**Observable:** Data producer. Each subscription causes independent execution of the observable.

**Subject:** Data producer, data consumer. Can be subscribed to. Can subscribe to other observables. All subscribers to a subject share the data it produces.

**BehaviorSubject:**

**zip:** *(you go, I go).* Merge results from each observable together.

**concat:** Subscribe to the observables in order. When the last observable completes then emit the values as you got them. Used when order is important, for instance in-order http requests.

**forkJoin:** *(final destination)* Don’t let trigger the subscription until after all observables are complete. Then pass the values as an array.

**mergeMap:** Only when the inner observable emits, let me know by merging the value to the outer Observable.

**pairwise:** Let me know when the Observable emits, but also give me the previous value.

**switchMap:** When source emits cancel any previous subscriptions of the inner observables.

**combineLatest:** *(going Dutch)*. When I go, I’ll take your values too. Let me know when *any* observable emits but also give me the latest value from the others.

**withLatetsFrom:** *(master slave operator)*. When master emits it grabs the last slave.

**take, takeUntil, first**: Complete the observables for you.

const firstColor$ = color$.pipe(take(1));

const firstLogo$ = logo$.pipe(first());

forkJoin(firstColor$, firstLogo$)

.subscribe(([color, logo]) => console.log(`${color} shirt with ${logo}`));