



Wyższa Szkoła Ekonomii i Informatyki w Krakowie

Programowanie reaktywne - RxJS

RxJS - operatory standalone i operatory pipeable pracujące na wielu strumieniach

- `race`, `raceWith` - zwraca pierwszy ze strumieni który zrobił next | err | complete
- `zip`, `zipWith` - sekwencyjnie i jednocześnie emituj wartości z każdego źródła w postaci tablicy
- `combineLatest`, `combineLatestWith` - dla każdej emisji z dowolnego źródła wyemituj komplet ostatnich wartości
- `forkJoin` - wyemituj ostatnie wartości gdy wszystkie observables się zakończą
- `merge` - złącz emisje w jeden strumień
- `concat`, `concatWith` - po zakończeniu pierwszego observabla emituj drugi, następnie kolejny itd
- `switchMap`, `mergeMap`, `concatMap`, `exhaustMap` - dla każdej emisji A, zapisz się do B

RxJS - własne operatory

- Na podstawie pipe()

Najczęstszy case - mamy zestaw operatorów w pipe() który jest wspólny dla wielu observabli

Przykład: sumujIDodajEtykieta

- Od zera. Tylko po co:)

Funkcja musi implementować interfejs

(obs: Observable<T>) => Observable

Należy pamiętać o pełnej implementacji Observable (next, error, complete)

Higher order observable

- Higher order observable to taki observable którego wartościami są inne observable, np:

```
subject.next(of(„Jan”, „Nowak”, „Kraków”))  
subject.next(of(„Katarzyna”, „Kowalska”, „Gdańsk”))  
subject.next(of(„Magdalena”, „Wiśniewska”, „Wrocław”))
```

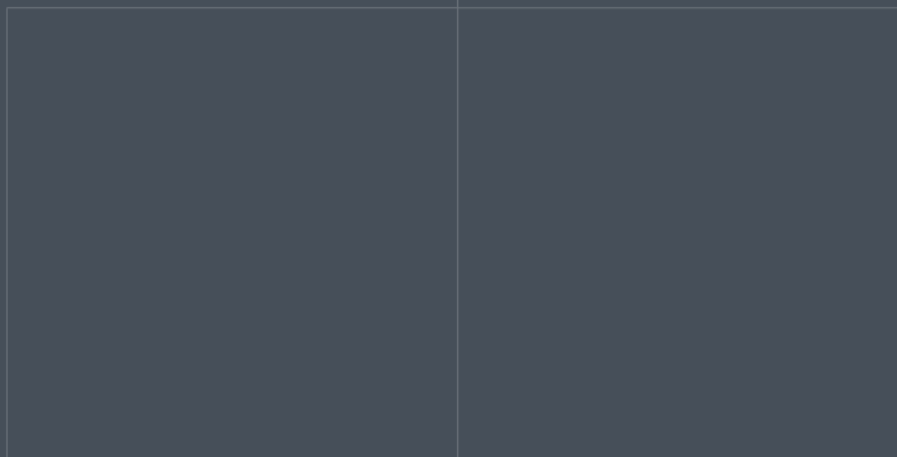
- operatory:
 - concatAll
 - combineLatestAll
 - switchAll
 - mergeAll
 - exhaustAll
 - zipAll

RxJS - schedulers

- Schedulers kontrolują:
 - kiedy i na jakich zasadach dostarczane są wartości ze strumienia
 - kiedy startuje subskrypcja
- Schedulers mogą zmieniać kontekst wykonania strumienia
- Rodzaje schedulers:
 - `null/queueScheduler` - scheduler synchroniczny, operacje rekursywne lub oparte o kolejkę
 - `asapScheduler` - scheduler asynchroniczny oparty na kolejce micro task - ta sama co Promise
 - `asyncScheduler` - scheduler asynchroniczny, strumień czasowy oparty na `setTimeout/setInterval`
 - `animationFrameScheduler` - zgodnie z rysowaniem ekranu

Testy jednostkowe

- Narzędzia: Jest, Testing library, Mocha, Jasmine (... i kilka innych)
- Problem: testowanie kodu reaktywnego w synchroniczny, deterministyczny sposób
- Problem: testowanie strumieni pracujących w dłuższym czasie (np. emisja co minutę)
- Rozwiązania:
 - konwersja do Promise
 - scheduler TestScheduler
 - Marbles - graficznie zobrazowanie działania strumienia
 - zewnętrzne biblioteki, np. [@hirezio/observer-spy](#)



Wyższa Szkoła Ekonomii
i Informatyki w Krakowie

