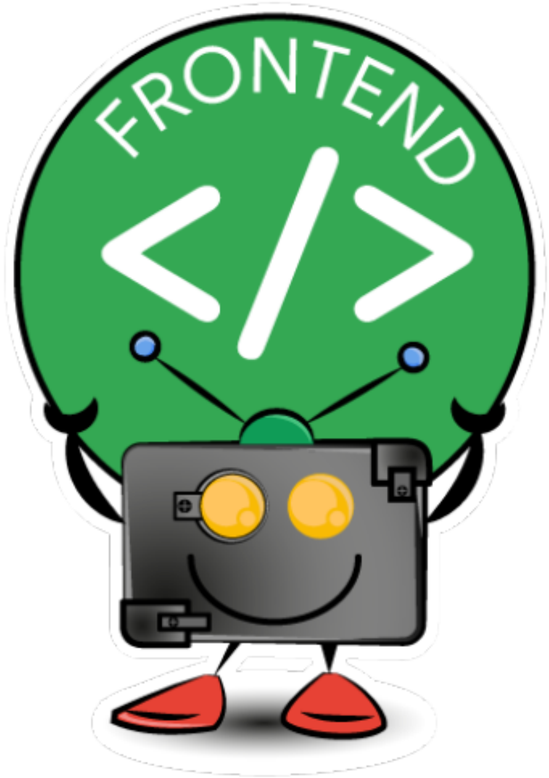
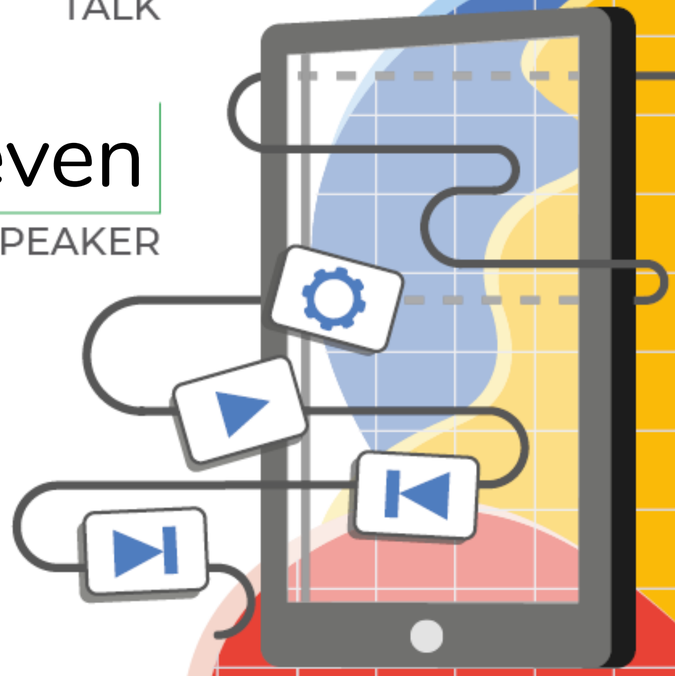


Imparare a nuotare con **RxJS**

TALK

Michele Stieven

SPEAKER



DEVPARTY TOGETHER 2020

GDG Italia

Michele Stieven

Consulente e Sviluppatore Front-End

www.accademia.dev

Accademia DEV
WEB DEVELOPER ACADEMY

Reactive Extensions for JavaScript


marble

Imparare a nuotare con RxJS

Programmazione Reattiva?

