**Create a Search Pipe to Dynamically Filter results with Angular 4**

**Background**

**Angular JS (version 1.x) — Filters**

In Angular JS there were *Filters*which allowed us to format and transform data. Angular JS came with a few built in Filters , such as:

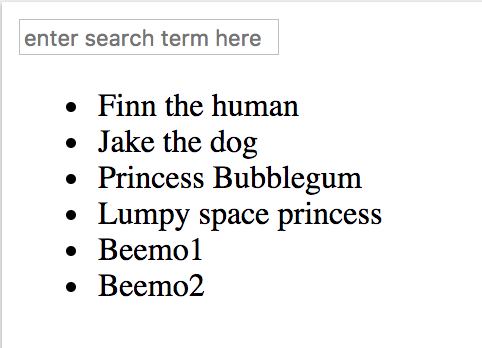
* uppercase — converts “Hello” to “HELLO”
* lowercase — converts “Hello” to “hello”
* currency — converts 123 to $123.00

A common built in Filter was called filter which took an array as input and returned a subset of that array based on the term we supplied the filter.

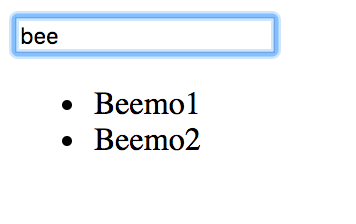
A very common use case of this is to have an input box where a user enters a search text and the results are filtered appropriately. Here’s some example code which does that:

*app.html the list of characters we display in the ul is declared in the charactersCtrl as $scope.characters = ['Finn the human', 'Jake the dog', etc...]*<div ng-app="myApp" ng-controller="charactersCtrl"><input ng-model="searchText" placeholder="enter search term here">  
 <ul>  
 <li ng-repeat="c in characters | filter : searchText">  
 {{ c }}  
 </li>  
 </ul></div>

The code above users the filter Filter to return a subset of our characters array.



Here is the entire unfiltered list



After the user enters a search term, a subset of the list where the character name matches that search term is returned

**Angular (version 2/4) — Pipes**

A few things have changed in Angular. First of all, tools that allow us to transform data are no longer called *Filters*they are now called *Pipes*.

Angular comes with a few built in Pipes, many of them are the same as the ones that came with Angular JS.

However Angular did not come with the order by or filter pipes that were supplied in Angular JS. The reason for this can be found [here](https://angular.io/guide/pipes#no-filter-pipe). To summarize, these pipes are expensive because they need to observe state changes and execute the filter function whenever a change occurs. For instance, the filter pipe must always keep track of any changes to the search text value so it can return the appropriate sub array to the template view. This can lead to a laggy user experience.

However, if we still want to filter some results with a search bar, we can create our own filter pipe. I’ll show you how to do that here with Angular 4.

**Implementation**

**Set up the project**

I’ll be using the Angular 4 quickstart repo as my project. You should add/modify the files as you see fit according to your project structure. You can get the quickstart repo set up by running the following commands:

$ git clone https://github.com/angular/quickstart.git quickstart   
$ cd quickstart   
$ npm install   
$ npm start

To make the structure easier to follow, I’m going to separate the template code from the component by adding a file called app.component.html. I’ll also add a file to hold our filter code called filter.pipe.ts . The file structure inside the app folder should look like this now:

app  
|-- app.component.html  
|-- app.component.ts  
|-- app.module.ts  
|-- filter.pipe.ts

**Create some list data in the component and display it in the template**

Since we broke out the template code into app.component.html, we’ll have to remove the template attribute in the Component decorator of app.component.ts and replace it with the following line: templateUrl: './app/component.html.

Let’s also add a list of character names to display in the template. We’ll filter these characters with our Filter Pipe later.

Your app.component.ts file should now look like this:

import { Component } from '[@angular/core](http://twitter.com/angular/core)';[@Component](http://twitter.com/Component)({  
 selector: 'my-app',  
 templateUrl: './app.component.html'  
})  
export class AppComponent {  
 name = 'Angular';characters = [  
 'Finn the human',  
 'Jake the dog',  
 'Princess bubblegum',  
 'Lumpy Space Princess',  
 'Beemo1',  
 'Beemo2'  
 ]}

Now lets modify app.component.html to display the characters in a list. Add this to the html file:

<ul>  
 <li \*ngFor="let c of characters">  
 {{c}}  
 </li>  
</ul>

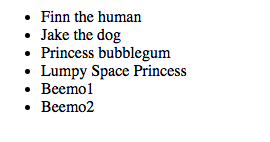
\*ngFor is Angular’s version of Angular JS’s ngRepeat. In order to use this, we’ll need to import the **Forms Module**into the app module. To do this add the following import statement at the top of app.module.ts: import { FormsModule } from '@angular/forms';.

We also need to import it in the NgModule decorator, so add it to the array of imports, after the BrowserModule.

Your app.module.ts file should now look like this:

import { NgModule } from '[@angular/core](http://twitter.com/angular/core)';  
import { FormsModule } from '[@angular/forms](http://twitter.com/angular/forms)';  
import { BrowserModule } from '[@angular/platform-browser](http://twitter.com/angular/platform-browser)';import { AppComponent } from './app.component';[@NgModule](http://twitter.com/NgModule)({  
 imports: [ BrowserModule, FormsModule ],  
 declarations: [ AppComponent],  
 bootstrap: [ AppComponent ]  
})  
export class AppModule { }

Look at the app in your [browser](http://localhost:3000/). It should look like this:



Just a List of character names

**Create the Filter Pipe**

Let’s populate the pipe with code for the filter. Copy and paste this code into filter.pipe.ts:

import { Pipe, PipeTransform } from '[@angular/core](http://twitter.com/angular/core)';[@Pipe](http://twitter.com/Pipe)({  
 name: 'filter'  
})export class FilterPipe implements PipeTransform {  
 transform(items: any[], searchText: string): any[] {  
 if(!items) return [];  
 if(!searchText) return items;searchText = searchText.toLowerCase();return items.filter( it => {  
 return it.toLowerCase().includes(searchText);  
 });  
 }  
}

This code will return a subset of an array of items if any item contains the searchText string.

**Use the Filter Pipe**

In order to use the pipe, first we need to import it into the app module. Below the import AppComponent statement, import the Filter Pipe: import { FilterPipe }from './filter.pipe';. Also add it to the array of declarations in the NgModule decorator.

Your app.module.ts file should now look like this:

import { NgModule } from '[@angular/core](http://twitter.com/angular/core)';  
import { FormsModule } from '[@angular/forms](http://twitter.com/angular/forms)';  
import { BrowserModule } from '[@angular/platform-browser](http://twitter.com/angular/platform-browser)';import { AppComponent } from './app.component';  
import { FilterPipe} from './filter.pipe';[@NgModule](http://twitter.com/NgModule)({  
 imports: [ BrowserModule, FormsModule ],  
 declarations: [ AppComponent, FilterPipe ],  
 bootstrap: [ AppComponent ]  
})  
export class AppModule { }

Now you can use the filter pipe in your App Component. Let’s modify app.component.html so that we have an input box where we can put our searchText and let’s modify the list so that it makes use of our filter.

Your app.component.html should now look like this:

<input [(ngModel)]="searchText" placeholder="search text goes here">  
<ul>  
 <li \*ngFor="let c of characters | filter : searchText">  
 {{c}}  
 </li>  
</ul>

Now look at your app. You should see an input box.

Type some text into the box and see that your list being dynamically filtered!

