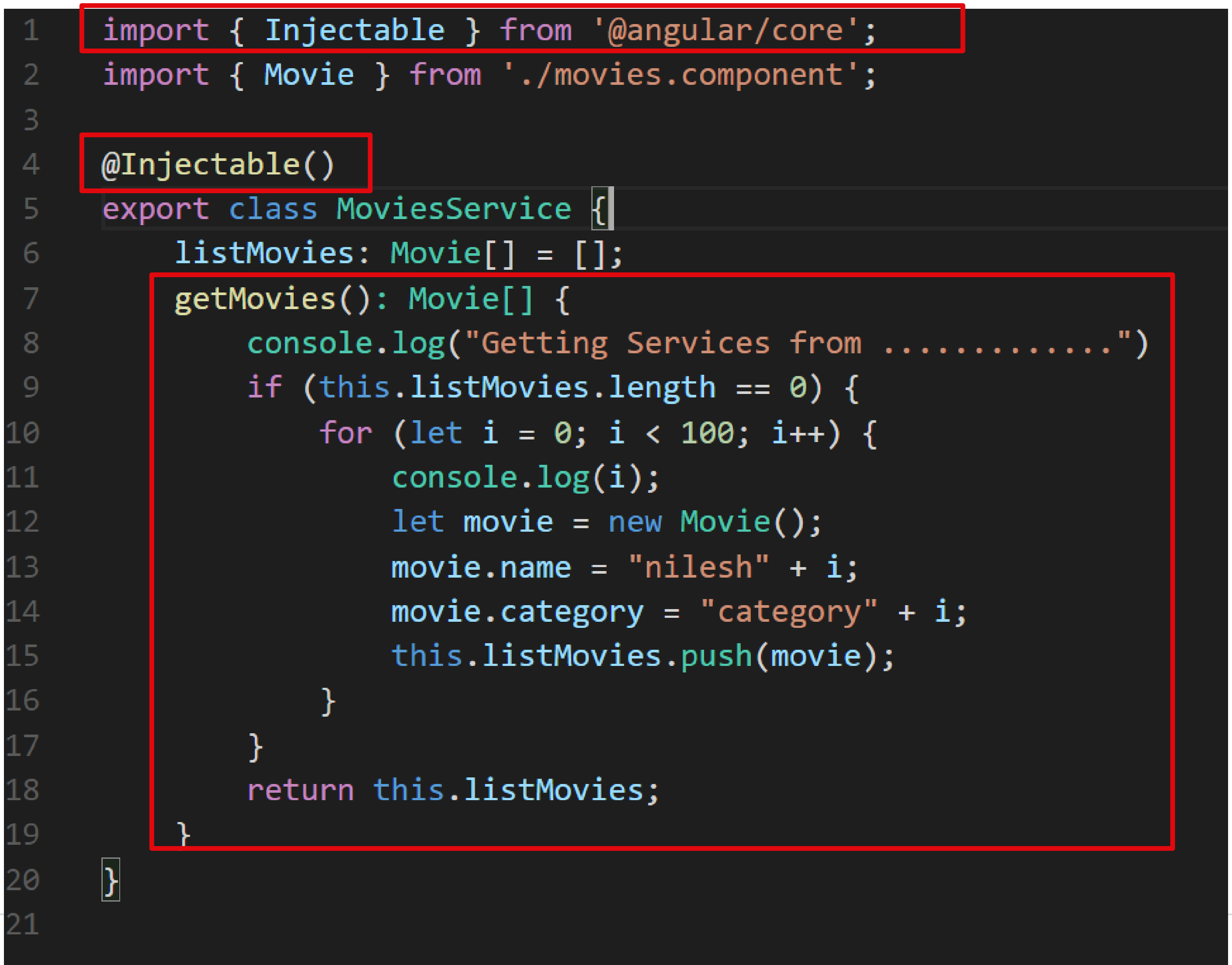
Services

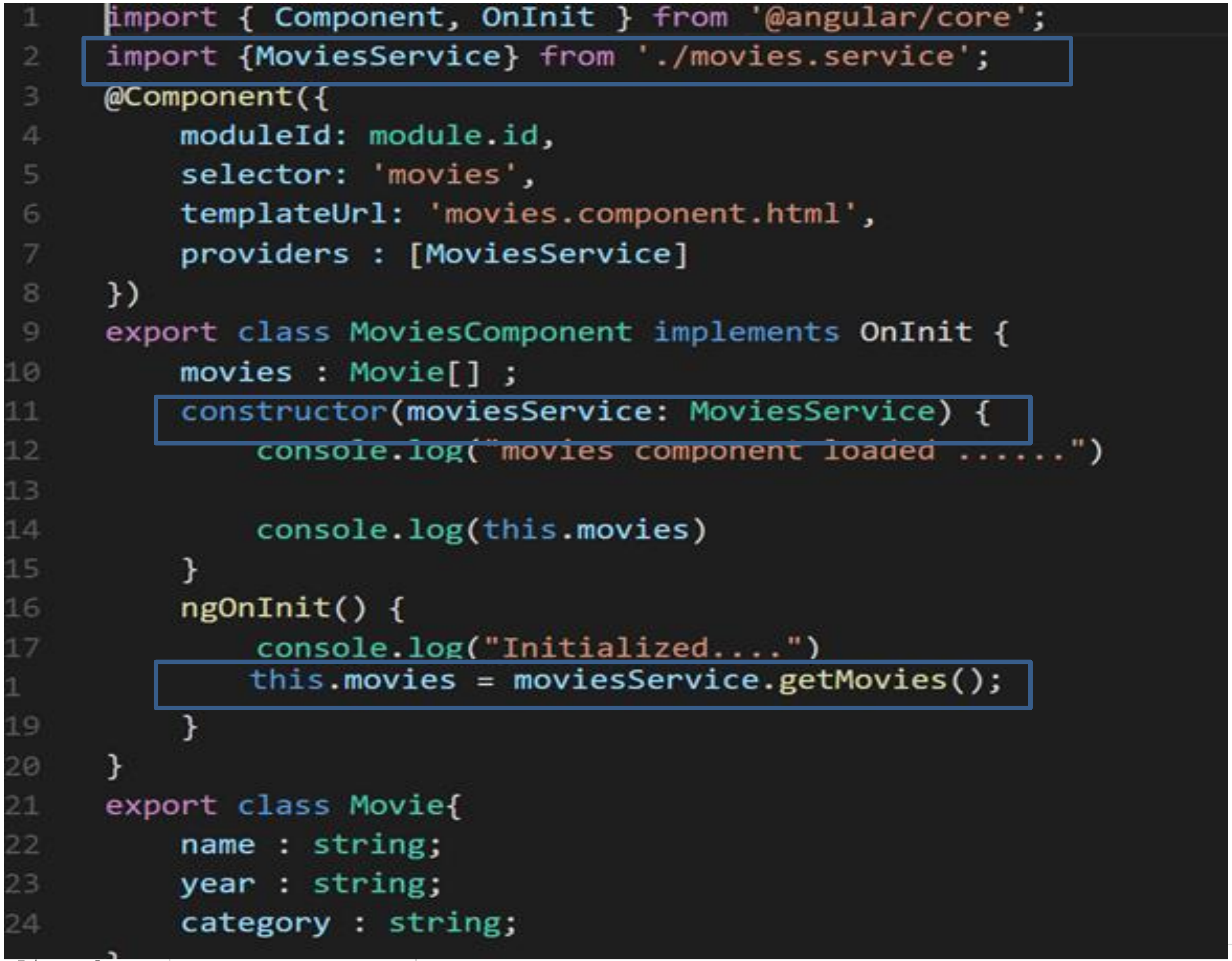
Developing a Service



```
import { Injectable } from '@angular/core';
import { Movie } from './movies.component';

@Injectable()
export class MoviesService {
    listMovies: Movie[] = [];
    getMovies(): Movie[] {
        console.log("Getting Services from .............")
        if (this.listMovies.length == 0) {
            for (let i = 0; i < 100; i++) {
                console.log(i);
                let movie = new Movie();
                movie.name = "nilesh" + i;
                movie.category = "category" + i;
                this.listMovies.push(movie);
            }
        }
        return this.listMovies;
    }
}
```