reactive

/rɪˈaktɪv/

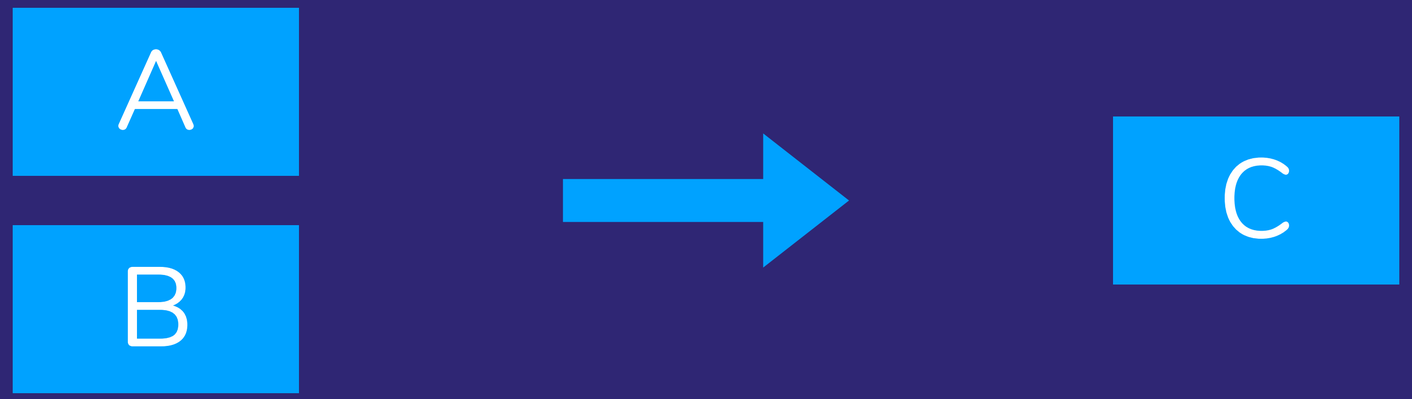
adjective

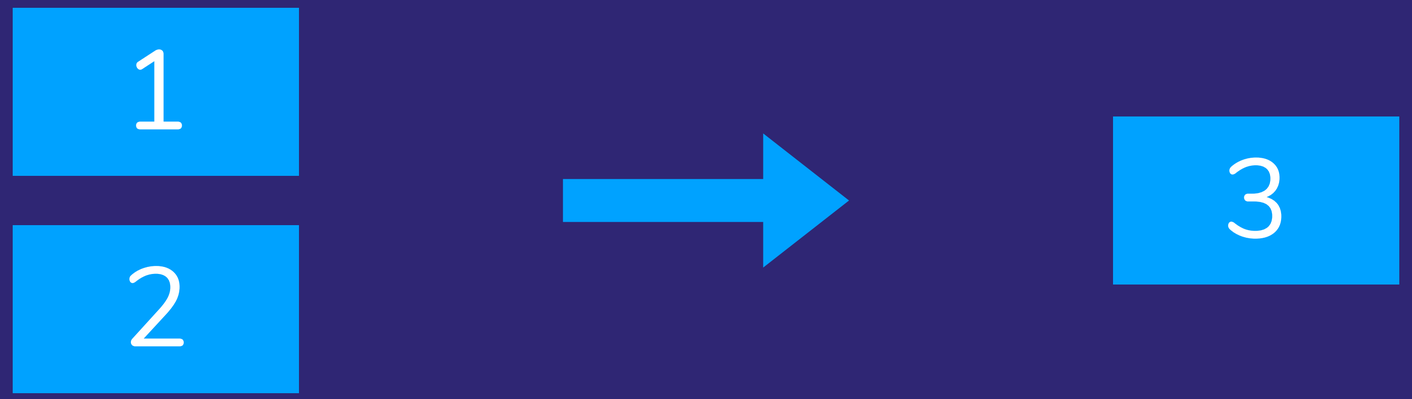
1. showing a response to a stimulus.
   "pupils are reactive to light”

2. acting in response to a situation rather than creating or controlling it.
   "a proactive rather than a reactive approach"

Architetture **Pull** e **Push**

Descrivono il modo in cui viene propagato un cambiamento di stato.


A
B
C

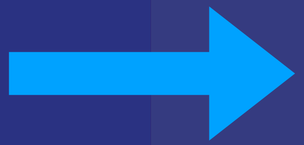

1
2
3

Stato

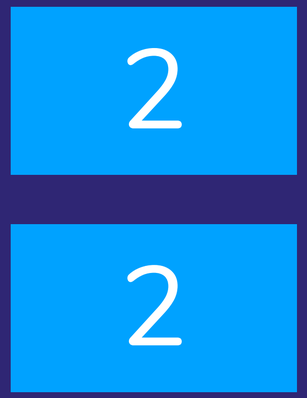
Stato derivato

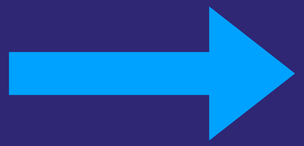
1

2



3


2
2




3

```
let a = 1;
let b = 2;

let c = a + b;

a = 2;
c = a + b;
```

Architettura **Pull**

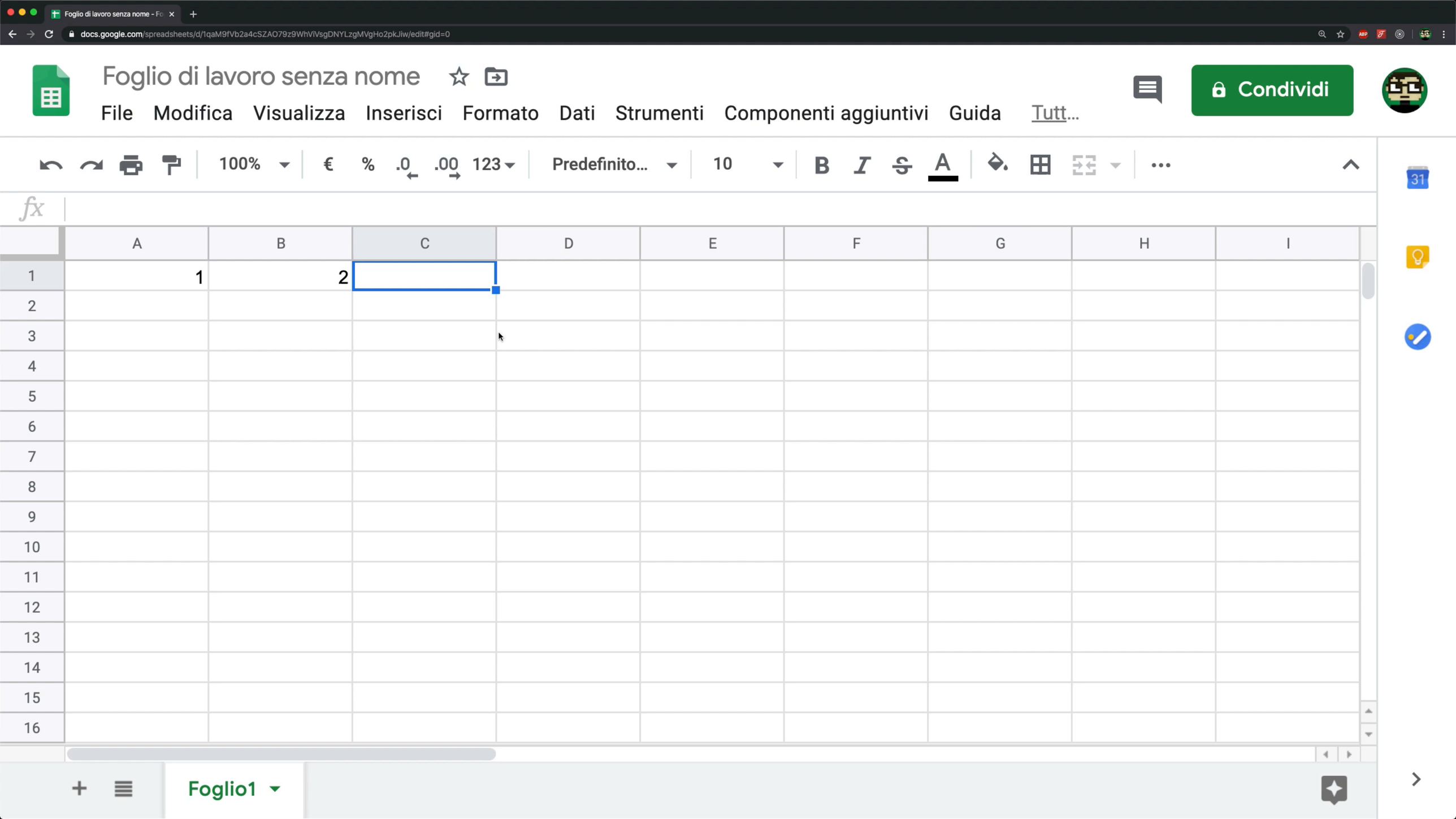
Se lo stato cambia, il nuovo valore dev’essere richiesto manualmente.

