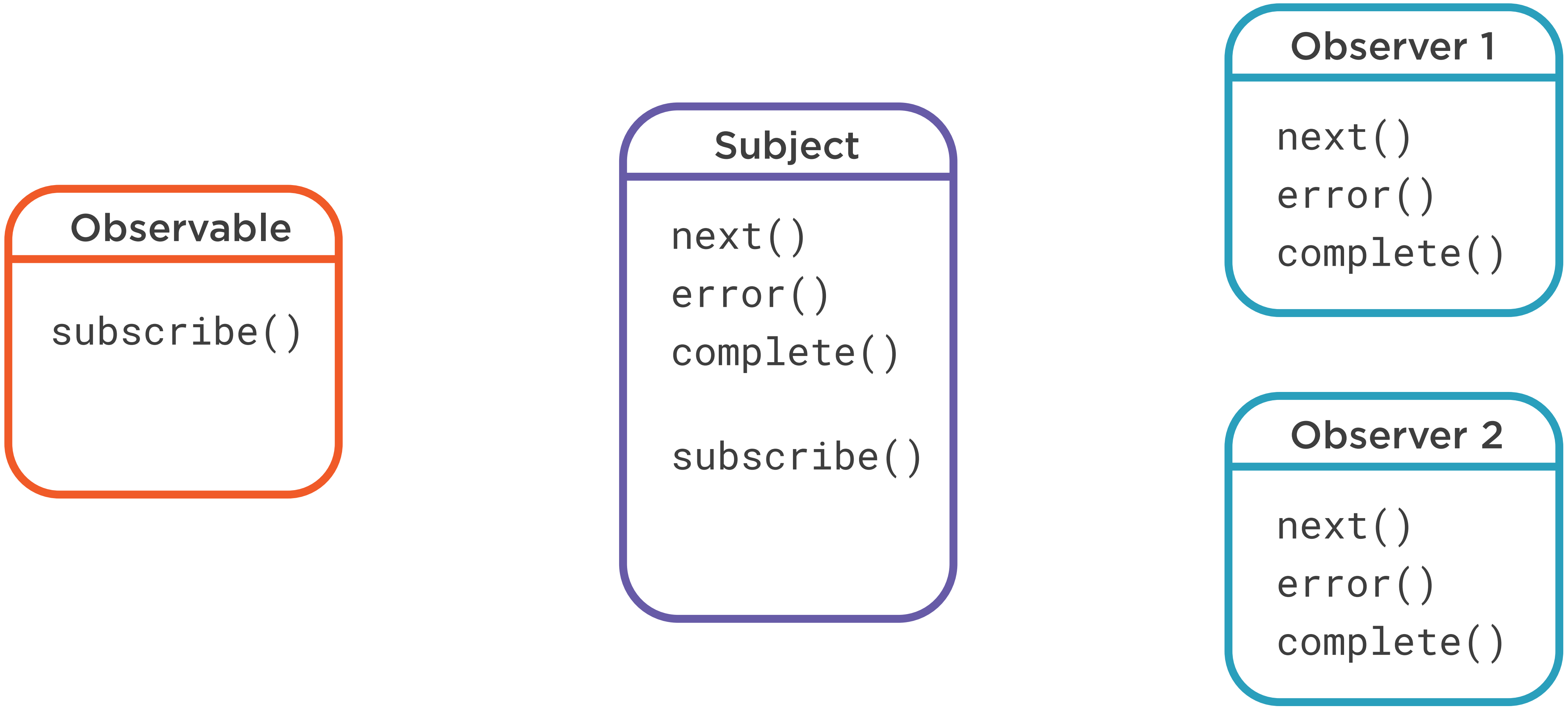
What Are Subjects?



