

Iterate over object in Angular

Asked 4 years, 3 months ago Active 2 months ago Viewed 155k times



108



32

I am trying to do some things in Angular 2 Alpha 28, and am having an issue with dictionaries and NgFor.

I have an interface in TypeScript looking like this:

```
interface Dictionary {  
    [ index: string ]: string  
}
```

In JavaScript this will translate to an object that with data might look like this:

```
myDict={'key1':'value1','key2':'value2'}
```

I want to iterate over this and tried this:

```
<div *ngFor="(#key, #value) of myDict">{{key}}:{{value}}</div>
```

But to no avail, none of the below worked either:

```
<div *ngFor="#value of myDict">{{value}}</div>  
<div *ngFor="#value of myDict #key=index">{{key}}:{{value}}</div>
```

In all cases I get errors like "Unexpected token" or "Cannot find 'iterableDiff' pipe supporting object"

What am I missing here? Is this not possible anymore? (The first syntax works in Angular 1.x) or is the syntax different for iterating over an object?



edited Jun 29 '17 at 13:05

user663031

asked Jul 18 '15 at 11:30



[Rickard Staaf](#)

1,325 2 10 13

What is a "dictionary"? I've never seen or heard that term in a JavaScript, Angular, or TypeScript context. Y – user663031 Jun 29 '17 at 13:05

Dictionary means a map I think, the term is not used at all in JS context but in Python or Ruby it does used. – Cesar Jr Rodriguez Aug 29 '18 at 20:10

2 I think @bersling answer is now the correct answer to this question. – Joshua Kissoon Nov 12 '18 at 2:20

1 Please mark the correct answer better. bersling is correct – [activedecay](#) Jan 18 at 17:55

17 Answers



It appears they do not want to support the syntax from ng1.

79

According to Miško Hevery ([reference](#)):



Maps have no orders in keys and hence they iteration is unpredictable. This was supported in ng1, but we think it was a mistake and will not be supported in NG2



The plan is to have a mapToIterable pipe

```
<div *ngFor="var item of map | mapToIterable">
```

So in order to iterate over your object you will need to use a "pipe". Currently there is no [pipe](#) implemented that does that.

As a workaround, here is a small example that iterates over the keys:

Component:

```
import {Component} from 'angular2/core';

@Component({
  selector: 'component',
  templateUrl: `
    <ul>
      <li *ngFor="#key of keys();">{{key}}:{{myDict[key]}}</li>
    </ul>
  `
})
export class Home {
  myDict : Dictionary;
```

```

constructor() {
  this.myDict = {'key1':'value1','key2':'value2'};
}

keys() : Array<string> {
  return Object.keys(this.myDict);
}
}

interface Dictionary {
  [ index: string ]: string
}

```

edited Apr 18 '16 at 6:29



Günter Zöchbauer

374k 88 1205 1090

answered Jul 21 '15 at 11:22



Jesse Good

39.7k 10 93 145

1 i am trying same on object with key as number and value as string but angular is throwing error expression has changed after it was checked ? why so any idea ? – [Pardeep Jain](#) Jan 19 '16 at 9:49

Yeah this is happening for me too. And same if I use @obsur's solution too. – [user2294382](#) Feb 1 '16 at 17:01

Please see bersling's answer because there is a trivial solution on the latest angular 7 – [activedecay](#) Jan 18 at 17:56

Angular 6.1.0+ Answer

112 Use the built-in [keyvalue -pipe](#) like this:

```

<div *ngFor="let item of myObject | keyvalue">
  Key: <b>{{item.key}}</b> and Value: <b>{{item.value}}</b>
</div>

```

or like this:

```

<div *ngFor="let item of myObject | keyvalue:mySortingFunction">
  Key: <b>{{item.key}}</b> and Value: <b>{{item.value}}</b>
</div>

```

where `mySortingFunction` is in your `.ts` file, for example:

```
mySortingFunction = (a, b) => {  
  return a.key > b.key ? -1 : 1;  
}
```

Stackblitz: <https://stackblitz.com/edit/angular-iterate-key-value>

You won't need to register this in any module, since Angular pipes work out of the box in any template.