Architettura **Pull**

- Più semplice per programmatori meno esperti
- Più performante (generalmente)
- Il consumatore non sa quando lo stato cambia...
  - Rischio di bug (es. UI non consistente)
  - Rischio di performance peggiori (es. troppe chiamate inutili se lo stato non cambia)
  - Codice ripetitivo, più difficile da leggere e da mantenere
- Ideale per operazioni one-time in cui lo scorrere del tempo non è importante

Architettura **Push**

Se lo stato cambia, veniamo notificati in automatico dal produttore del dato.


Foglio di lavoro senza nome - Fo
docs.google.com/spreadsheets/d/1qaM9fVb2a4cSZAO79z9WhViVsgDNYLzgMVgHo2pkJiw/edit#gid=0
Foglio di lavoro senza nome
File Modifica Visualizza Inserisci Formato Dati Strumenti Componenti aggiuntivi Guida Tutt...
Condividi
100% € % .0 .00 123 Predefinito... 10
fx
A B C D E F G H I
1 1 2
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Foglio1

Programmazione Reattiva

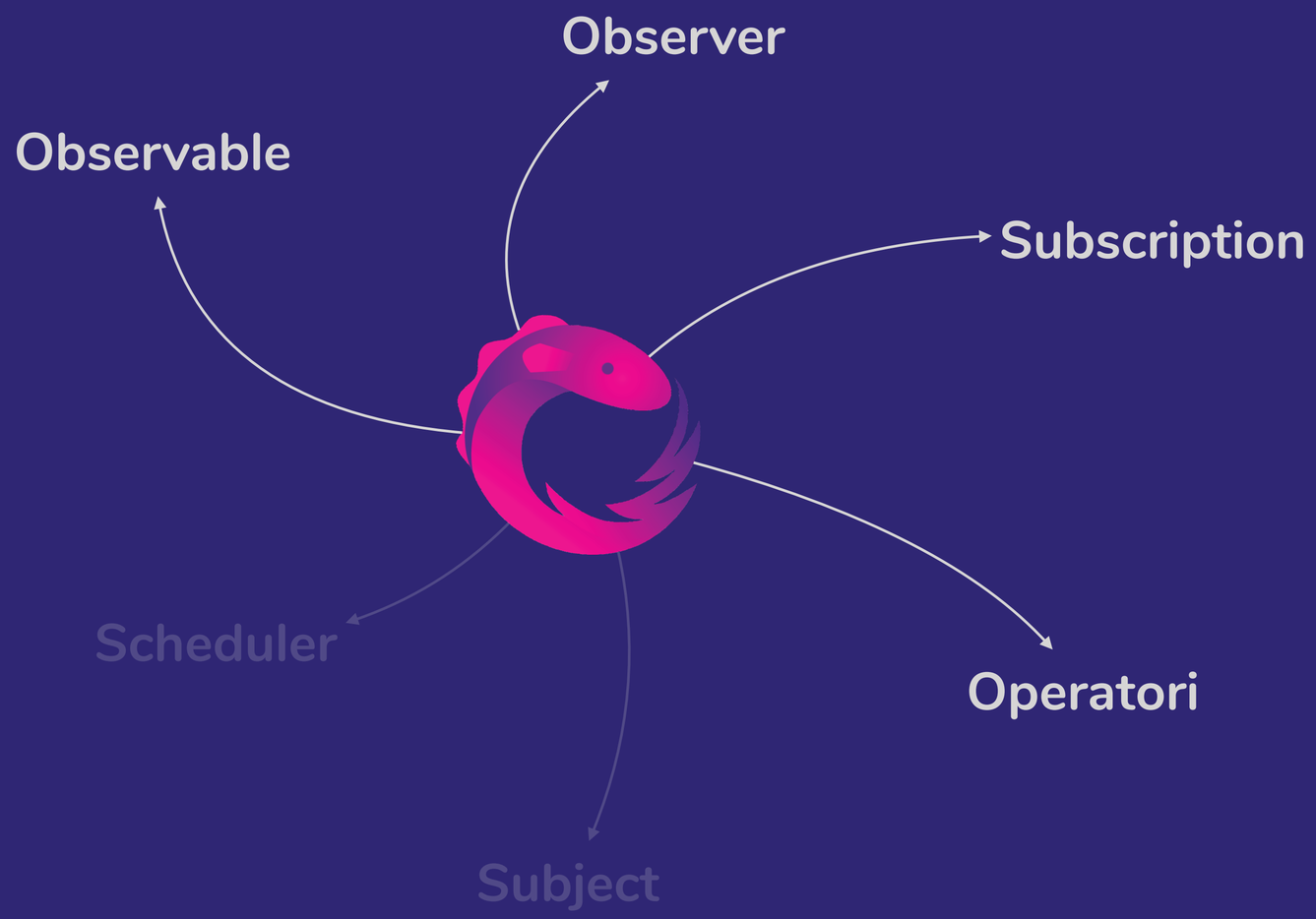
Stile di programmazione in cui utilizziamo un’architettura push per ascoltare degli eventi nel tempo e reagire di conseguenza.