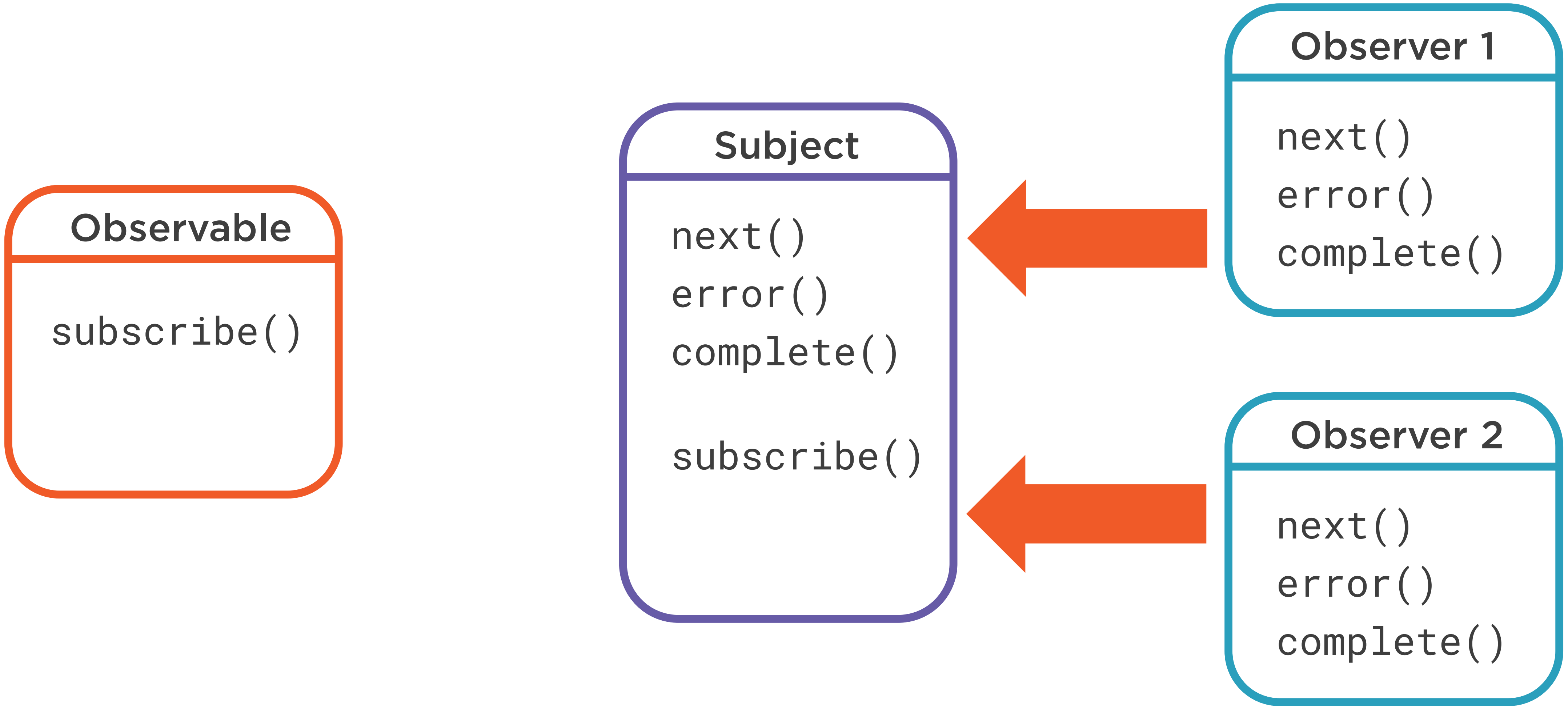
What Are Subjects?