It also works for [Javascript-Maps](#).

Pre-Angular 6 Answer

As the other answers have mentioned, it's not supported in ngx, so here's a workaround with **key-value-pairs**:

The pipe:

```
import { Pipe, PipeTransform } from '@angular/core';  
@Pipe({  
  name: 'mapToIterable'  
})  
export class MapToIterable implements PipeTransform {  
  transform(dict: Object) {  
    var a = [];  
    for (var key in dict) {  
      if (dict.hasOwnProperty(key)) {  
        a.push({key: key, val: dict[key]});  
      }  
    }  
    return a;  
  }  
}
```

The usage:

```
<div *ngFor="let keyValuePair of someObject | mapToIterable">  
  This is the key {{keyValuePair.key}} and this is the value {{keyValuePair.val}}.  
</div>
```

Stackblitz Example: <https://stackblitz.com/edit/map-to-iterable-pipe>



bersling

7,347 3 27 34

You should add `implements PipeTransform` in the class definition (see angular.io/guide/pipes#custom-pipes) – [toioski](#) Mar 9 '18 at 16:39

1 @toioski thanks, i've added it and updated to the new syntax of the for loop. – [bersling](#) Mar 9 '18 at 17:29

Great answer, used this to `ngFor` my Dictionary. I had to do `keyValuePair.val[0]` though as my values ended up `[{}]` and not `{}` – [jhhoff02](#) Jun 4 '18 at 19:13

Is there an advantage to this over just `return Object.keys(dict).map(key => ({key, val: dict[key]}))` ? – [Justin Morgan](#) Sep 14 '18 at 15:28

1 this should have been marked as correct answer – [Reza](#) Nov 7 '18 at 17:19

try to use this pipe

71

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

@Pipe({ name: 'values', pure: false })
export class ValuesPipe implements PipeTransform {
  transform(value: any, args: any[] = null): any {
    return Object.keys(value).map(key => value[key]);
  }
}
```

```
<div *ngFor="#value of object | values"> </div>
```

edited Oct 11 '18 at 8:08



aloisdg

11.3k 2 51 65

answered Dec 3 '15 at 19:26



obscur

711 5 2

5 Brilliant, and if I want to keep the reference to the key I will just map an object with both key and value instead. I wish I could mark several answers as accepted answer, since this is the solution to my problem whilst the marked answer is the answer to my question. – [Rickard Staaf](#) Dec 5 '15 at 7:03

1 @obscur - If I do the above now, I get an error "expression has changed after it was checked" using `angular2.beta.0.0`. Any thoughts? – [user2294382](#) Feb 1 '16 at 23:13

Thats because `pure: false` requires a `change detectionStrategy` to be injected – [Judson Terrell](#) May 5 '16 at 4:13

1 Why setting it to impure? – [tom10271](#) Dec 8 '16 at 9:22

This worked well for me. Only thing was I could not use # in the ngFor. Used let instead. – [MartinJH](#) Sep 10 '18 at 9:26

▲ In addition to @obscur's answer, here is an example of how you can access both the `key` and `value` from the `@View`.

19

▼ Pipe:

```
@Pipe({
  name: 'keyValueFilter'
})

export class keyValueFilterPipe {
  transform(value: any, args: any[] = null): any {

    return Object.keys(value).map(function(key) {
      let pair = {};
      let k = 'key';
      let v = 'value'

      pair[k] = key;
      pair[v] = value[key];

      return pair;
    });
  }
}
```

View:

```
<li *ngFor="let u of myObject |
keyValueFilter">First Name: {{u.key}} <br> Last Name: {{u.value}}</li>
```

So if the object were to look like:

```
myObject = {
  Daario: Naharis,
  Victarion: Greyjoy,
  Quentyn: Ball
}
```

The generated outcome would be:

First name: Daario
Last Name: Naharis

First name: Victarion
Last Name: Greyjoy

First name: Quentyn
Last Name: Ball

edited Apr 4 at 17:47

answered Mar 3 '16 at 18:14



[SimonHawesome](#)
1,068 2 12 18

2 just one thing to mention you need to change the View: as `<li *ngFor="let u of myObject | keyValueFilter">First Name: {{u.key}}
 Last Name: {{u.value}}` . +1 from me. – [sib10](#) Jul 27 '17 at 19:04 ✎

The code inside your map function could be simplified as: `return { 'key' : key, 'value' : value[key] };` – [Makotosan](#) Apr 9 at 13:33

13 Adding to SimonHawesome's [excellent answer](#). I've made an succinct version which utilizes some of the new typescript features. I realize that SimonHawesome's version is intentionally verbose as to explain the underlying details. I've also added an early-out check so that the pipe works for [falsy](#) values. E.g., if the map is `null` .

Note that using a iterator transform (as done here) can be more efficient since we do not need to allocate memory for a temporary array (as done in some of the other answers).

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

@Pipe({
  name: 'mapToIterable'
})
export class MapToIterable implements PipeTransform {
  transform(map: { [key: string]: any }, ...parameters: any[]) {
    if (!map)
      return undefined;
    return Object.keys(map)
      .map((key) => ({ 'key': key, 'value': map[key] }));
  }
}
```

edited Jun 29 '17 at 12:53

answered May 27 '16 at 9:15



Frederik Aalund

1,000 2 12 15

-
- 3 love this thread, with one comment building ontop of another! I was about to write the same thing when I saw your code – [David](#) Jul 1 '16 at 13:42
-
- 3 the only thing in this solution: it should implement `PipeTransform` – [iRaS](#) Oct 15 '16 at 8:08
-
- @iRaS Good point. I've updated my answer. I also return undefined instead of null. – [Frederik Aalund](#) Jun 29 '17 at 12:54
-

Here's a variation on some of the above answers that supports multiple transforms (keyval, key, value):

9

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

type Args = 'keyval' | 'key' | 'value';

@Pipe({
  name: 'mapToIterable',
  pure: false
})
export class MapToIterablePipe implements PipeTransform {
  transform(obj: {}, arg: Args = 'keyval') {
    return arg === 'keyval' ?
      Object.keys(obj).map(key => ({key: key, value: obj[key]})) :
    arg === 'key' ?
      Object.keys(obj) :
    arg === 'value' ?
      Object.keys(obj).map(key => obj[key]) :
    null;
  }
}
```

Usage

```
map = {
  'a': 'aee',
  'b': 'bee',
  'c': 'see'
}
```



```

<div *ngFor="let o of map | mapToIterable">{{o.key}}: {{o.value}}</div>
  <div>a: aee</div>
  <div>b: bee</div>
  <div>c: see</div>

<div *ngFor="let o of map | mapToIterable:'keyval'">{{o.key}}: {{o.value}}</div>
  <div>a: aee</div>
  <div>b: bee</div>
  <div>c: see</div>

<div *ngFor="let k of map | mapToIterable:'key'">{{k}}</div>
  <div>a</div>
  <div>b</div>
  <div>c</div>

<div *ngFor="let v of map | mapToIterable:'value'">{{v}}</div>
  <div>aee</div>
  <div>bee</div>
  <div>see</div>

```

answered Jan 11 '17 at 22:32



t.888

2,955

3

17

30

1 pure: false is really important for instant reflections. – [Fırat KÜÇÜK](#) Apr 15 '17 at 16:47

Updated : Angular is now providing the pipe for lopping through the json Object via `keyvalue` :

8

```

<div *ngFor="let item of myDict | keyvalue">
  {{item.key}}:{{item.value}}
</div>

```

[WORKING DEMO](#) , and for more detail [Read](#)

Previously (For Older Version) : Till now the best / shortest answer I found is (Without any Pipe Filter or Custom function from Component Side)

Component side :

