Value Transformation

David Mann

@MannD | Labs.HeirloomSoftware.com

Operators

- concatMap
- concatMapTo
- defaultIfEmpty
- endWith
- startWith
- exhaustMap

- expand
- map
- mapTo
- scan
- mergeScan
- pluck

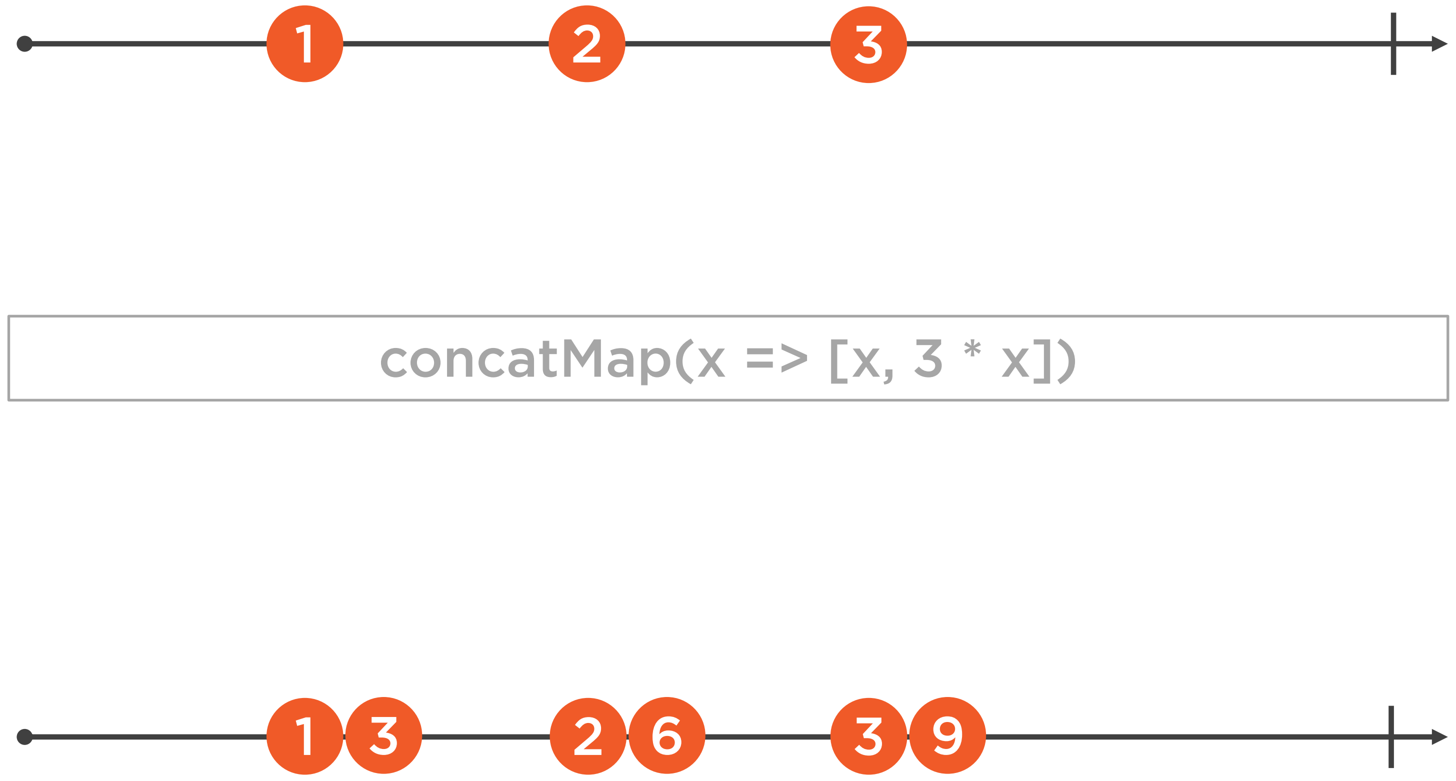
- reduce
- switchMap/flatMap
- mergeMapTo
- switchMapTo
- materialize
- dematerialize

