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
    - A Function is a lazily evaluated computation that synchronously returns a single value on invocation.
    - A generator is a lazily evaluated computation that synchronously returns zero to (potentially) infinite values on iteration.
    - A Promise is a computation that may (or may not) eventually return a single value.
    - An Observable is a lazily evaluated computation that can synchronously or asynchronously return zero to (potentially) infinite values from the time it's invoked onwards.
4. Observable - наблюдаемый
    - represents the idea of an invokable collection of future values or events.
    - lazy push collection
    - обратный вызов next -
    - обратный вызов error -
    - обратный вызов complete -
5. Observer - наблюдатель
    - is a collection of callbacks that knows how to listen to values delivered by the Observable
6. Subject
    - разновидность наблюдаемого, можно подписаться нескольким наблюдателям - multicast
    - is equivalent to an EventEmitter, and the only way of multicasting a value or event to multiple Observers
    - asyncSubject
    - replaySubject - кэширование и повторение
    - behaviorSubject - есть начальное значение

```
    const setFabricRunners$: Subject<{ id: number; task$:
Observable<any> }[]> = new Subject();
    const getFabricRunners$ = setFabricRunners$.asObservable().pipe();
    getFabricRunners$.subscribe();

    //
    const observable = new Observable(subscriber => {
        subscriber.next(1);
        subscriber.next(2);
        subscriber.next(3);
        setTimeout(() => {
            subscriber.next(4);
            subscriber.complete();
        }, 1000);
    });

    observable.subscribe({
        next(x) { console.log('got value ' + x); },
        error(err) { console.error('something wrong occurred: ' +
err); },
        complete() { console.log('done'); }
    });
```

7. Operators
    - are pure functions that enable a functional programming style of dealing with collections with operations like map, filter, concat, reduce, etc.
8. Subscription
    - экземпляр выполняемого Observable
    - отменяемый(disposable) через unsubscribe объект
9. Schedulers
    - are centralized dispatchers to control concurrency, allowing us to coordinate when computation happens on e.g. setTimeout or requestAnimationFrame or others.
    - pipe(ObserveOn(asapScheduler))
10. Как обработать ошибку в Observable?
    - pipe(catchError())
11. multicasting
    - горячий источник с переиспользованием(share) побочных эффектов(источников)
    - publish
    - multicast
    - share
    - shareReplay

12. 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

13. Стратегии слияния/схлопывания flattening
    - Merge - слияние
        - без потерь
        - без сохранения очерёдности(order)
        - без кэширования
        - без отписок
    - Concat - объединение
        - без потерь
        - с сохранением очерёдности
        - кэширует все новые потоки
        - переподписывается когда текущий поток завершён
        - для получения словарей по id
    - Switch - переключение
        - с потерями
        - с сохранением очерёдности
        - подходит для автодополнения, выбора из списка с подгрузкой значений
        - переподписывается(отписывается от старого) на новый поток
    - Exhaust - истощение
        - с потерями
        - с сохранением очерёдности
        - подходит для авторизации, исключения гонки асинхронных событий
        - игнорирует новые потоки пока не закончится текущий
    - 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

14. подходы к комбинированию потоков

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

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

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

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

17. 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.

18. Холодные потоки

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

19. Горячие потоки

    - обычно multicast
    - Subject/BehaviorSubject
    - стартуют без подписок
  ```