```
objectKeys = Object.keys;
```

Template side :

```
<div *ngFor='let key of objectKeys(jsonObj)'\>
  Key: {{key}}

  <div *ngFor='let obj of jsonObj[key]'\>
    {{ obj.title }}
    {{ obj.desc }}
  </div>
</div>
```

WORKING DEMO

edited Nov 16 '18 at 14:40

answered Feb 9 '18 at 6:26



Vivek Doshi

28k 4 53 66

let item of myDict | keyValue this solved my problem. – [silambarasan R.D](#) 2 days ago

I had a similar issue, built something for objects and Maps.

4

```
import { Pipe } from 'angular2/core.js';

/**
 * Map to Iterable Pipe
 *
 * It accepts Objects and [Maps](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map)
 *
 * Example:
 *
 * <div *ngFor="#keyValuePair of someObject | mapToIterable">
 *   key {{keyValuePair.key}} and value {{keyValuePair.value}}
 * </div>
 */
```

```
*/
@Pipe({ name: 'mapToIterable' })
export class MapToIterable {
  transform(value) {
    let result = [];

    if(value.entries) {
      for (var [key, value] of value.entries()) {
        result.push({ key, value });
      }
    } else {
      for(let key in value) {
        result.push({ key, value: value[key] });
      }
    }

    return result;
  }
}
```

[Run code snippet](#)[Copy snippet to answer](#)[Expand snippet](#)

answered Apr 13 '16 at 23:27

[amcdnl](#)

4,551 12 50 86

1 This works good, except that in TypeScript you should add `implements PipeTransform` to the class definition – [GeorgDangl](#) Jul 12 '16 at 12:15

Angular 2.x && Angular 4.x do not support this out of the box

3

You can use this two pipes to iterate either by **key** or by **value**.

Keys pipe:

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

@Pipe({
  name: 'keys',
  pure: false
```

```
  })
  export class KeysPipe implements PipeTransform {
    transform(value: any, args: any[] = null): any {
      return Object.keys(value)
    }
  }
}
```

Values pipe:

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

@Pipe({
  name: 'values',
  pure: false
})
export class ValuesPipe implements PipeTransform {
  transform(value: any, args: any[] = null): any {
    return Object.keys(value).map(key => value[key])
  }
}
```

How to use:

```
let data = {key1: 'value1', key2: 'value2'}

<div *ngFor="let key of data | keys"></div>
<div *ngFor="let value of data | values"></div>
```

answered May 30 '17 at 6:15



M K

4,854

4

33

39



2



I have been tearing my hair out with trying to parse and use data returned from a JSON query/ api call. Im not sure exactly where i was going wrong, i feel like i have been circling the answer for days, chasing various error codes like:

"Cannot find 'iterableDiff' pipe supporting object"

"Generic TYPe Array requires one argument(s)"

JSON parsing Errors, and im sure others

Im assuming i just had the wrong combination of fixes.

So here's a bit of a summary of gotchas and things to look for.

Firstly check the result of your api calls, your results may be in the form of an object, an array, or an array of objects.

i wont go into it too much, suffice to say the OP's original Error of not being iterable is generally caused by you trying to iterate an object, not an Array.

[Heres some of my debugging results showing variables of both arrays and objects](#)

So as we generally would like to iterate over our JSON result we need to ensure it is in the form of an Array. I tried numerous examples, and perhaps knowing what i know now some of those would in fact work, but the approach i went with was indeed to implement a pipe and the code i used was that the posted by t.888

```
transform(obj: {[key: string]: any}, arg: string) {  
  if (!obj)  
    return undefined;  
  
  return arg === 'keyval' ?  
    Object.keys(obj).map((key) => ({ 'key': key, 'value': obj[key] }))) :  
    arg === 'key' ?  
      Object.keys(obj) :  
      arg === 'value' ?  
        Object.keys(obj).map(key => obj[key]) :  
        null;  
}
```

Honestly i think one of the things that was getting me was the lack of error handling, by adding the 'return undefined' call i believe we are now allowing for non expected data to be sent to the pipe, which obviously was occurring in my case.

if you don't want to deal with argument to the pipe (and look i don't think it's necessary in most cases) you can just return the following