concatMap

Map values to provided function/Observable

Emits all values

Sequential

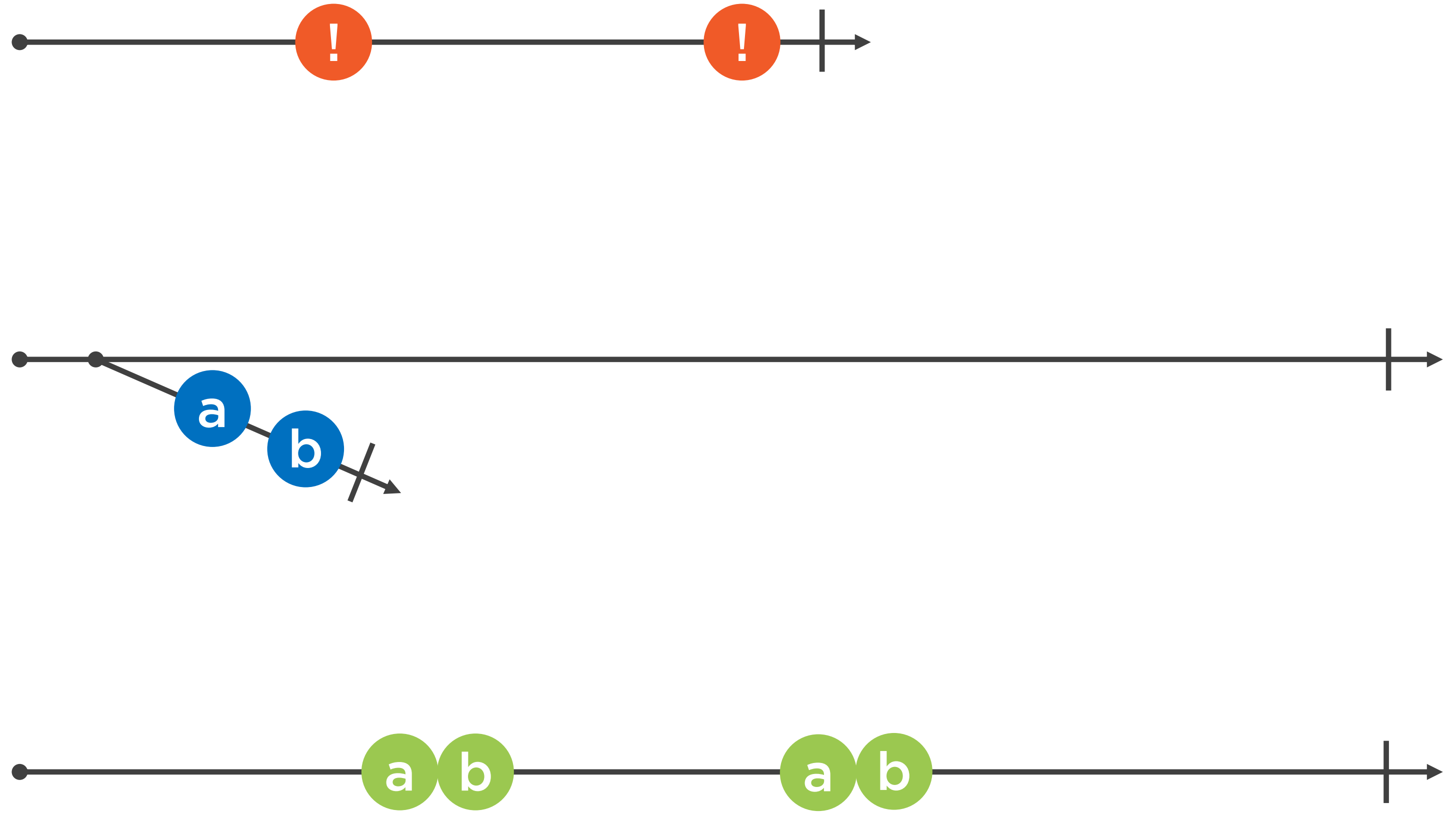


concatMapTo

Source as signal

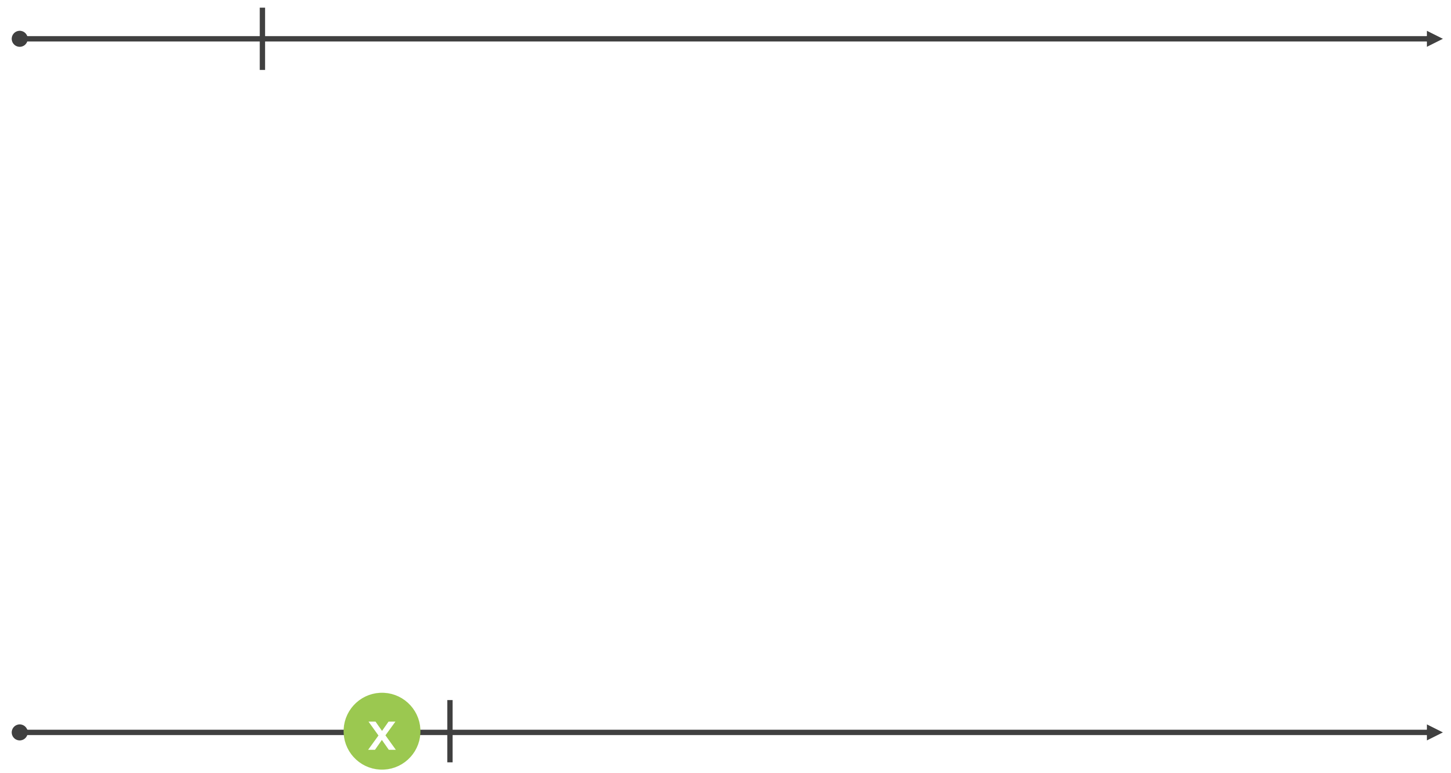
Subscribe when notified

Emit combined results