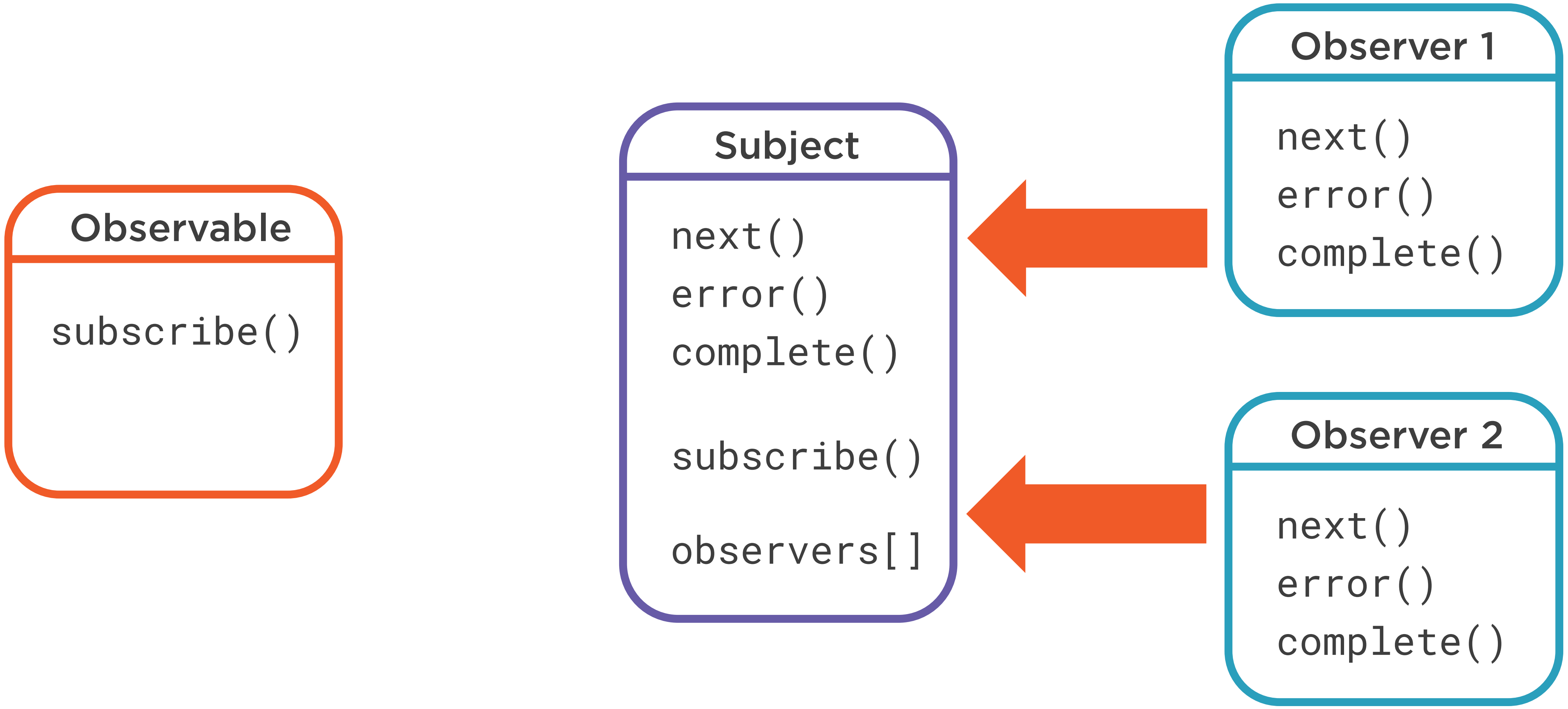
What Are Subjects?