```
if (!obj)  
  return undefined;  
return Object.keys(obj);
```

Some Notes on creating your pipe and page or component that uses that pipe

is i was receiving errors about 'name_of_my_pipe' not being found

Use the 'ionic generate pipe' command from the CLI to ensure the pipe modules.ts are created and referenced correctly. ensure you add the following to the mypage.module.ts page.

```
import { PipesModule } from '.../.../pipes/pipes.module';
```

(not sure if this changes if you also have your own custom_module, you may also need to add it to the custommodule.module.ts)

if you used the 'ionic generate page' command to make your page, but decide to use that page as your main page, remember to remove the page reference from app.module.ts (here's another answer i posted dealing with that <https://forum.ionicframework.com/t/solved-pipe-not-found-in-custom-component/95179/13?u=dreaser>)

In my searching for answers there where a number of ways to display the data in the html file, and i don't understand enough to explain the differences. You may find it better to use one over another in certain scenarios.

```
<ion-item *ngFor="let myPost of posts">
  
  
  <img [src]='https://somewhereOnTheInternet/' + myPost.ImageUrl' />
</ion-item>
```

However what worked that allowed me to display both the value and the key was the following:

```
<ion-list>
  <ion-item *ngFor="let myPost of posts | name_of_pip:'optional_Str_Variable'">

    <h2>Key Value = {{posts[myPost]}}

    <h2>Key Name = {{myPost}} </h2>

  </ion-item>
</ion-list>
```

to make the API call it looks like you need to import HttpModule into app.module.ts

```
import { HttpModule } from '@angular/http';
.
.
imports: [
  BrowserModule,
  HttpModule,
```

and you need Http in the page you make the call from

```
import {Http} from '@angular/http';
```

When making the API call you seem to be able to get to the children data (the objects or arrays within the array) 2 different ways, either seem to work

either during the call

```
this.http.get('https://SomeWebsiteWithAPI').map(res =>
  res.json().anyChildren.OrSubChildren).subscribe(
    myData => {
```

or when you assign the data to your local variable

```
posts: Array<String>;
this.posts = myData['anyChildren'];
```

(not sure if that variable needs to be an Array String, but thats what i have it at now. It may work as a more generic variable)

And final note, it was not necessary to use the inbuilt JSON library however you may find these 2 calls handy for converting from an object to a string and vica versa

```
var stringifiedData = JSON.stringify(this.movies);
console.log("***mResults in Stringify");
console.log(stringifiedData);

var mResults = JSON.parse(<string>stringifiedData);
console.log("***mResults in a JSON");
console.log(mResults);
```

I hope this compilation of info helps someone out.

edited Jan 7 '18 at 8:07

answered Jan 7 '18 at 7:24



Dreaser

31 2



If someone is wondering how to work with multidimensional object, here is the solution.

2 lets assume we have following object in **service**

```
getChallenges() {
  var objects = {};
  objects['0'] = {
    title: 'Angular2',
    description : "Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur."
  };

  objects['1'] = {
    title: 'AngularJS',
    description : "Lorem Ipsum is simply dummy text of the printing and typesetting
industry."
  };

  objects['2'] = {
    title: 'Bootstrap',
    description : "Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.",
  };
  return objects;
}
```

in component add following function

```
challenges;

constructor(testService : TestService){
  this.challenges = testService.getChallenges();
}
keys() : Array<string> {
  return Object.keys(this.challenges);
}
```

finally in view do following