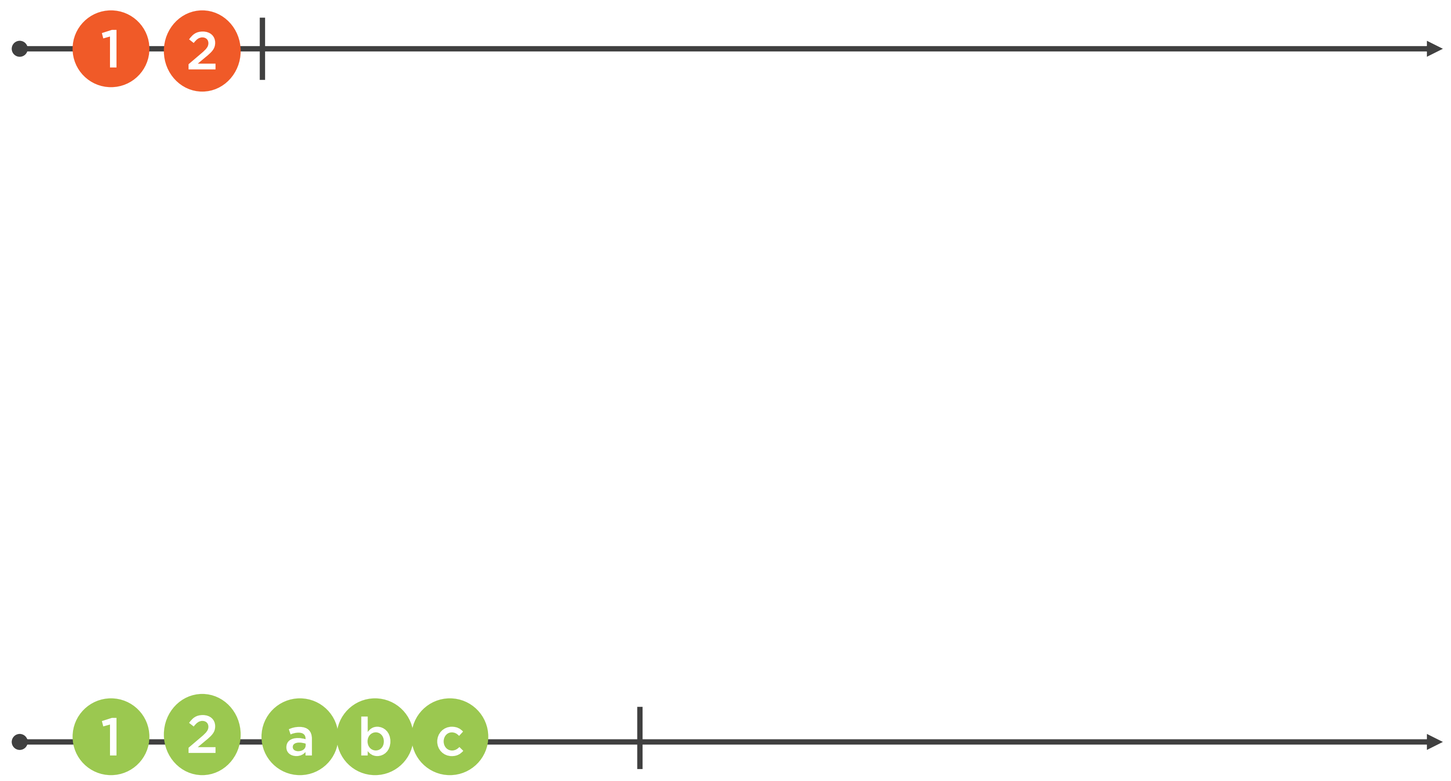
defaultIfEmpty

Default Value



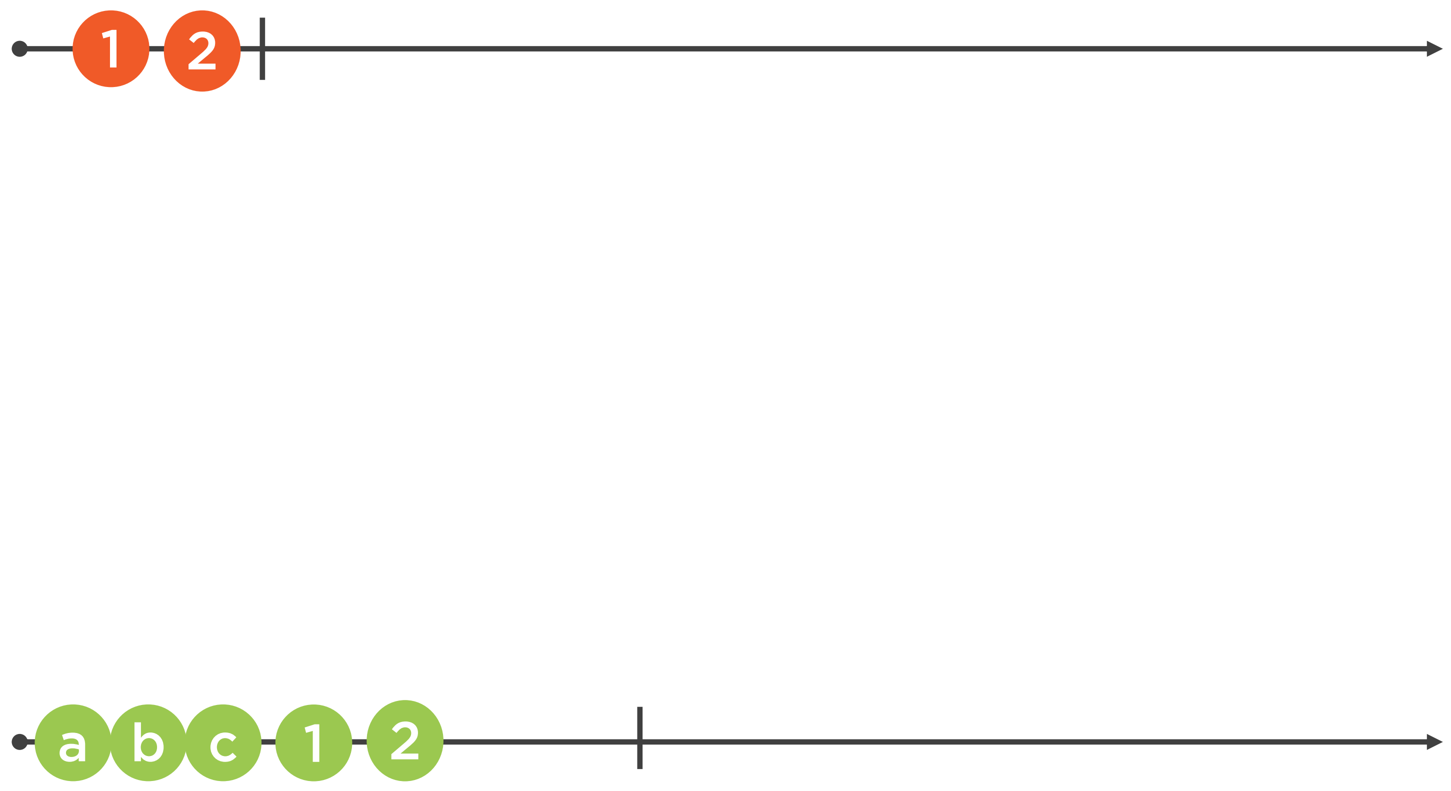
endWith

Append values



startWith

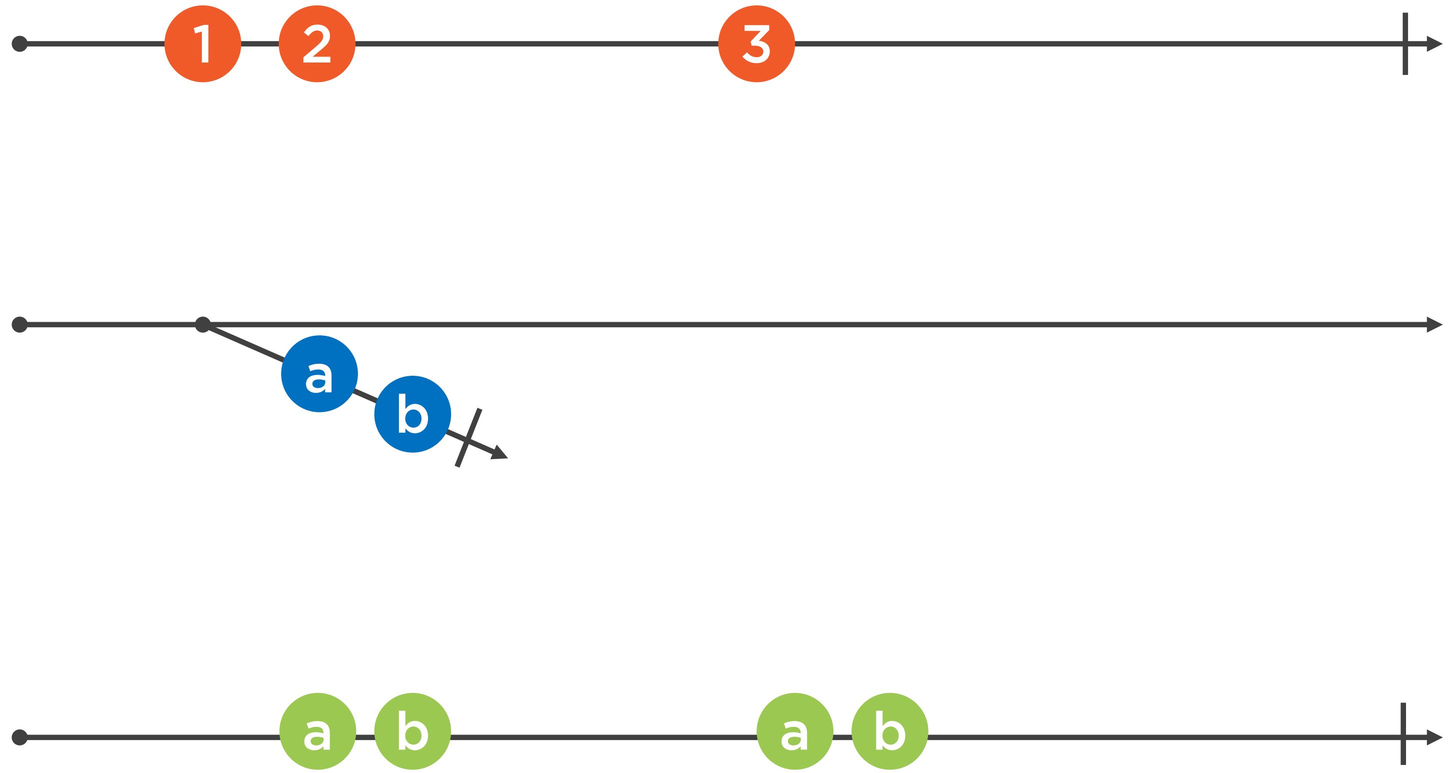
Prepend values



exhaustMap

Map values to generated Observables