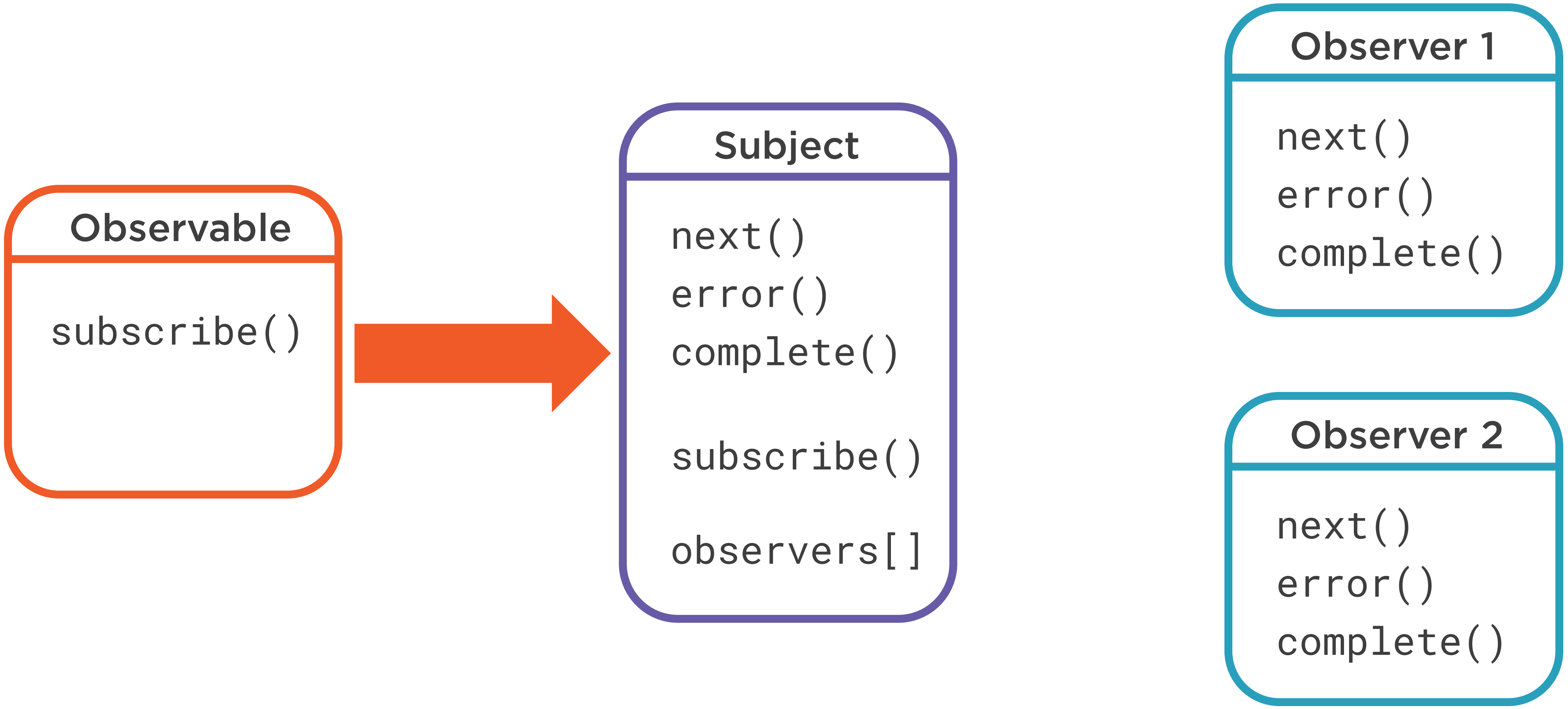
What Are Subjects?