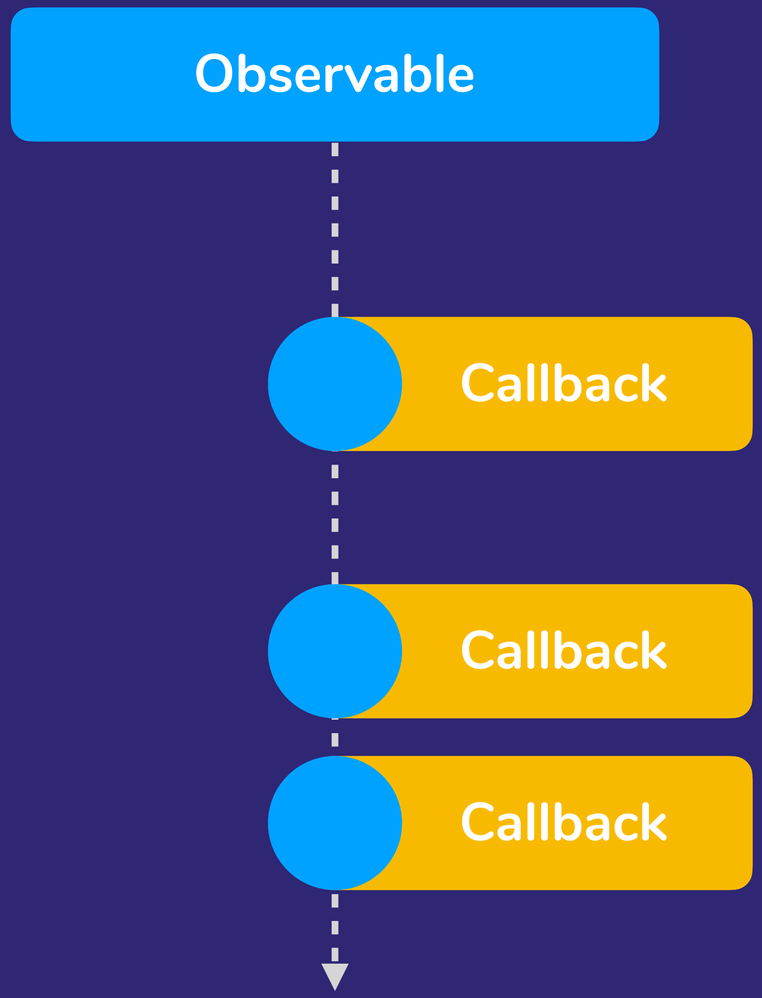
Programmazione Reattiva

Paradigma

Stile di programmazione in cui utilizziamo un’architettura push per ascoltare degli eventi nel tempo e reagire di conseguenza.

Stream (flusso)


Observer
Observable
Subscription
Scheduler
Operatori
Subject


Observable
Callback
Callback
Callback

Subscription

Observable

Observer

source$.subscribe(callback)