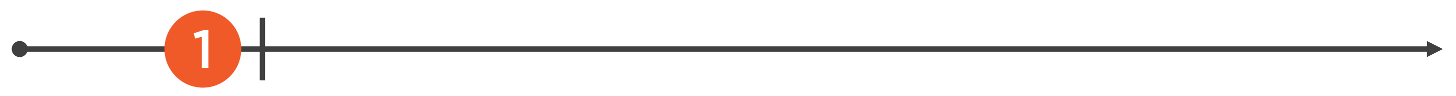
Emit results



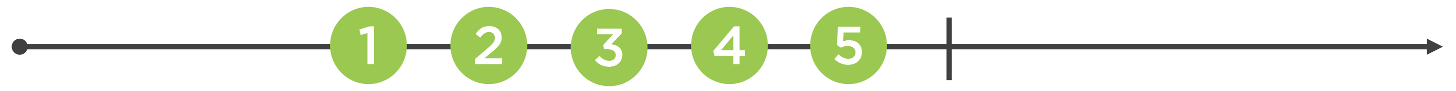
expand

Apply function source values

recursive

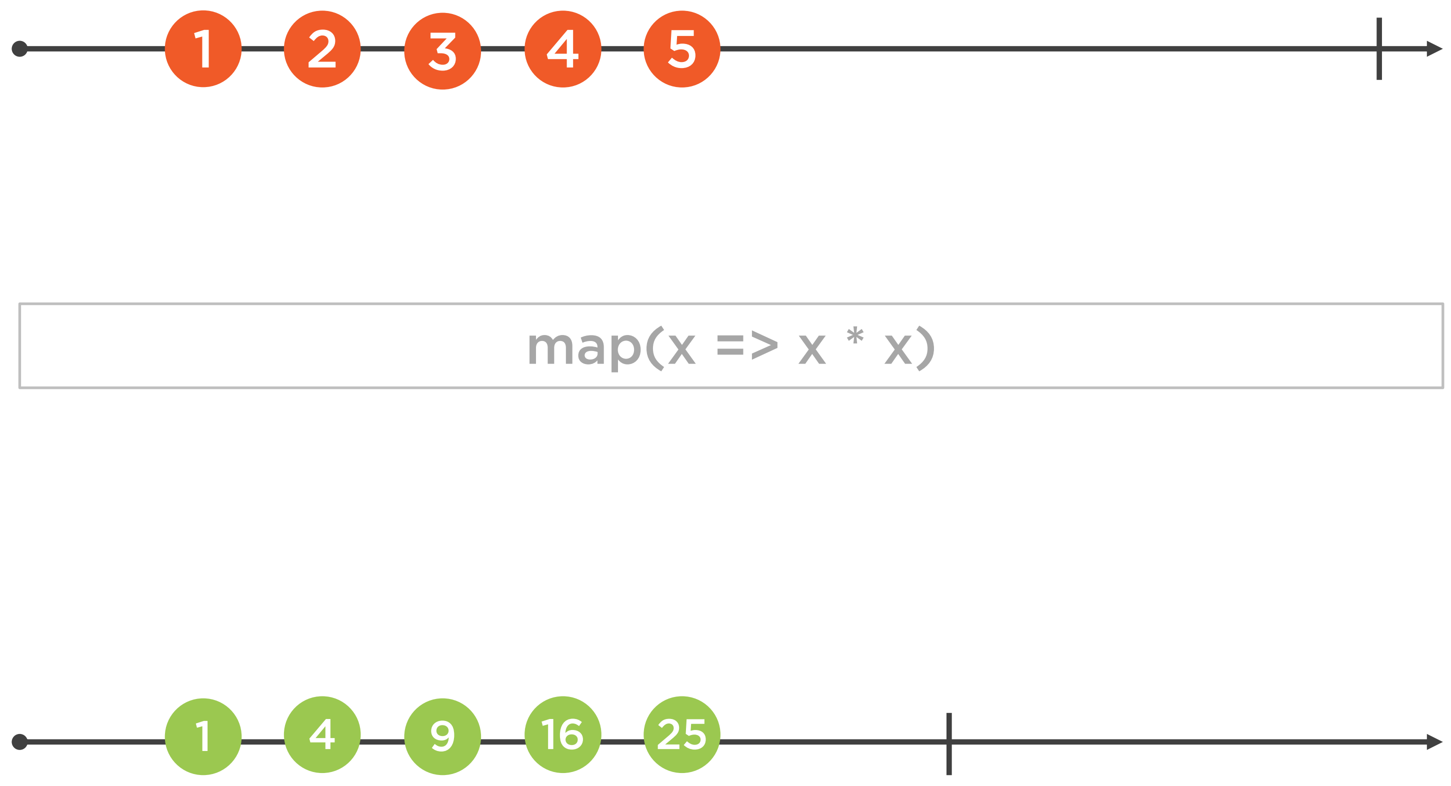


```
expand(x => {return x < 5 ? of(x + 1) : empty() ;})
```