Using Services in Component



```
import { Component, OnInit } from '@angular/core';
import {MoviesService} from './movies.service';
@Component({
    moduleId: module.id,
    selector: 'movies',
    templateUrl: 'movies.component.html',
    providers : [MoviesService]
})
export class MoviesComponent implements OnInit {
    movies : Movie[] ;
    constructor(moviesService: MoviesService) {
        console.log("movies component loaded ......")

        console.log(this.movies)
    }
    ngOnInit() {
        console.log("Initialized....")
        this.movies = moviesService.getMovies();
    }
}
export class Movie{
    name : string;
    year : string;
    category : string;
```

Using Pipes

Built in Pipes

- **In-Built Pipes**
  - DatePipe
    - <p>Released on {{ movie.release| date:"MM/dd/yy" }} </p>
  - UpperCasePipe
    - <p>movie Name{{movie.name | uppercase}}
  - LowerCasePipe
    - <p>movie Name{{movie.name | lowercase}}
  - CurrencyPipe
    - <p>Movie Revenue{{movie.revenue | currency:‘USD'}}
  - PercentPipe
    - <p> movie.votes {{movie.popular | percent}}
    - <p> movie.votes {{movie.popular | percent:’4.3-5}}

Pipes

- {{ name |uppercase }} ---
- {{ name |lowercase }}
- {{ name|slice:’2’:’4’ }} ------ excludes index 4
- {{ name | replace:’the’:’hello’ }}

- {{ 8.567: number:1.2-3}}   -----before decimal minimum one number
-   ------after decimal min 2 numbers or maximum 3 numbers
- {{ 8.567|number:2.2-2}}   ---o/p is 08.57 it will round the number because max
- --------- digits after decimal is 2
-             {{8.567| currency : ‘Euro’ }}   -------o/p will be in Euro
-
- {{8.567| currency : ‘USD’ }}   USD8.567
- {{8.567| currency : ‘USD’:true }}   ------$8.567 :true indicates show the symbol
- {{8.567| currency : ‘GBP’:true }}   -----great Brittan pounds
- {{ server_date|date:”fullDate”}

Custom pipes

- **Builtin Pipes**
  - Dates
  - Currency
  - percentage
  - character cases
- **but you can also easily define custom pipes**
- **example**
  - create a pipe that takes a string and reverses the string.

Steps

1. **Create class that implement** PipeTransform interface
2. Add transform method in the class
3. Decorate class with @Pipe assign some name to it.
4. Add custom pipe as a declaration in your app module:

Step 1

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'reverseStr'})
export class ReverseStr implements PipeTransform {
  transform(value: string): string {
    let newStr: string = "";
    for (var i = value.length - 1; i >= 0; i--) {
      newStr += value.charAt(i);
    }
    return newStr;
  }
}
```

App module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';
import { ReverseStr } from './reverse-str.pipe.ts';

@NgModule({
  declarations: [
    AppComponent,
    ReverseStr
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Pipe with arguments

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'filesize' })
export class FileSizePipe implements PipeTransform {
  transform(size: number, extension: string = 'MB') {
    return (size / (1024 * 1024)).toFixed(2) + extension;
  }
}
```

- **To use it in template**
  - {{ file.size | filesize:'megabyte’ }}
- **file.size value will be passed to size argument and ‘megabyte’ will be passed to extension argument**

Pipe to filter the items in the list

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'filter'
})
export class FilterPipe implements PipeTransform {

  transform(items: any[], searchText: string, fieldName: string): any[] {

    // return empty array if array is falsy
    if (!items) { return []; }

    // return the original array if search text is empty
    if (!searchText) { return items; }

    // convert the searchText to lower case
    searchText = searchText.toLowerCase();