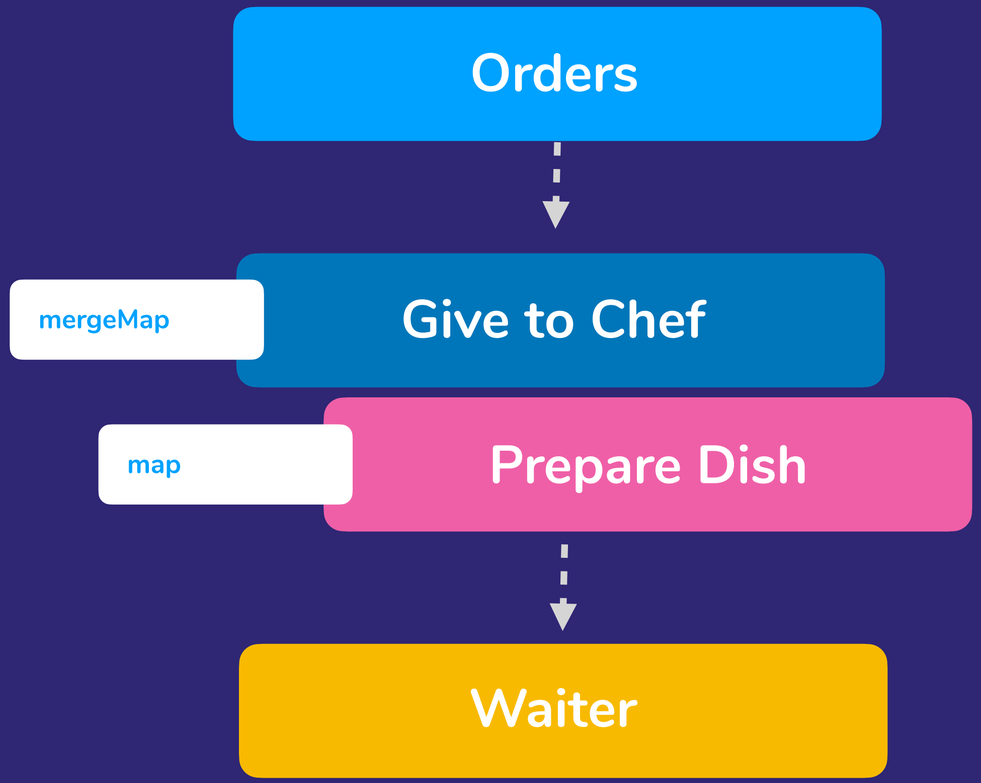
el.addEventListener(callback)

Cosa possiamo farci?

Cosa possiamo farci?