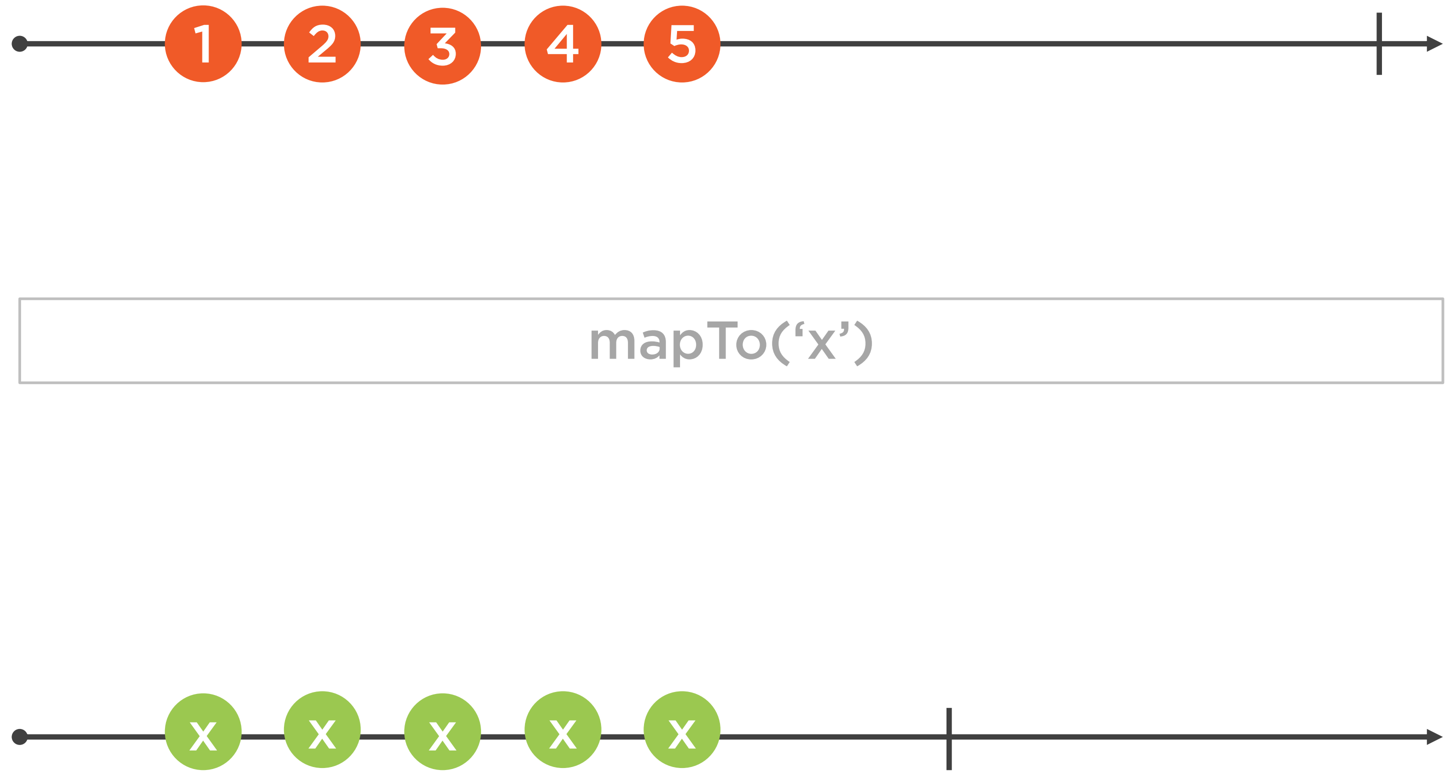
map

Pass all values through a function



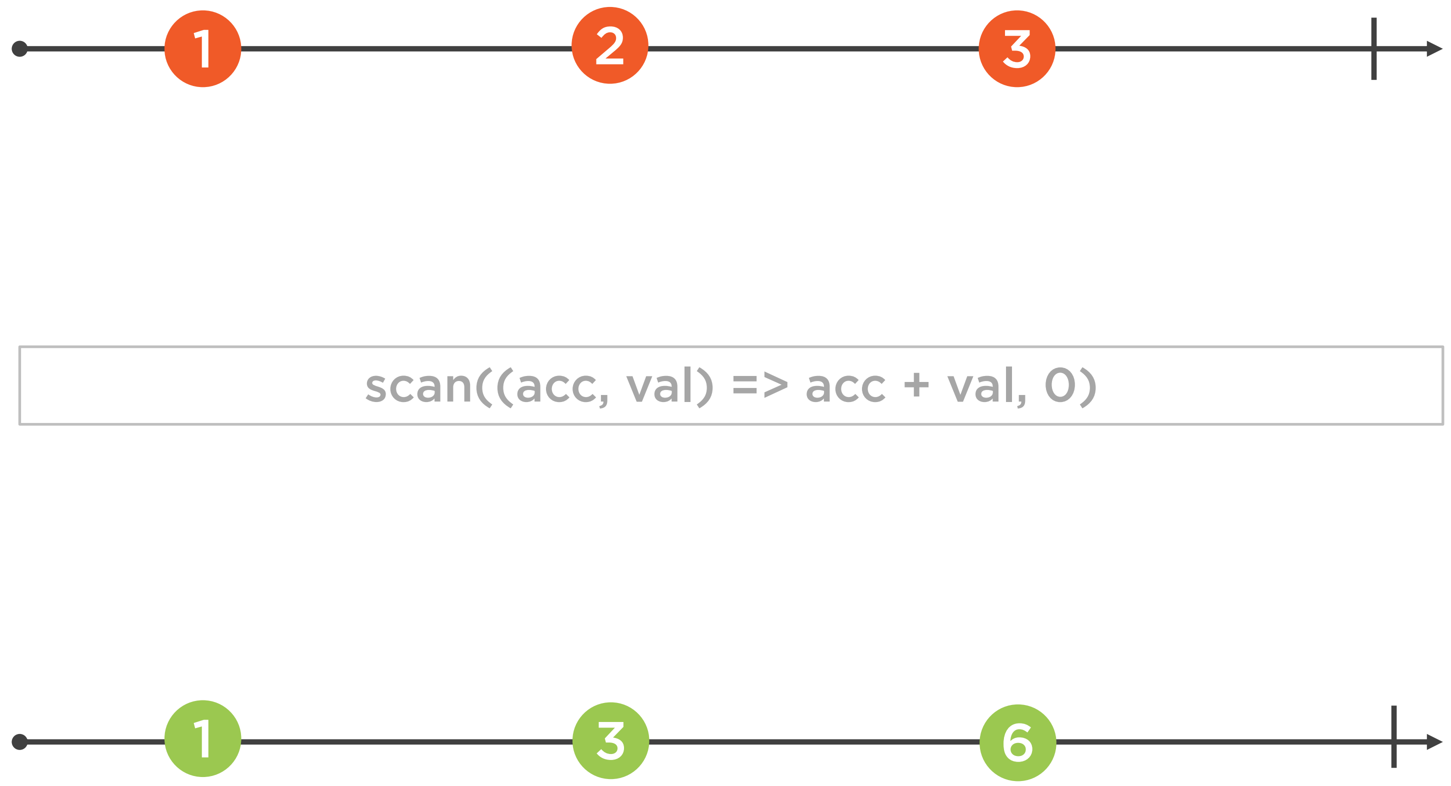
mapTo

Convert every value to same constant



scan