    // retrun the filtered array
    return items.filter(item => {
      if (item && item[fieldName]) {
        return item[fieldName].toLowerCase().includes(searchText);
      }
      return false;
    });
   }
}
```

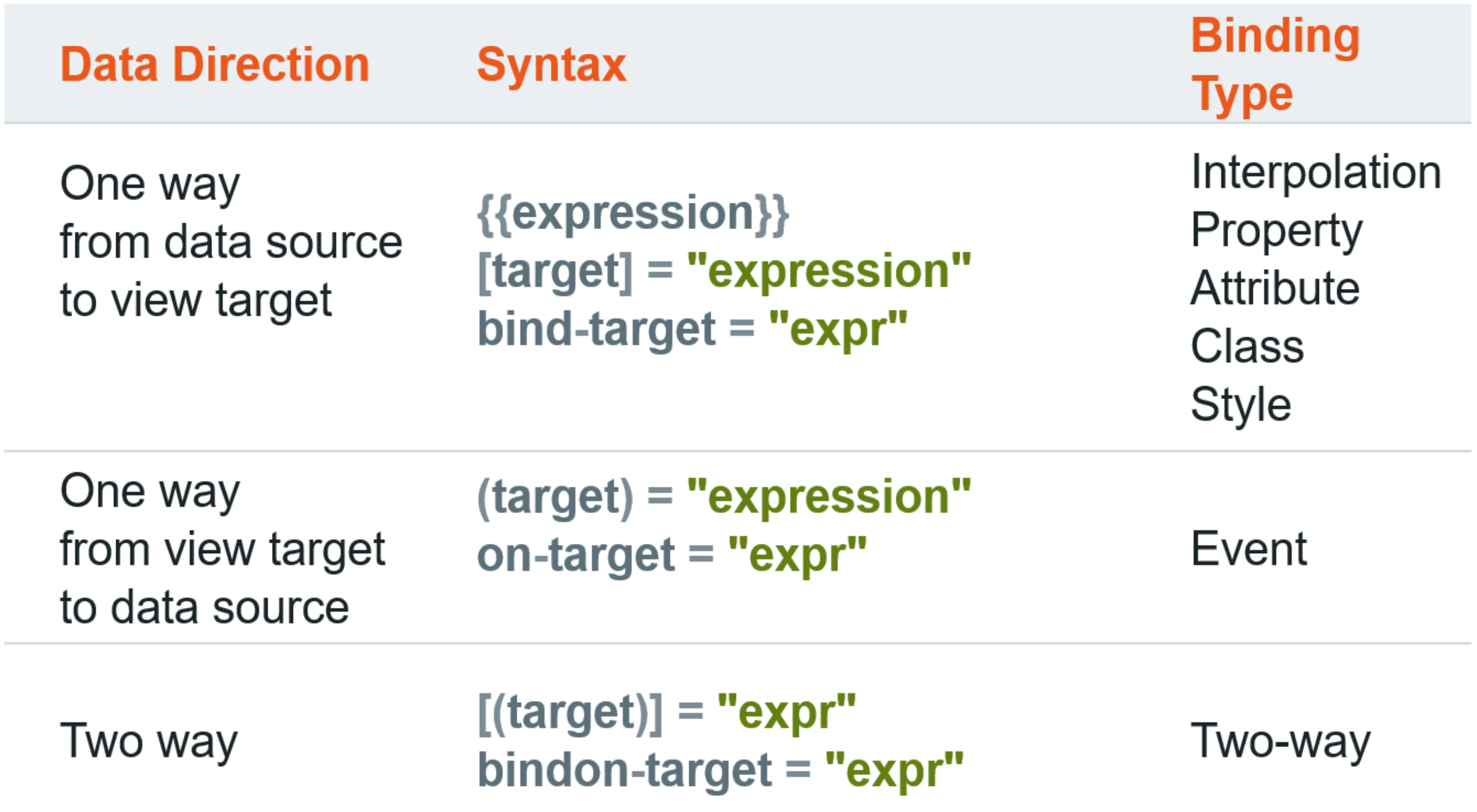
Using filter pipes

```
<!-- Search box -->
<input type="text" [(ngModel)]="searchText" placeholder="Search
Category" />

<!-- List of categories -->
<ul>
  <li *ngFor="let item of categories | filter : searchText : 'categoryName’ ">
              {{item.categoryName}}
  </li>
</ul>
```

Binding

Binding in templates



Data Direction	Syntax	Binding Type

One way
from data source
to view target
	{{expression}}
	[target] = "expression"
	bind-target = "expr"
		Interpolation
		Property
		Attribute
		Class
		Style

One way from view target to data source	(target) = "expression" on-target = "expr"	Event
Two way	[(target)] = "expr" bindon-target = "expr"	Two-way

Binding Targets

Binding Type	Target	Example
Property	Element Property	**<img [src]='property'>**
	Component property	**<myapp [cat]="cat"/>**
	Directive Property	**<div [ngClass]={s1:isSelected}**
Event	Element Event	**<button (Click)="save()">**
	Component Event	**<myapp (click)="save()">**
	Directive Event	**<div myClick (myClick) ="clicked=$event">**
Two Way	Property	***<input [(ngModel)]="property"/>***