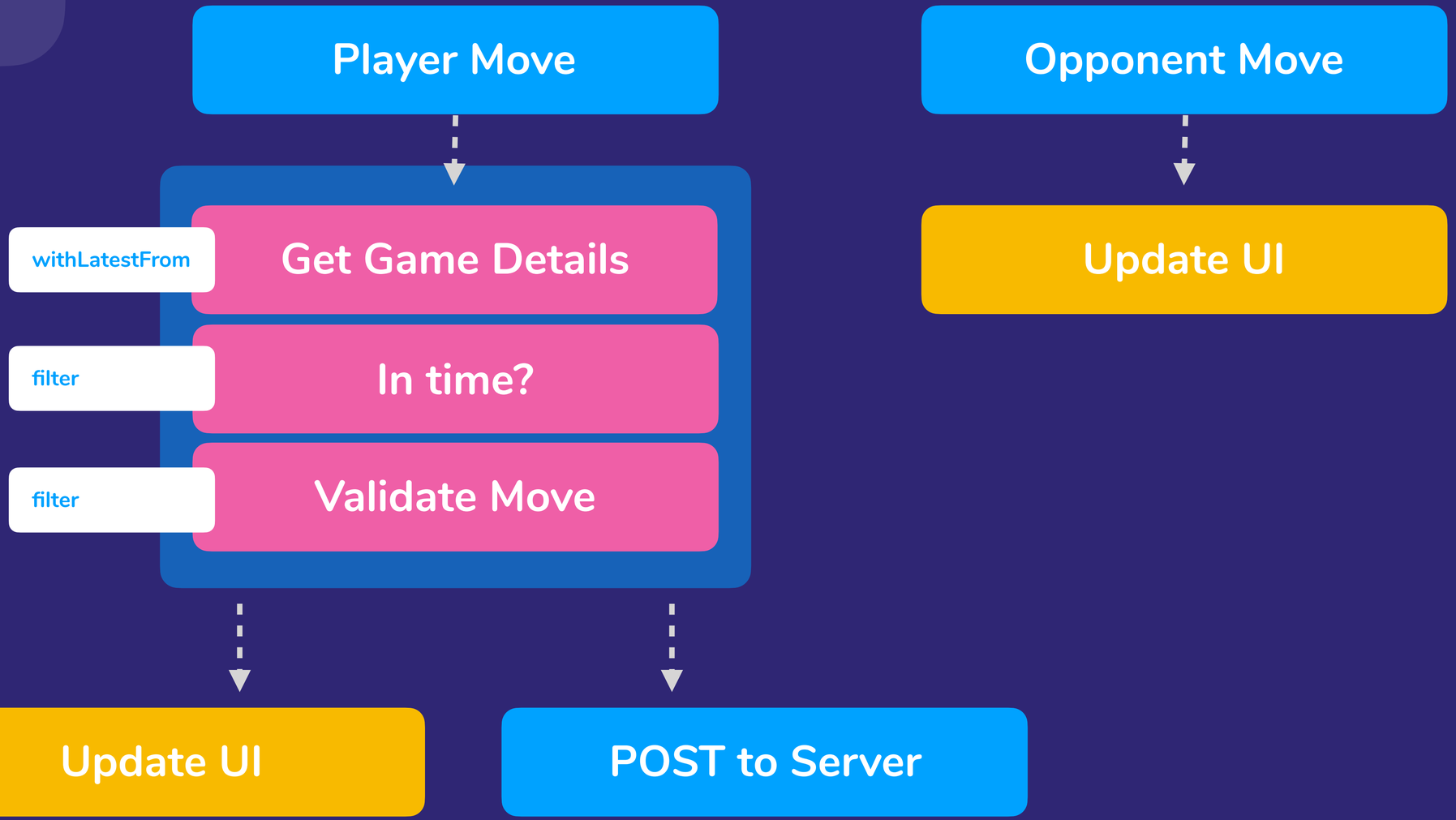
TUTTO

Ristorante



Macchina del caffè

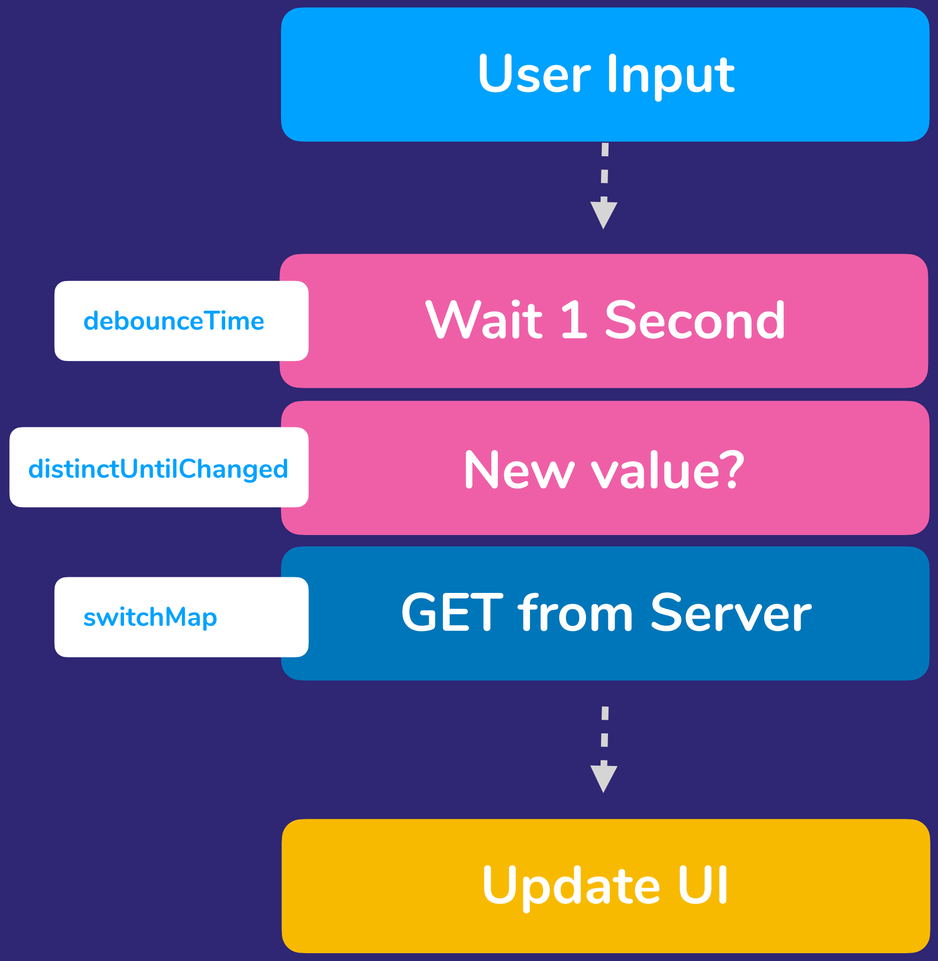
Partita a scacchi



Partita a scacchi

```
const validMove$ = playerMove$.pipe(
    withLatestFrom(gameDetails$),
    filter(([move, game]) => isInTime(game))
    filter(isValidMove)
);

validMove$.subscribe(updateUI);

validMove$.subscribe(postMove);

opponentMove$.subscribe(updateUI);
```