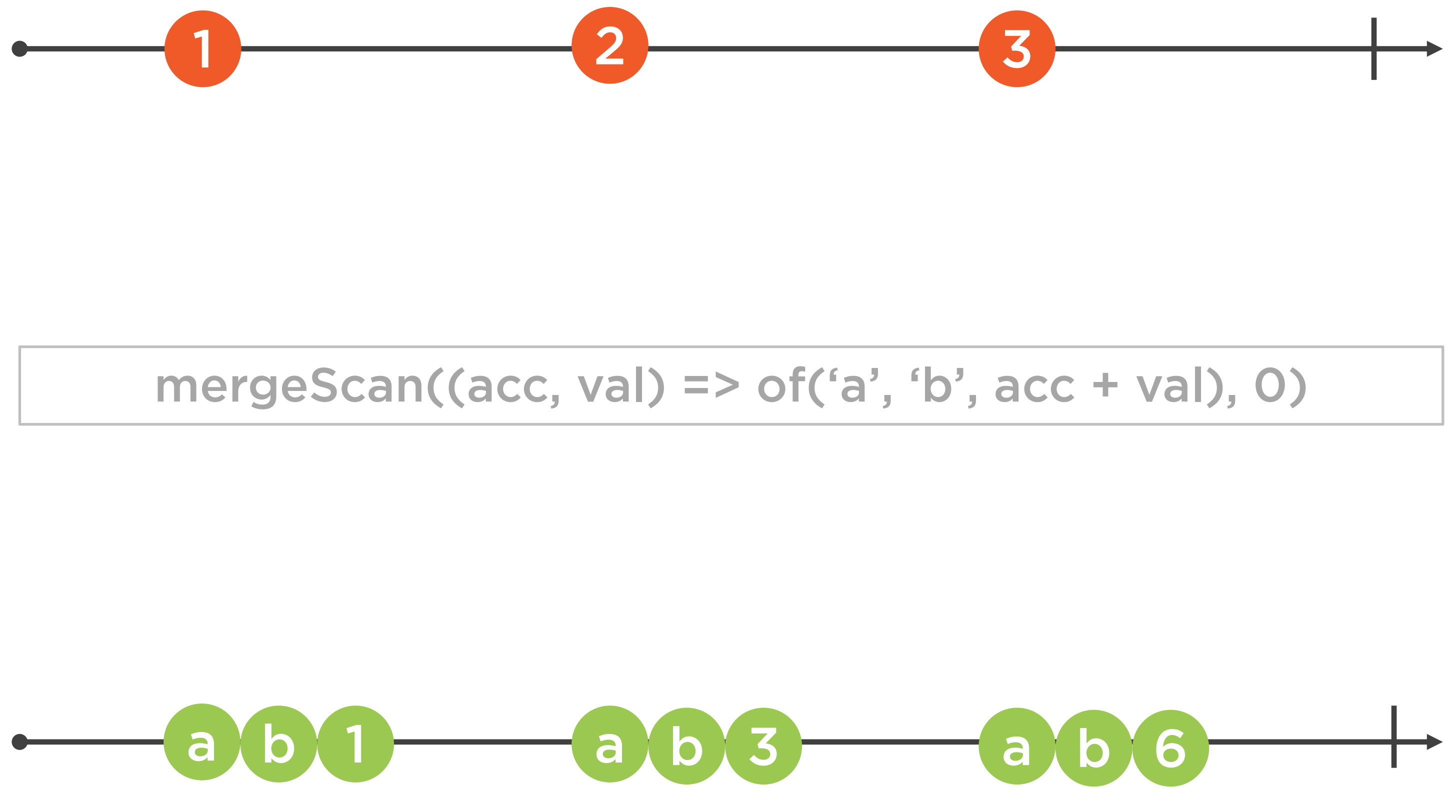
Emit cumulative results



mergeScan

Emit cumulative results

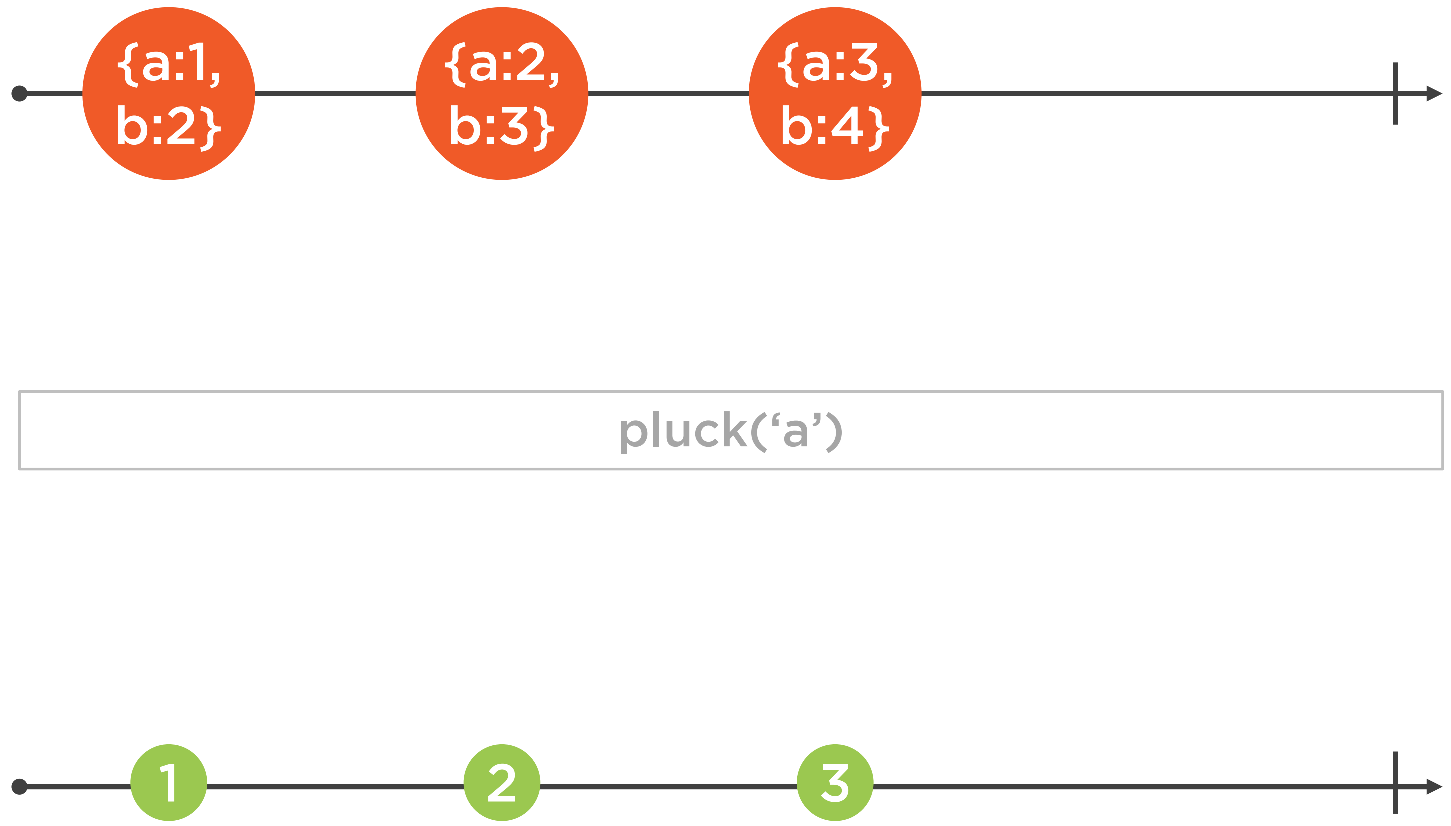
Plus other values



mergeScan((acc, val) => of('a', 'b', acc + val), 0)

