RxJS

1. курсы https://app.pluralsight.com/library/courses/rxjs-angular-reactive-development
2. В чем преимущество Observable для HttpClient в Angular.
    - их можно отменить
    - перехватчики
3. Promise vs Observable. Какая разница? В чем преимущество?
    - отменяемые
    - много потоков
    - готовые операторы
    - серия значений, а не только одно
4. Observable - наблюдаемый
    - lazy push collection
    - обратный вызов next -
    - обратный вызов error -
    - обратный вызов complete -
5. Subject
    - разновидность наблюдаемого, можно подписаться нескольким наблюдателям - multicast
    - asyncSubject
    - replaySubject - кэширование и повторение
    -
6. Subscription
    - отменяемый(disposable) через unsubscribe объект
7. BehaviourSubject что это, в чем отличие?
    - начальное значение
8. Как обработать ошибку в Observable?
    - pipe(catchError())
9. Как осуществить multicasting? Приведите пример.
    -
10. Higher order observable
    - Observable emits Observable

```
    of(1).pipe( // outer observable
        map(item=>of(item)) // inner observable
    ).subscribe(outer=>outer.subscribe(inner=>console.log(inner,
outer)))
```

- наблюдаемые высшего порядка уплощаются операторами concatMap, switchMap, MergeMap

11. Стратегии слияния/схлопывания flattening
    - Merge - слияние
        - без потерь
        - без сохранения очерёдности
        - без кэширования
        - без отписок
    - Switch - переключение
        - подходит для автодополнения, выбора из списка с подгрузкой значений
        - с потерями
        - с сохранением очерёдности
        - переподписывается(отписывается от старого) на новый поток
    - Concat - объединение
        - кэширует все новые потоки
        - переподписывается когда текущий поток завершён
        - без потерь
        - с сохранением очерёдности
        - для получения словарей по id
    - Exhaust - истощение
        - подходит для авторизации, исключения гонки асинхронных событий
        - игнорирует новые потоки пока не закончится текущий
        - с потерями
        - с сохранением очерёдности
    - share - делиться
        - подписывается на входящий поток, когда подписываются на него внешние подписчики
        - отписывается, если все отписались от него
        - делает поток горячим - новые подписчики получают значения только с момента своей подписки
        - возвращает subject
        - multicasting
        - для кэширования(shareReplay) и websocket

12. подходы к комбинированию потоков
    - just in time по требованию - mergeMap, switchMap=>(item)=>mergeMap(item)
    - get it all предзагрузка - combineLatest.pipe(filter())

13. лучше комбинировать все потоки в один для упрощения связывания кода в HTML

```
    <div *ngIf = combineLatest$ | async as model>
        <div>{{model.data1}}
            <div>{{model.data2}}
```

14. Функции создания потоков как combineLatest необходимо импортировать из rxjs, а не из rxjs/operators

15. сигнальные потоки Actions

    - Нельзя заменять в сигнальном потоке ошибки через EMPTY - поток может завершиться
    - необходимо сразу стартовать поток со значением, чтобы не потерять значения внутренних потоков - BehaviorSubject
    - сигнальный поток не завершается, потому некоторые операторы высшего порядка могут бесконечно ждать завершения сигнального потока.

16. Холодные потоки

    - обычно unicast
    - не стартуют, пока нет подписок
    - of(,,,), from([,,,])

17. Горячие потоки

    - обычно multicast
    - Subject/BehaviorSubject
    - стартуют без подписок