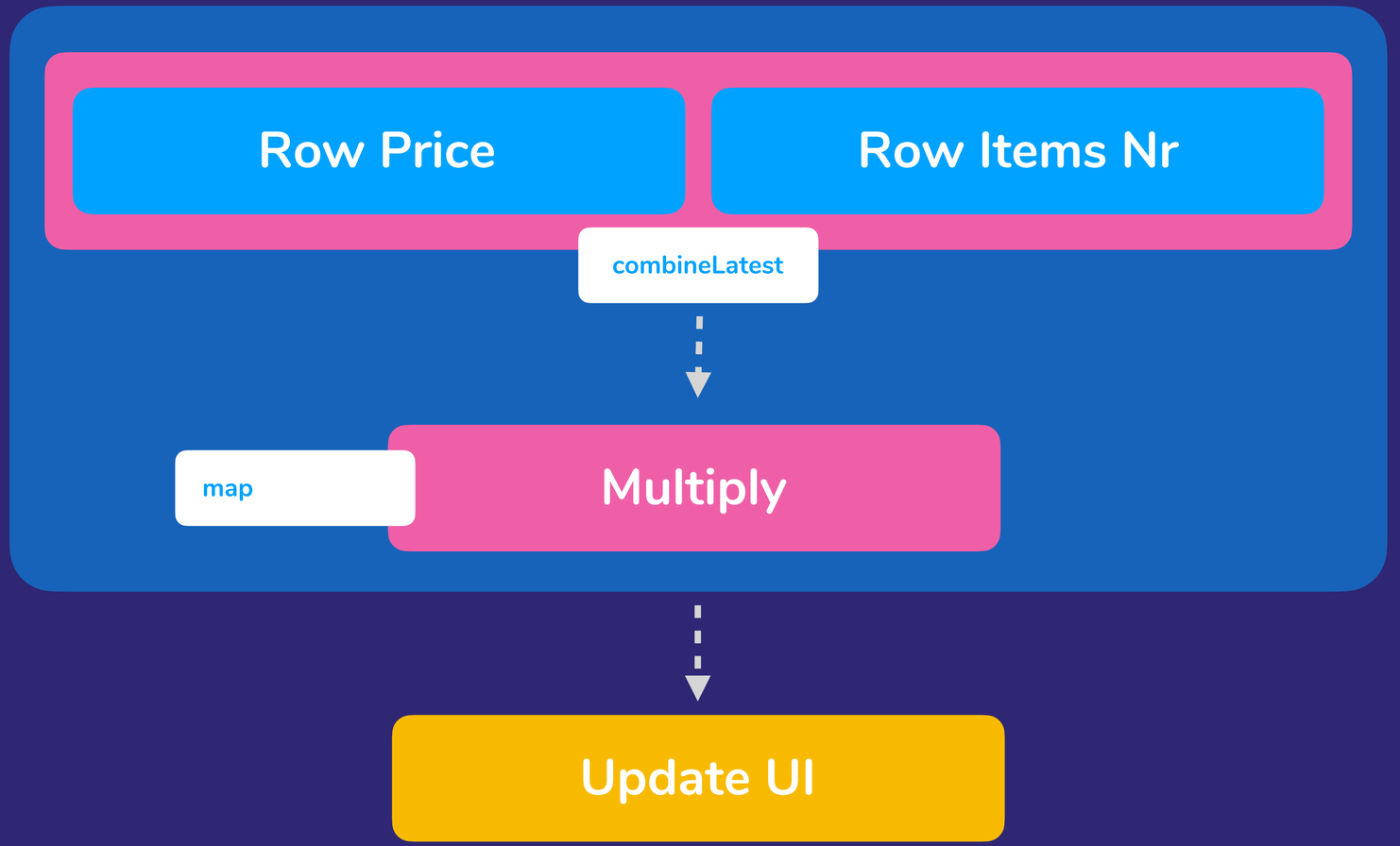
Lookup Form


User Input
debounceTime
Wait 1 Second
distinctUntilChanged
New value?
switchMap
GET from Server
Update UI

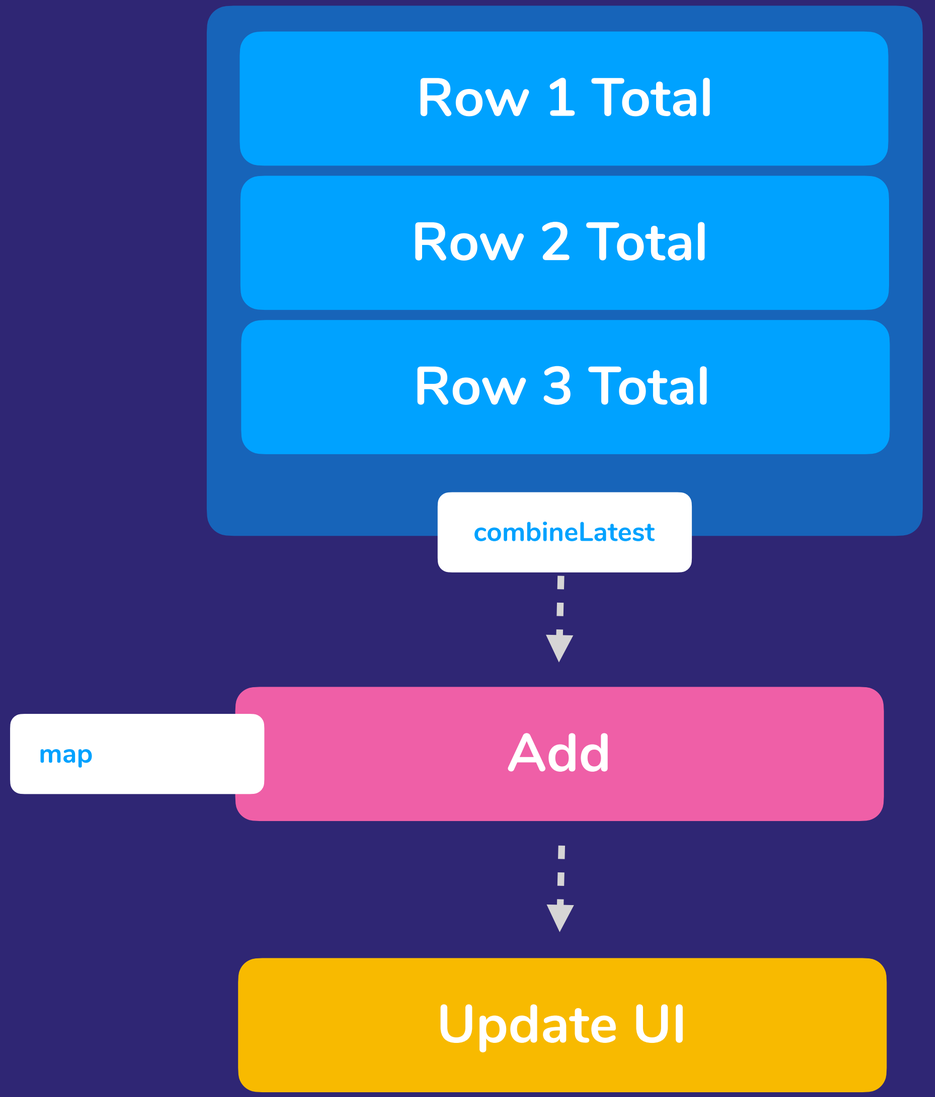
Lookup Form

```
const lookup$ = userInput$.pipe(
    debounceTime(1000),
    distinctUntilChanged()
    switchMap(getResults)
);

const getResults = x => ajax(...);

lookup$.subscribe(updateUI);
```