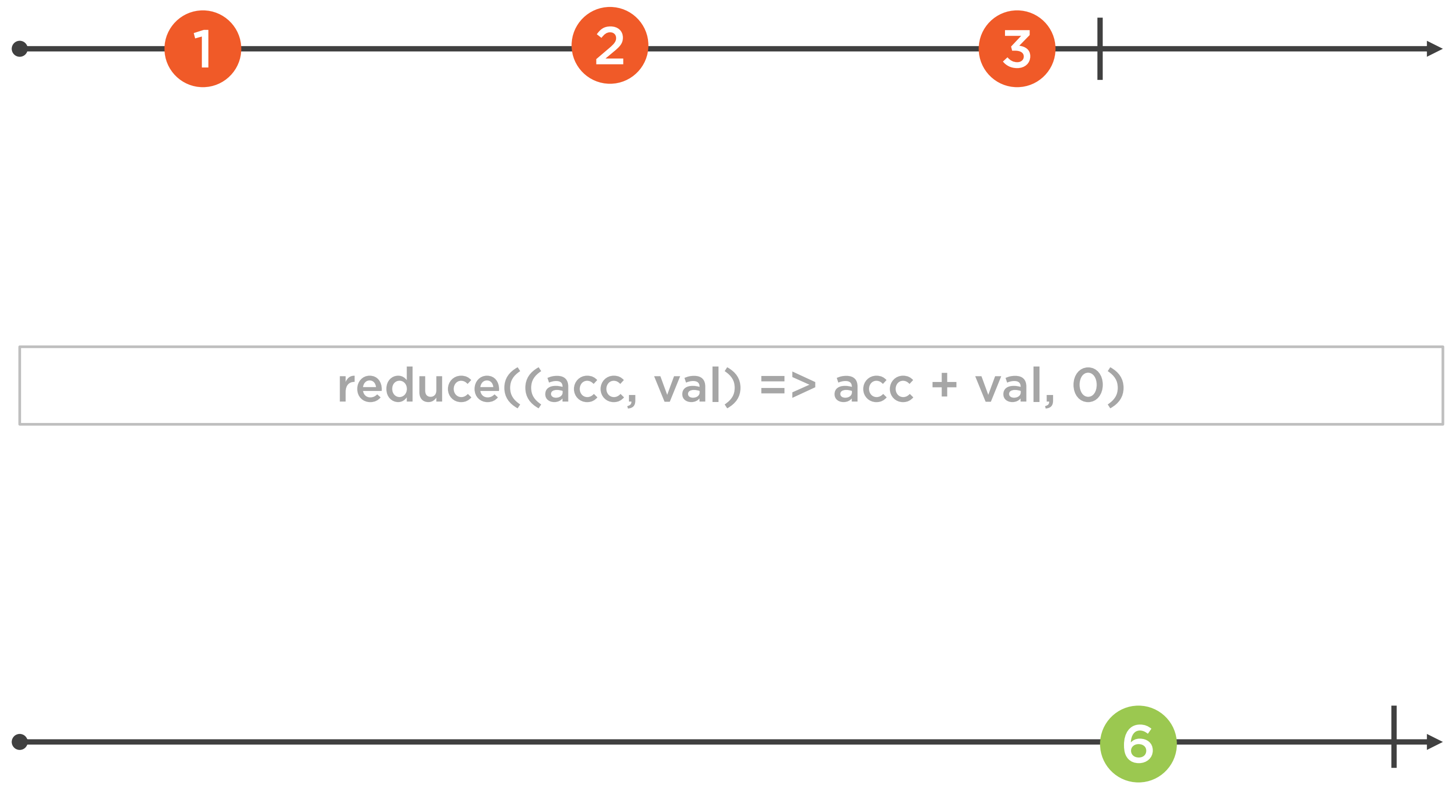
pluck

Pull values from objects



reduce

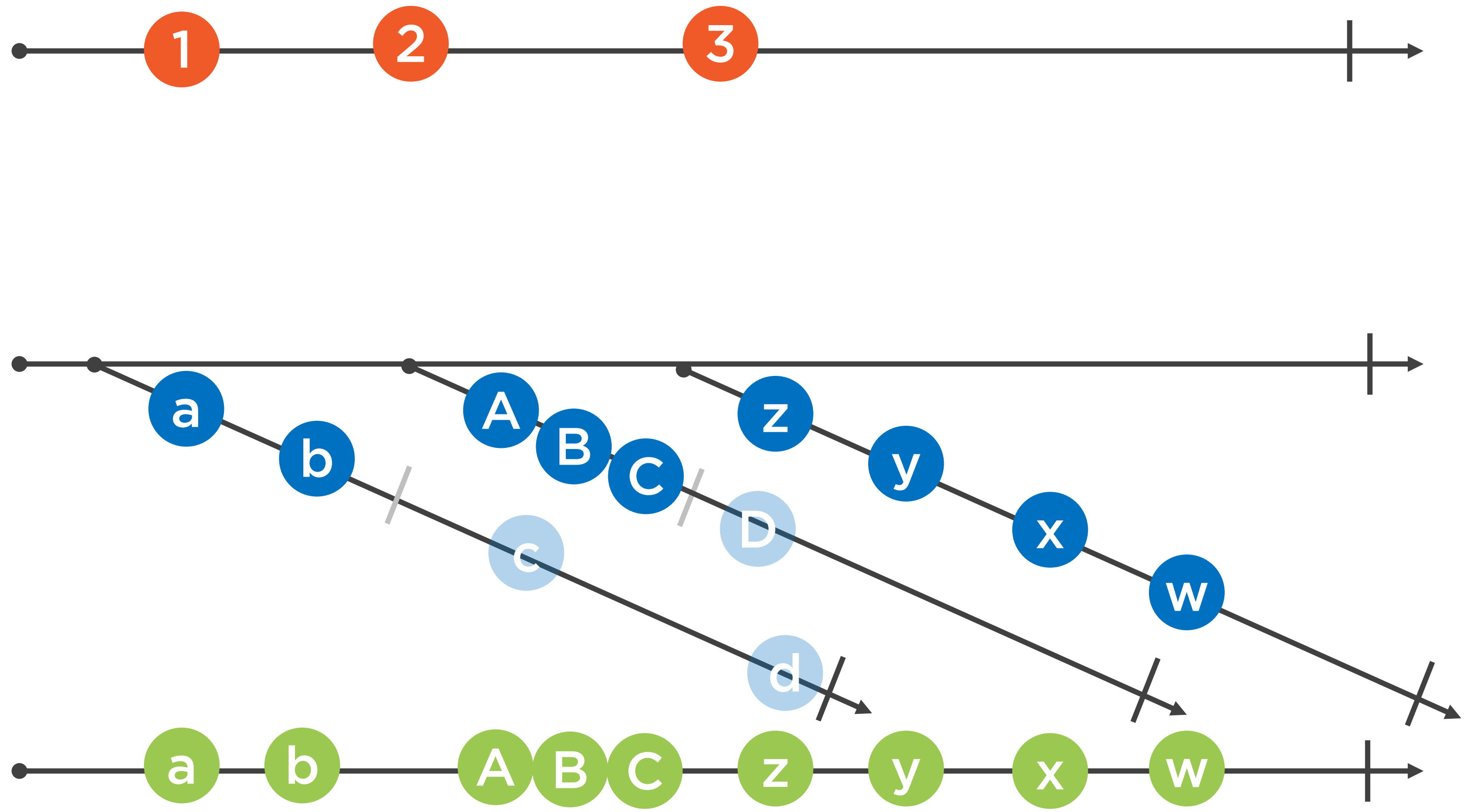
Emit final cumulative result



switchMap/flatMap

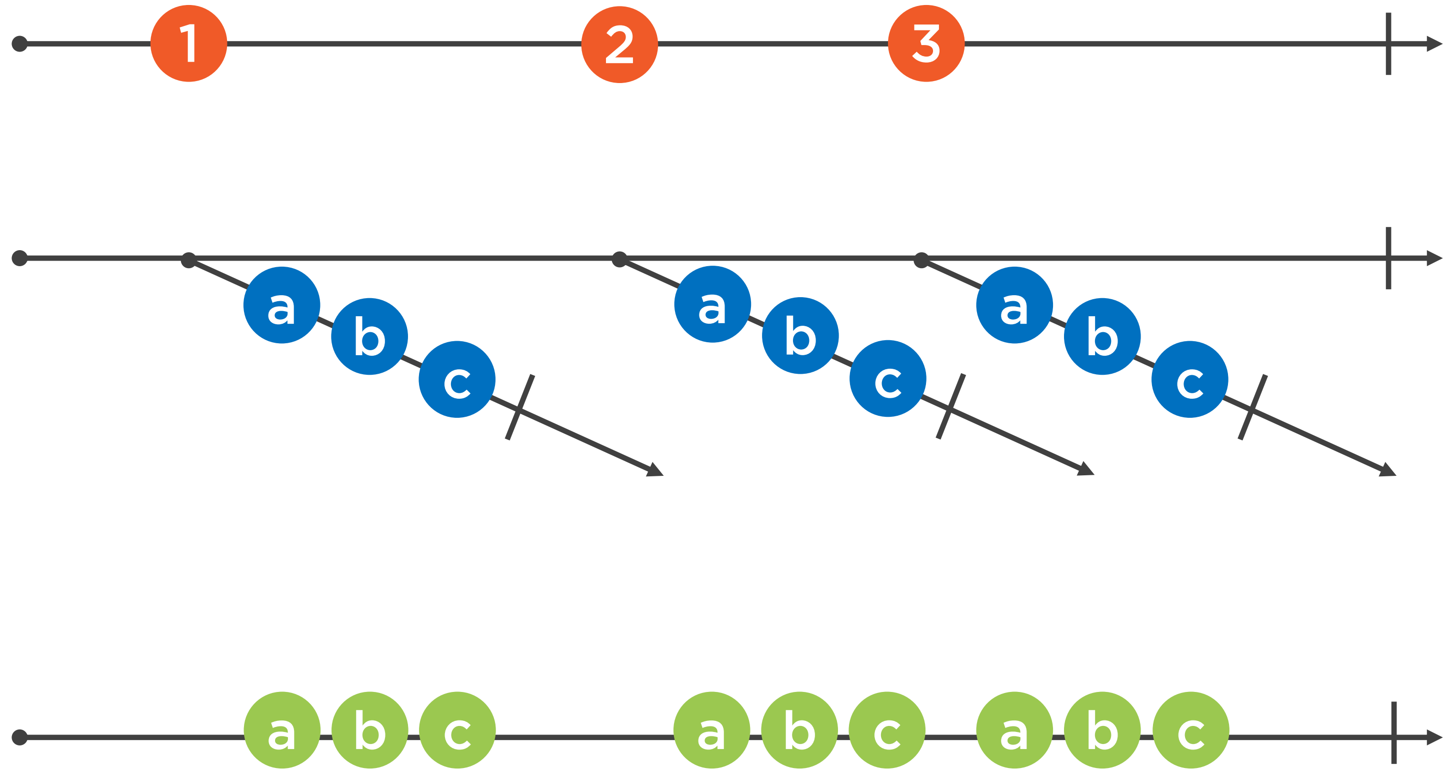
Subscribe to internal Observable when source emits

Cancel previous subscription



mergeMapTo

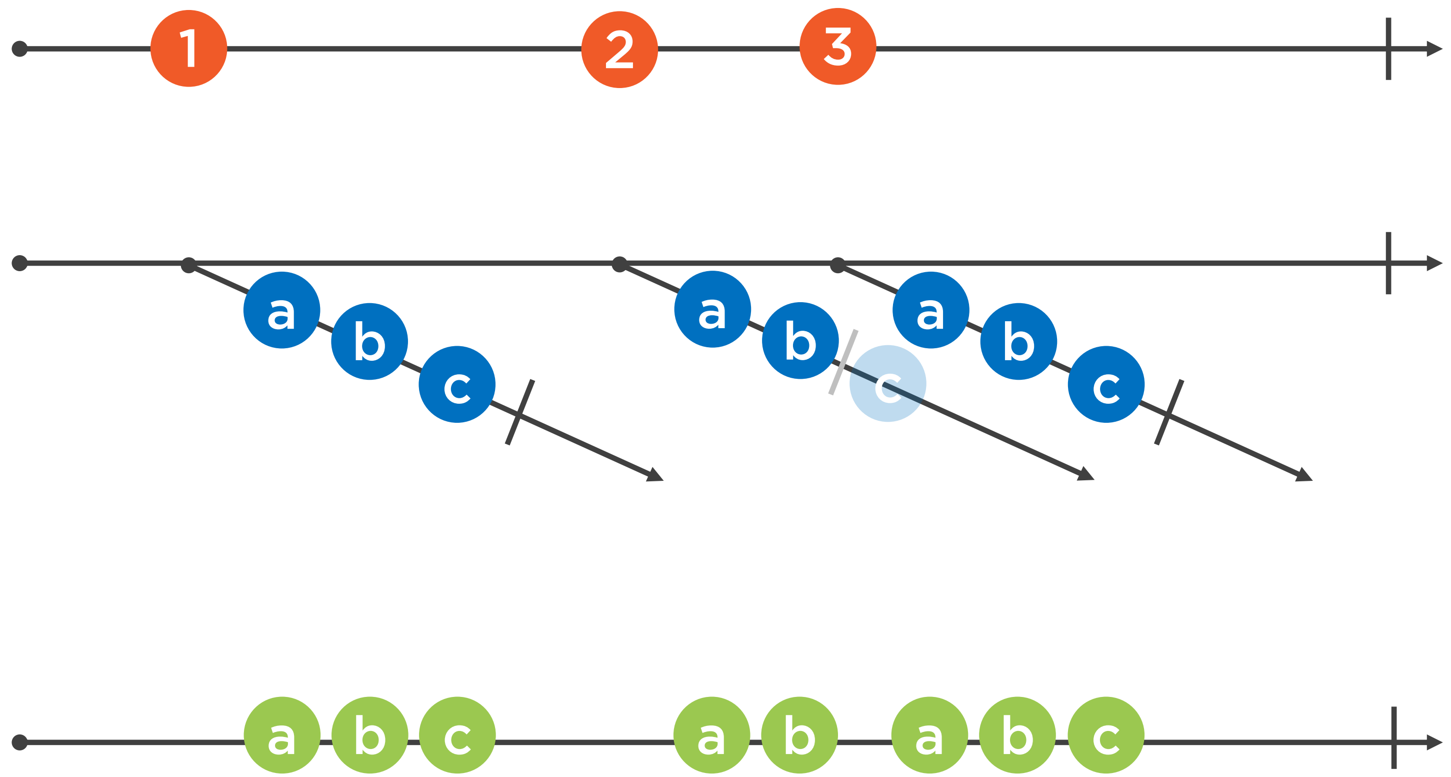
Emit inner Observable for each source value