```
<div *ngFor="#key of keys();">
  <h4 class="heading">{{challenges[key].title}}</h4>
  <p class="description">{{challenges[key].description}}</p>
</div>
```

answered Jan 8 '17 at 20:32

▲ The dictionary is an object, not an array. I believe ng-repeat requires an array in Angular 2.

1

▼ The simplest solution would be to create a pipe/filter that converts the object to an array on the fly. That said, you probably want to use an array as @basarat says.

answered Jul 19 '15 at 12:59



Martin

11.5k 3 34 50

▲ If you have es6-shim or your tsconfig.json target es6 , you could use ES6 [Map](#) to make it.

1

▼

```
var myDict = new Map();  
myDict.set('key1','value1');  
myDict.set('key2','value2');
```

```
<div *ngFor="let keyVal of myDict.entries()">  
  key:{{keyVal[0]}}, val:{{keyVal[1]}}  
</div>
```

answered Sep 21 '16 at 4:47



Sing

2,627 2 17 34

▲ In JavaScript this will translate to an object that with data might look like this

1


▼ Interfaces in TypeScript are a dev time construct (purely for tooling ... 0 runtime impact). You should write the same TypeScript as your JavaScript.

answered Jul 19 '15 at 12:49



basarat

156k 29 295 397

that's not true, sry. type script forces you to write cleaner code. its much easier to abstract classes. which you simply don't have. C++ compiles to some asm - asm also has no classes or even types, nevertheless you write different c++ then ur asm code :P – YAMM Jan 16 '16 at 15:45 

Define the `MapValuesPipe` and implement [PipeTransform](#):

0

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

@Pipe({name: 'mapValuesPipe'})
export class MapValuesPipe implements PipeTransform {
  transform(value: any, args?: any[]): Object[] {
    let mArray:
    value.forEach((key, val) => {
      mArray.push({
        mKey: key,
        mValue: val
      });
    });

    return mArray;
  }
}
```

Add your pipe in your pipes module. This is important if you need to use the [same pipe in more than one components](#):

```
@NgModule({
  imports: [
    CommonModule
  ],
  exports: [
    ...
    MapValuesPipe
  ],
  declarations: [..., MapValuesPipe, ...]
})
export class PipesAggrModule {}
```

Then simply use the pipe in your html with `*ngFor` :

```
<tr *ngFor="let attribute of mMap | mapValuesPipe">
```

Remember, you will need to declare your PipesModule in the component where you want to use the pipe:

```
@NgModule({
  imports: [
    CommonModule,
    PipesAggrModule
  ],
  ...
})
export class MyModule {}
```

edited May 12 '18 at 2:27

answered May 11 '18 at 0:39



Menelaos Kotsollaris

3,191 5 38 54

0

```
//Get solution for ng-repeat
//Add variable and assign with Object.key

export class TestComponent implements OnInit{
  objectKeys = Object.keys;
  obj: object = {
    "test": "value"
    "test1": "value1"
  }
}
//HTML
<div *ngFor="let key of objectKeys(obj)">
  <div>
    <div class="content">{{key}}</div>
    <div class="content">{{obj[key]}}</div>
  </div>
```

answered Apr 17 at 15:30



Rohit Jangid

191 2 5

So I was going to implement my own helper function, `objLength(obj)`, which returns just `Object(obj).keys.length`. But then when I was adding it to my template `*ngIf` function, my IDE suggested `objectKeys()`. I tried it, and it worked. Following it to its declaration, it appears

0 to be offered by lib.es5.d.ts, so there you go!

Here's how I implemented it (I have a custom object that uses server-side generated keys as an index for files I've uploaded):

```
<div *ngIf="fileList !== undefined && objectKeys(fileList).length > 0">
  <h6>Attached Files</h6>
  <table cellpadding="0" cellspacing="0">
    <tr *ngFor="let file of fileList | keyvalue">
      <td><a href="#">{{file.value['fileName']}}</a></td>
      <td class="actions">
        <a title="Delete File" (click)="deleteAFile(file.key);">
          </a>
        </td>
      </tr>
    </table>
  </div>
```

answered Apr 23 at 19:19



[mausolos](#)

46 3