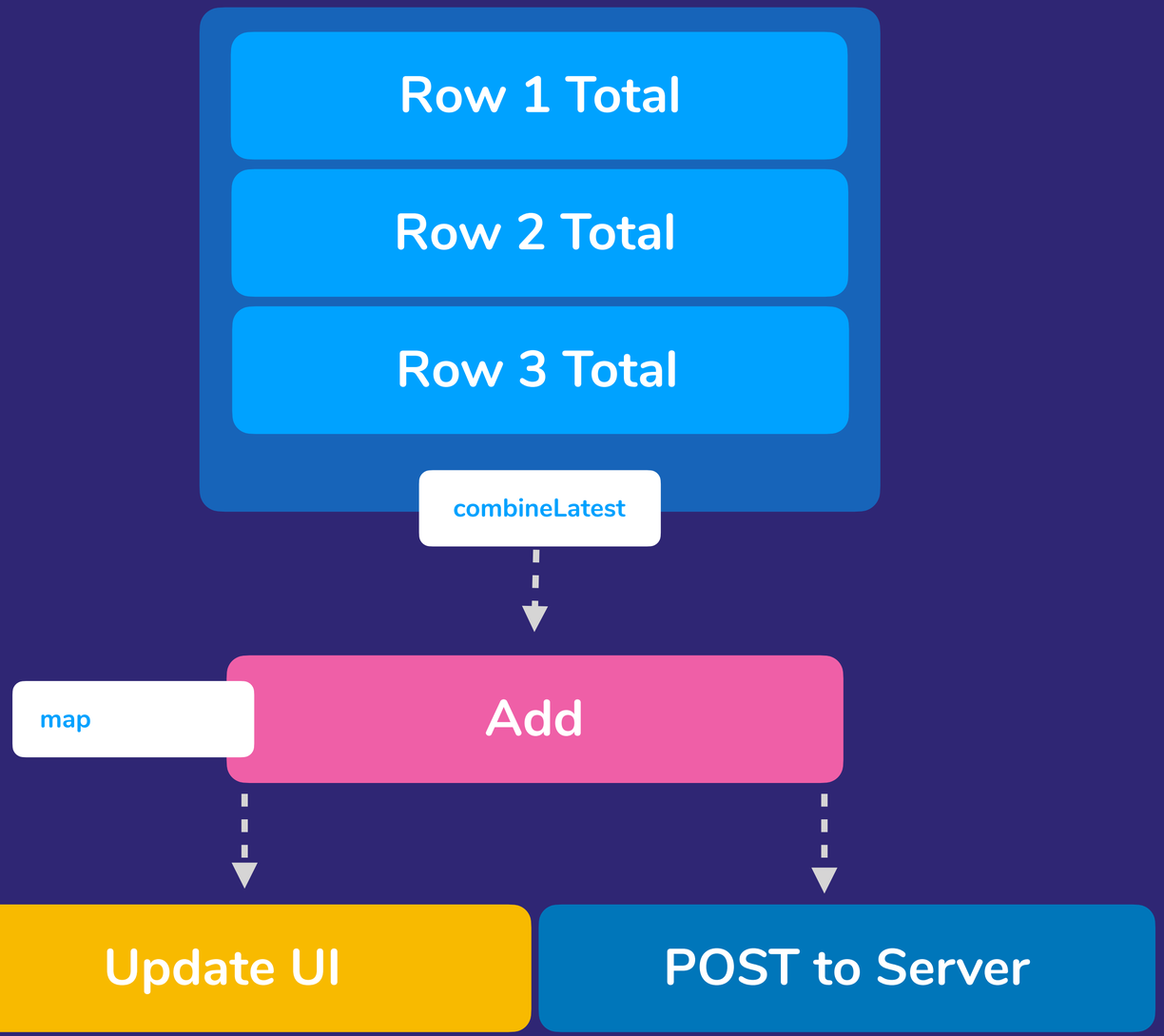
Totali fattura



Totali fattura



Totali fattura



Cosa ci guadagniamo?

- Codice molto parlante (se scritto bene)
- Codice *predictable* e *testabile*
- Riutilizzo del codice
- Gestione del tempo: una passeggiata
- Gestione errori: una passeggiata
- Gestione parallelismo: una passeggiata
- Separazione delle responsabilità con Observable e funzioni ad hoc
- Spesso, performance addirittura migliori
- Una riga molto importante in più sul curriculum

Cose di cui non abbiamo parlato:

- Concorrenza
- Operatori avanzati
- Ciclo di vita
- Subject
- Multicasting, Observable Hot
- Scheduler
- Testing

Michele Stieven

www.accademia.dev

Accademia DEV
WEB DEVELOPER ACADEMY

github.com/UserGalileo/talks

www.accademia.dev

Accademia DEV
WEB DEVELOPER ACADEMY