# Products

this.filteredProducts$ = **this.store.pipe(select(selectProducts));**

# Filtering

<div class="bar">

**<div>Search: <input type="text" (input)="filter($event)" /></div>**

**<!-- <div>**

**Sort:**

**<select (change)="sort($event)">**

**<option value=""></option>**

**<option value="price">Price</option>**

**<option value="name">Name</option>**

**</select>**

**</div> -->**

</div>

const products$ = this.store.pipe(select(selectProducts));

**this.filteredProducts$ = combineLatest([**

**products$,**

**this.filterSubject.pipe(startWith('')),**

**]).pipe(**

**map(([products, filter]) => {**

**const filteredProducts = products.filter((product) =>**

**product.name.toLowerCase().includes(filter.toLowerCase())**

**);**

**return filteredProducts;**

**})**

**);**

**2B - Debounce**

this.filterSubject.pipe(**debounceTime(500)**, startWith('')),

# Price

**const prices$ = this.store.pipe(select(selectPrices));**

const products$ = this.store.pipe(select(selectProducts**)).pipe(**

**withLatestFrom(prices$),**

**map(([products, prices]) => {**

**return products.map((product) => ({**

**...product,**

**price: prices[product.id],**

**}));**

**})**

**);**

# Sorting

<div class="bar">

<div>Search: <input type="text" (input)="filter($event)" /></div>

**<div>**

**Sort:**

**<select (change)="sort($event)">**

**<option value=""></option>**

**<option value="price">Price</option>**

**<option value="name">Name</option>**

**</select>**

**</div>**

</div>

this.filteredProducts$ = combineLatest([

products$,

this.filterSubject.pipe(debounceTime(300), startWith('')),

**this.sortSubject.pipe(startWith('')),**

]).pipe(

map(([products, filter**, sortCriteria**]) => {

const filteredProducts = products.filter((product) =>

product.name.toLowerCase().includes(filter.toLowerCase())

);

**filteredProducts.sort((a, b) => {**

**if (sortCriteria === 'price') {**

**return a.price < b.price ? -1 : 1;**

**}**

**if (sortCriteria === 'name') {**

**return this.collator.compare(a.name, b.name);**

**}**

**return 0;**

**});**

return filteredProducts;

})

);

# Price delta

**const priceDelta$ = prices$.pipe(**

**pairwise(),**

**map(([oldPrices, newPrices]) =>**

**Object.keys(newPrices).reduce((aggregation, key) => {**

**const newPrice = newPrices[key];**

**const oldPrice = oldPrices[key];**

**if (oldPrice !== undefined && newPrice !== undefined) {**

**aggregation[key] = roundTwoDecimals(newPrice - oldPrice);**

**}**

**return aggregation;**

**}, {} as { [id: string]: number })**

**)**

**);**

const products$ = this.store.pipe(select(selectProducts)).pipe(

withLatestFrom(prices$, **priceDelta$**),

map(([products, prices, **priceDelta**]) => {

return products.map((product) => ({

...product,

price: prices[product.id],

**priceDelta: priceDelta[product.id],**

}));

})

);