switchMapTo

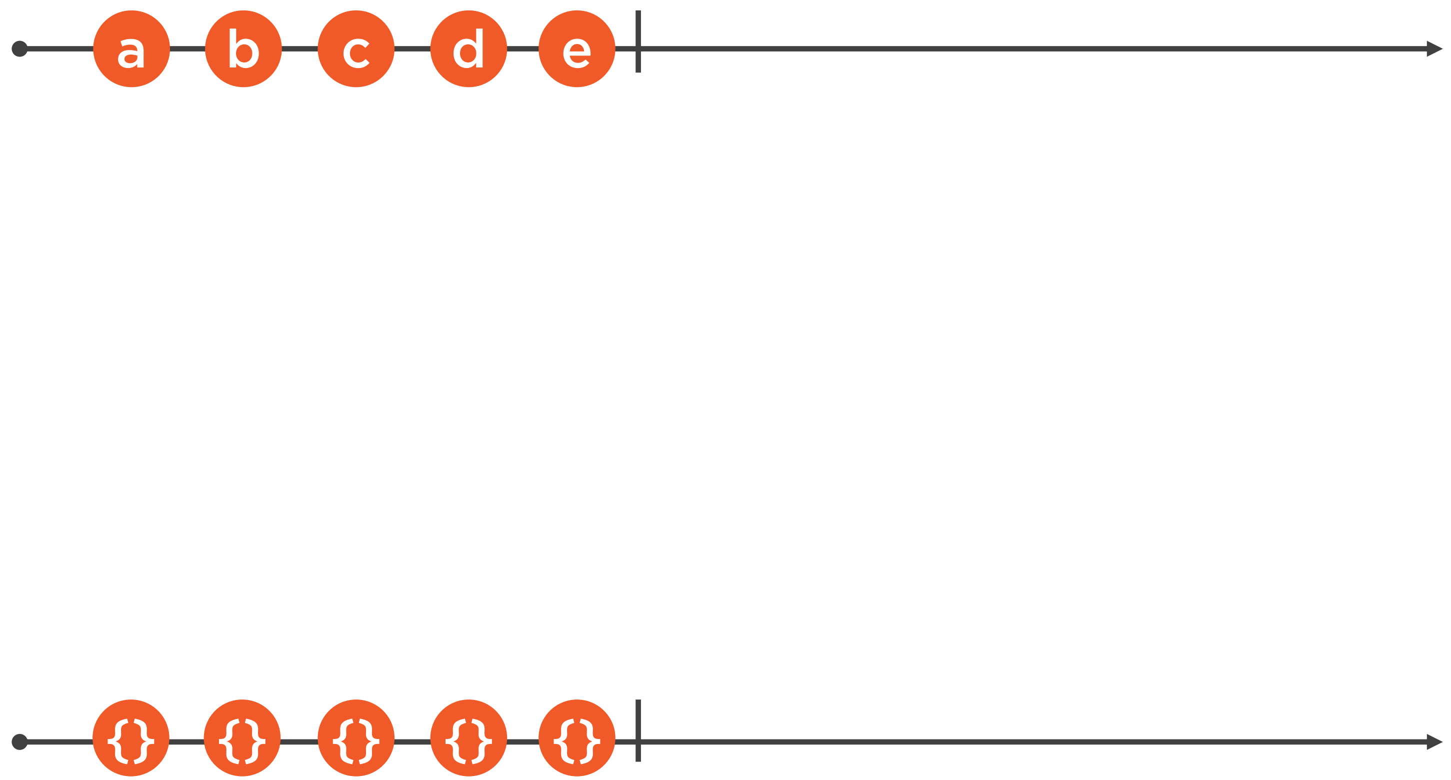
Emit inner Observable for each source value

Cancel previous subscription



materialize

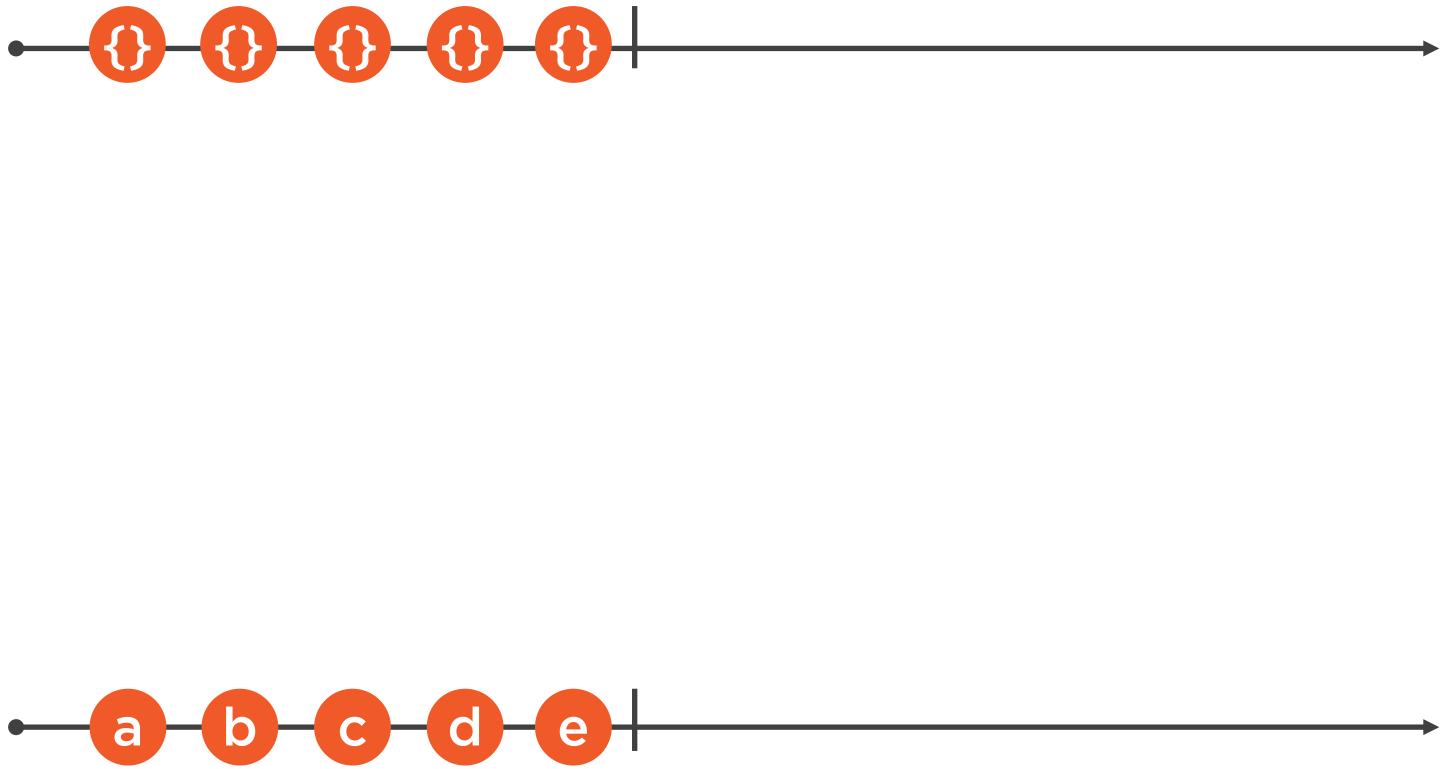
Replace emissions with Notifications



{ kind: 'N', value: *<value>*, error: undefined, hasValue: true }
{ kind: 'C', value: undefined, error: undefined, hasValue:false}
{ kind: ‘E', value: undefined, error: *<error contents>*, hasValue:false}

dematerialize

Replace Notifications with standard emissions



{ kind: 'N', value: *<value>*, error: undefined, hasValue: true }
{ kind: 'C', value: undefined, error: undefined, hasValue:false}
{ kind: ‘E', value: undefined, error: *<error contents>*, hasValue:false}