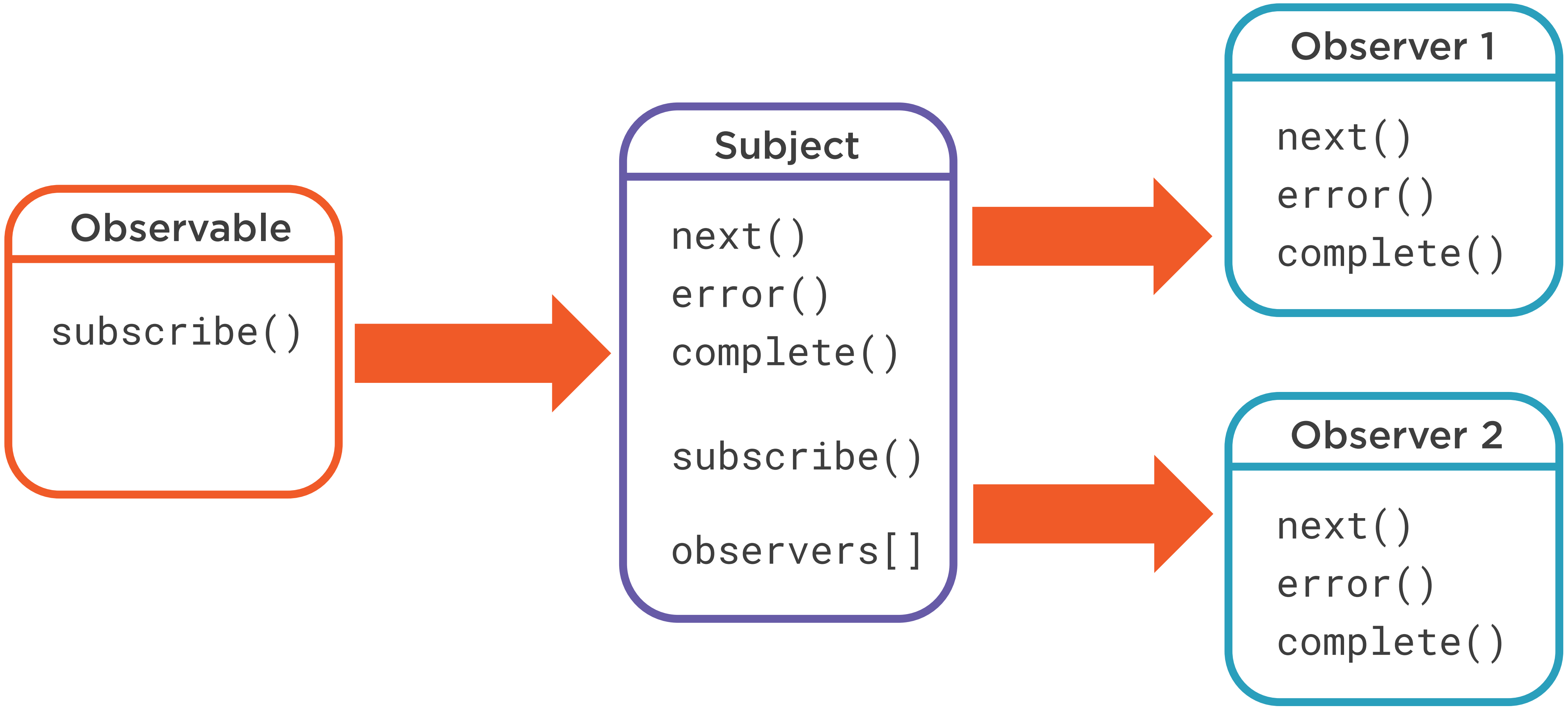
What Are Subjects?



What Are Subjects?



What Are Subjects?



Cold vs. Hot Observables

Cold Observables

Value producer created inside the observable

One observer per execution

Unicast

Examples include interval(), ajax()

Hot Observables

Value producer exists outside the observable

Shared producer allows for multiple observers

Multicast

Examples include Observables that wrap DOM events, WebSockets

Multicasting Operators

multicast()

- Takes a Subject as a parameter
- Must call connect() to begin execution

refCount()

- Executes when observers > 0

publish()

- Thin wrapper about multicast()
- Not required to pass it a Subject

share()

- Executes when observers > 0
- Re-subscribes as necessary

Specialized Subjects

AsyncSubject

Only emits the last value received

Used by the publishLast() operator

BehaviorSubject

Emits initial seed value if source has not yet produced a value

Emits most recent value otherwise

Used by the publishBehavior() operator

ReplaySubject

Stores and emits multiple values to all observers

Used by the publishReplay() operator